SCHOOL OF COMPUTER ENGINEERING

Winter Semester - 2021

PROJECT REPORT

CSE3020 – DATA VISUALIZATION

Analysis of Road Traffic Casualties Using DV Techniques

TEAM NO: 16

19BCE2077	**Neil Kavalakkat**
19BCE0521	**Aman Anand**

Submitted to

Dr. Archana Tamizharasan

SCOPE

Project at a Glance

No. of objectives considered:

14

Language used:

Python3

Statistical Measures used:

1. **K-Mean Clustering**
2. **Naive Bayes Classification**
3. **Decision Tree Classification**

Library(s) used:

1. **pandas**
2. **matplotlib (pyplot)**
3. **numpy**
4. **seaborn**
5. **sklearn**

Total no. of Visuals created:	**20**
No of Individual Chart types used:	**7**
Bar chart	**5**
Line chart	**4**

Heatmap	**3**
Countplot	**3**
Decision Tree	**2**
Pie chart	**1**
Connected Scatterplot	**1**
Boxplot	**1**
Total Charts in project	**20**

1.1 Project Statement

With an exponential rise in population and growth in technology, most people now have access to the luxury of mobility. Road Traffic is increasing day by day and so are accidents. People neglect to take proper precautions and don't follow rules. Also, most of the time these accidents do not get notified to the concerned authority during the golden hour when the victims' lives can be saved. This leads to a rise in the number of deaths due to road accidents.

Roadway traffic safety is a major concern for transportation agencies as well as citizens. To tackle this problem, there is a need for an intricate analysis of road accidents that have occurred in the past, to understand trends and other details pertaining to them. To give safe driving suggestions, we plan to do a careful analysis of roadway traffic data that is critical to find out variables that are closely related to fatal accidents.

1.2 Project Objective

Through **Analysis of Road Traffic Casualties**, we aim to apply statistical analysis and data visualization algorithms on the Car Accident dataset in an attempt to address the above problem.

We also intend to create a Machine Learning model that can perform a predictive forecast to further analyze the dataset and make predictions on the severity of an accident.

We plan to make use of algorithms to discover association laws and hence create a classification model, on which the K-means clustering algorithm will be applied to create clusters. Based on the results of the culminated statistics, association guidelines, classification model, and clusters collected, some safety driving recommendations for increasing safety will be made.

After collecting actionable information from the above analysis, we aim to employ data visualization techniques to create relevant and observable statistics around our analysis. We will use in-built Python modules to achieve the goal as well as employ some indispensable tools present in the Spyder application.

Even though the data collection is limited to a few select attributes, our methodology will derive some valuable and assertable knowledge from it, which can be used to hence make preventive efforts in these areas.

1.3 Modules

1.3.1 Preparation Module: Collection of the dataset and understanding the dataset. Cleaning, validating and structuring the data in the dataset.

1.3.2 Statistical Analysis Module: Basic statistical analysis on the dataset to gain a more comprehensive and cohesive understanding.

1.3.3 Visualization Module: Applying Data Visualization techniques on the dataset to obtain cogent subjective conclusions for the same

1.3.4 Forecasting Module: Using a Machine Learning model to perform predictive forecasting measures on the dataset to predict severity accidents and relevant details.

1.4 Code with Visuals

Code with Visuals

May 28, 2021

1 Code with Visuals

1.1 Introduction to the dataset and cleaning the data

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.tree import export_graphviz
from six import StringIO
from IPython.display import Image
import pydotplus
```

[2]:
```
casual_df=pd.read_csv('Casualties0514.csv')
accident_df=pd.read_csv('Accidents0514.csv')
vehicle_df=pd.read_csv('Vehicles0514.csv')
```

```
C:\Anaconda\lib\site-packages\IPython\core\interactiveshell.py:3155:
DtypeWarning: Columns (31) have mixed types.Specify dtype option on import or
set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

[3]:
```
casual_df.head()
```

[3]:

	Accident_Index	Vehicle_Reference	Casualty_Reference	Casualty_Class	Sex_of_Casualty	Age_of_Casualty	Age_Band_of_Casualty	Casualty_Severity
0	200501BS00001	1	1	3	1	37	7	2
1	200501BS00002	1	1	2	1	37	7	3
2	200501BS00003	2	1	1				
3	200501BS00004	1	1	3				
4	200501BS00005	1	1	1				

```
2                1              62                    9                   3
3                1              30                    6                   3
4                1              49                    8                   3

   Pedestrian_Location  Pedestrian_Movement  Car_Passenger  \
0                    1                    1              0
1                    0                    0              0
2                    0                    0              0
3                    5                    2              0
4                    0                    0              0

   Bus_or_Coach_Passenger  Pedestrian_Road_Maintenance_Worker  Casualty_Type  \
0                       0                                 -1              0
1                       4                                 -1             11
2                       0                                 -1              9
3                       0                                 -1              0
4                       0                                 -1              3

   Casualty_Home_Area_Type
0                        1
1                        1
2                        1
3                        1
4                       -1
```

[4]:
```
accident_df.head()
```

[4]:
```
   Accident_Index  Location_Easting_OSGR  Location_Northing_OSGR  Longitude  \
0  200501BS00001               525680.0                178240.0  -0.191170
1  200501BS00002               524170.0                181650.0  -0.211708
2  200501BS00003               524520.0                182240.0  -0.206458
3  200501BS00004               526900.0                177530.0  -0.173862
4  200501BS00005               528060.0                179040.0  -0.156618

    Latitude  Police_Force  Accident_Severity  Number_of_Vehicles  \
0  51.489096             1                  2                   1
1  51.520075             1                  3                   1
2  51.525301             1                  3                   2
3  51.482442             1                  3                   1
4  51.495752             1                  3                   1

   Number_of_Casualties        Date  …  Pedestrian_Crossing-Human_Control  \
0                     1  04/01/2005  …                                  0
1                     1  05/01/2005  …                                  0
2                     1  06/01/2005  …                                  0
3                     1  07/01/2005  …                                  0
4                     1  10/01/2005  …                                  0
```

```
  Pedestrian_Crossing-Physical_Facilities  Light_Conditions  \
0                                       1                 1
1                                       5                 4
2                                       0                 4
3                                       0                 1
4                                       0                 7

  Weather_Conditions  Road_Surface_Conditions  Special_Conditions_at_Site  \
0                  2                       2                           0
1                  1                       1                           0
2                  1                       1                           0
3                  1                       1                           0
4                  1                       2                           0

   Carriageway_Hazards  Urban_or_Rural_Area  \
0                    0                    1
1                    0                    1
2                    0                    1
3                    0                    1
4                    0                    1

   Did_Police_Officer_Attend_Scene_of_Accident  LSOA_of_Accident_Location
0                                            1                 E01002849
1                                            1                 E01002909
2                                            1                 E01002857
3                                            1                 E01002840
4                                            1                 E01002863

[5 rows x 32 columns]
```

[5]:
```
vehicle_df.head()
```

[5]:
```
   Accident_Index  Vehicle_Reference  Vehicle_Type  Towing_and_Articulation  \
0   200501BS00001                  1             9                        0
1   200501BS00002                  1            11                        0
2   200501BS00003                  1            11                        0
3   200501BS00003                  2             9                        0
4   200501BS00004                  1             9                        0

   Vehicle_Manoeuvre  Vehicle_Location-Restricted_Lane  Junction_Location  \
0                 18                                0                  0
1                  4                                0                  3
2                 17                                0                  0
3                  2                                0                  0
4                 18                                0                  0
```

```
   Skidding_and_Overturning  Hit_Object_in_Carriageway  \
0                         0                          0
1                         0                          0
2                         0                          4
3                         0                          0
4                         0                          0

   Vehicle_Leaving_Carriageway  …  Was_Vehicle_Left_Hand_Drive?  \
0                            0  …                             1
1                            0  …                             1
2                            0  …                             1
3                            0  …                             1
4                            0  …                             1

   Journey_Purpose_of_Driver  Sex_of_Driver  Age_of_Driver  \
0                         15              2             74
1                          1              1             42
2                          1              1             35
3                         15              1             62
4                         15              2             49

   Age_Band_of_Driver  Engine_Capacity_(CC)  Propulsion_Code  Age_of_Vehicle  \
0                  10                    -1               -1              -1
1                   7                  8268                2               3
2                   6                  8300                2               5
3                   9                  1762                1               6
4                   8                  1769                1               4

   Driver_IMD_Decile  Driver_Home_Area_Type
0                  7                      1
1                 -1                     -1
2                  2                      1
3                  1                      1
4                  2                      1

[5 rows x 22 columns]
```

[6]:
```
accident_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1640597 entries, 0 to 1640596
Data columns (total 32 columns):
 #   Column                                       Non-Null Count    Dtype
---  ------                                       --------------    -----
 0   Accident_Index                               1640597 non-null  object
 1   Location_Easting_OSGR                        1640486 non-null  float64
 2   Location_Northing_OSGR                       1640486 non-null  float64
 3   Longitude                                    1640486 non-null  float64
```

```
 4   Latitude                                    1640486 non-null  float64
 5   Police_Force                                1640597 non-null  int64
 6   Accident_Severity                           1640597 non-null  int64
 7   Number_of_Vehicles                          1640597 non-null  int64
 8   Number_of_Casualties                        1640597 non-null  int64
 9   Date                                        1640597 non-null  object
 10  Day_of_Week                                 1640597 non-null  int64
 11  Time                                        1640464 non-null  object
 12  Local_Authority_(District)                  1640597 non-null  int64
 13  Local_Authority_(Highway)                   1640597 non-null  object
 14  1st_Road_Class                              1640597 non-null  int64
 15  1st_Road_Number                             1640597 non-null  int64
 16  Road_Type                                   1640597 non-null  int64
 17  Speed_limit                                 1640597 non-null  int64
 18  Junction_Detail                             1640597 non-null  int64
 19  Junction_Control                            1640597 non-null  int64
 20  2nd_Road_Class                              1640597 non-null  int64
 21  2nd_Road_Number                             1640597 non-null  int64
 22  Pedestrian_Crossing-Human_Control           1640597 non-null  int64
 23  Pedestrian_Crossing-Physical_Facilities     1640597 non-null  int64
 24  Light_Conditions                            1640597 non-null  int64
 25  Weather_Conditions                          1640597 non-null  int64
 26  Road_Surface_Conditions                     1640597 non-null  int64
 27  Special_Conditions_at_Site                  1640597 non-null  int64
 28  Carriageway_Hazards                         1640597 non-null  int64
 29  Urban_or_Rural_Area                         1640597 non-null  int64
 30  Did_Police_Officer_Attend_Scene_of_Accident 1640597 non-null  int64
 31  LSOA_of_Accident_Location                   1520023 non-null  object
dtypes: float64(4), int64(23), object(5)
memory usage: 400.5+ MB
```

[7]:
```
vehicle_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3004425 entries, 0 to 3004424
Data columns (total 22 columns):
 #   Column                          Dtype
---  ------                          -----
 0   Accident_Index                  object
 1   Vehicle_Reference               int64
 2   Vehicle_Type                    int64
 3   Towing_and_Articulation         int64
 4   Vehicle_Manoeuvre               int64
 5   Vehicle_Location-Restricted_Lane int64
 6   Junction_Location               int64
 7   Skidding_and_Overturning        int64
 8   Hit_Object_in_Carriageway       int64
 9   Vehicle_Leaving_Carriageway     int64
```

```
 10  Hit_Object_off_Carriageway        int64
 11  1st_Point_of_Impact               int64
 12  Was_Vehicle_Left_Hand_Drive?      int64
 13  Journey_Purpose_of_Driver         int64
 14  Sex_of_Driver                     int64
 15  Age_of_Driver                     int64
 16  Age_Band_of_Driver                int64
 17  Engine_Capacity_(CC)              int64
 18  Propulsion_Code                   int64
 19  Age_of_Vehicle                    int64
 20  Driver_IMD_Decile                 int64
 21  Driver_Home_Area_Type             int64
dtypes: int64(21), object(1)
memory usage: 504.3+ MB
```

[8]:
```
casual_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2216720 entries, 0 to 2216719
Data columns (total 15 columns):
 #   Column                                Dtype
---  ------                                -----
 0   Accident_Index                        object
 1   Vehicle_Reference                     int64
 2   Casualty_Reference                    int64
 3   Casualty_Class                        int64
 4   Sex_of_Casualty                       int64
 5   Age_of_Casualty                       int64
 6   Age_Band_of_Casualty                  int64
 7   Casualty_Severity                     int64
 8   Pedestrian_Location                   int64
 9   Pedestrian_Movement                   int64
 10  Car_Passenger                         int64
 11  Bus_or_Coach_Passenger                int64
 12  Pedestrian_Road_Maintenance_Worker    int64
 13  Casualty_Type                         int64
 14  Casualty_Home_Area_Type               int64
dtypes: int64(14), object(1)
memory usage: 253.7+ MB
```

[9]:
```
first_df=pd.merge(casual_df,accident_df,on='Accident_Index')
df=pd.merge(first_df,vehicle_df,on='Accident_Index')
```

[10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4287593 entries, 0 to 4287592
Data columns (total 67 columns):
 #   Column                                          Dtype
```

```
---  ------                                        -----
 0   Accident_Index                                object
 1   Vehicle_Reference_x                           int64
 2   Casualty_Reference                            int64
 3   Casualty_Class                                int64
 4   Sex_of_Casualty                               int64
 5   Age_of_Casualty                               int64
 6   Age_Band_of_Casualty                          int64
 7   Casualty_Severity                             int64
 8   Pedestrian_Location                           int64
 9   Pedestrian_Movement                           int64
 10  Car_Passenger                                 int64
 11  Bus_or_Coach_Passenger                        int64
 12  Pedestrian_Road_Maintenance_Worker            int64
 13  Casualty_Type                                 int64
 14  Casualty_Home_Area_Type                       int64
 15  Location_Easting_OSGR                         float64
 16  Location_Northing_OSGR                        float64
 17  Longitude                                     float64
 18  Latitude                                      float64
 19  Police_Force                                  int64
 20  Accident_Severity                             int64
 21  Number_of_Vehicles                            int64
 22  Number_of_Casualties                          int64
 23  Date                                          object
 24  Day_of_Week                                   int64
 25  Time                                          object
 26  Local_Authority_(District)                    int64
 27  Local_Authority_(Highway)                     object
 28  1st_Road_Class                                int64
 29  1st_Road_Number                               int64
 30  Road_Type                                     int64
 31  Speed_limit                                   int64
 32  Junction_Detail                               int64
 33  Junction_Control                              int64
 34  2nd_Road_Class                                int64
 35  2nd_Road_Number                               int64
 36  Pedestrian_Crossing-Human_Control             int64
 37  Pedestrian_Crossing-Physical_Facilities       int64
 38  Light_Conditions                              int64
 39  Weather_Conditions                            int64
 40  Road_Surface_Conditions                       int64
 41  Special_Conditions_at_Site                    int64
 42  Carriageway_Hazards                           int64
 43  Urban_or_Rural_Area                           int64
 44  Did_Police_Officer_Attend_Scene_of_Accident   int64
 45  LSOA_of_Accident_Location                     object
 46  Vehicle_Reference_y                           int64
```

```
 47  Vehicle_Type                                int64
 48  Towing_and_Articulation                     int64
 49  Vehicle_Manoeuvre                           int64
 50  Vehicle_Location-Restricted_Lane            int64
 51  Junction_Location                           int64
 52  Skidding_and_Overturning                    int64
 53  Hit_Object_in_Carriageway                   int64
 54  Vehicle_Leaving_Carriageway                 int64
 55  Hit_Object_off_Carriageway                  int64
 56  1st_Point_of_Impact                         int64
 57  Was_Vehicle_Left_Hand_Drive?                int64
 58  Journey_Purpose_of_Driver                   int64
 59  Sex_of_Driver                               int64
 60  Age_of_Driver                               int64
 61  Age_Band_of_Driver                          int64
 62  Engine_Capacity_(CC)                        int64
 63  Propulsion_Code                             int64
 64  Age_of_Vehicle                              int64
 65  Driver_IMD_Decile                           int64
 66  Driver_Home_Area_Type                       int64
dtypes: float64(4), int64(58), object(5)
memory usage: 2.2+ GB
```

[11]:
```
df.head()
```

[11]:

	Accident_Index	Vehicle_Reference_x	Casualty_Reference	Casualty_Class	\
0	200501BS00001	1	1	3	
1	200501BS00002	1	1	2	
2	200501BS00003	2	1	1	
3	200501BS00003	2	1	1	
4	200501BS00004	1	1	3	

	Sex_of_Casualty	Age_of_Casualty	Age_Band_of_Casualty	Casualty_Severity	\
0	1	37	7	2	
1	1	37	7	3	
2	1	62	9	3	
3	1	62	9	3	
4	1	30	6	3	

	Pedestrian_Location	Pedestrian_Movement	…	\
0	1	1	…	
1	0	0	…	
2	0	0	…	
3	0	0	…	
4	5	2	…	

	Was_Vehicle_Left_Hand_Drive?	Journey_Purpose_of_Driver	Sex_of_Driver	\

```
0                              1                          15                2
1                              1                           1                1
2                              1                           1                1
3                              1                          15                1
4                              1                          15                2

   Age_of_Driver  Age_Band_of_Driver  Engine_Capacity_(CC)  Propulsion_Code  \
0             74                  10                    -1               -1
1             42                   7                  8268                2
2             35                   6                  8300                2
3             62                   9                  1762                1
4             49                   8                  1769                1

   Age_of_Vehicle  Driver_IMD_Decile  Driver_Home_Area_Type
0              -1                  7                      1
1               3                 -1                     -1
2               5                  2                      1
3               6                  1                      1
4               4                  2                      1

[5 rows x 67 columns]
```

[12]:
```
df.isnull().sum()
```

[12]:
```
Accident_Index           0
Vehicle_Reference_x      0
Casualty_Reference       0
Casualty_Class           0
Sex_of_Casualty          0
                        ..
Engine_Capacity_(CC)     0
Propulsion_Code          0
Age_of_Vehicle           0
Driver_IMD_Decile        0
Driver_Home_Area_Type    0
Length: 67, dtype: int64
```

[13]:
```
df.drop('LSOA_of_Accident_Location',axis=1,inplace=True)
df.dropna(subset=['Location_Easting_OSGR','Location_Northing_OSGR',␣
 ↪'Longitude', 'Latitude'],axis=0,inplace=True)
df.dropna(subset=['Time'],axis=0,inplace=True)
```

[14]:
```
df.isnull().values.any()
```

[14]:
```
False
```

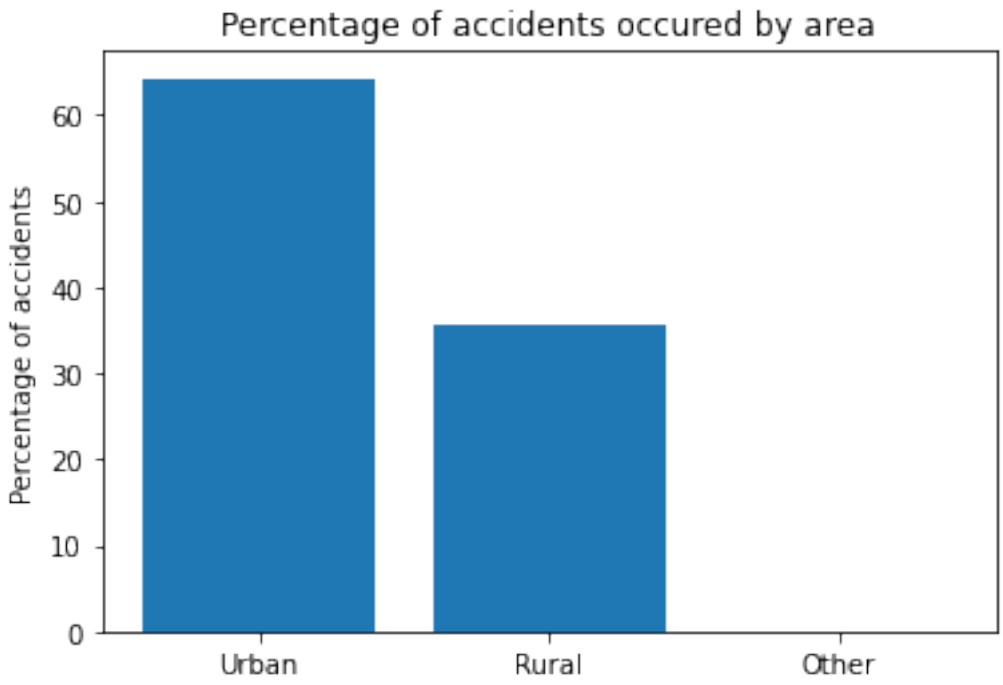
1.2 Exploring the dataset through the principal objective questions

Q1. What fraction of accidents occur in urban, rural and other (NA) areas?

Data Visualization done on individual datasets

[15]:
```
urban_acci =len(accident_df[accident_df['Urban_or_Rural_Area']==1])
rural_acci =len(accident_df[accident_df['Urban_or_Rural_Area']==2])
na_acci =len(accident_df[accident_df['Urban_or_Rural_Area']==3])
total_acci = urban_acci + rural_acci + na_acci
urban_pct = urban_acci / total_acci * 100
rural_pct = rural_acci / total_acci *100
na_pct = na_acci / total_acci * 100
print("Percentage of accidents occur in urban areas is {0:.0f}%".
 ↪format(urban_pct))
print("Percentage of accidents occur in rural areas is {0:.0f}%".
 ↪format(rural_pct))
print("Percentage of accidents occur in other areas is {0:.0f}%".format(na_pct))
x = ['Urban', 'Rural', 'Other']
y = [urban_pct, rural_pct,na_pct]
x_pos =list(range(len(x)))
plt.bar(x_pos, y)
plt.ylabel('Percentage of accidents')
plt.xticks(x_pos,x)
plt.title("Percentage of accidents occured by area")
plt.show()
```

```
Percentage of accidents occur in urban areas is 64%
Percentage of accidents occur in rural areas is 36%
Percentage of accidents occur in other areas is 0%
```

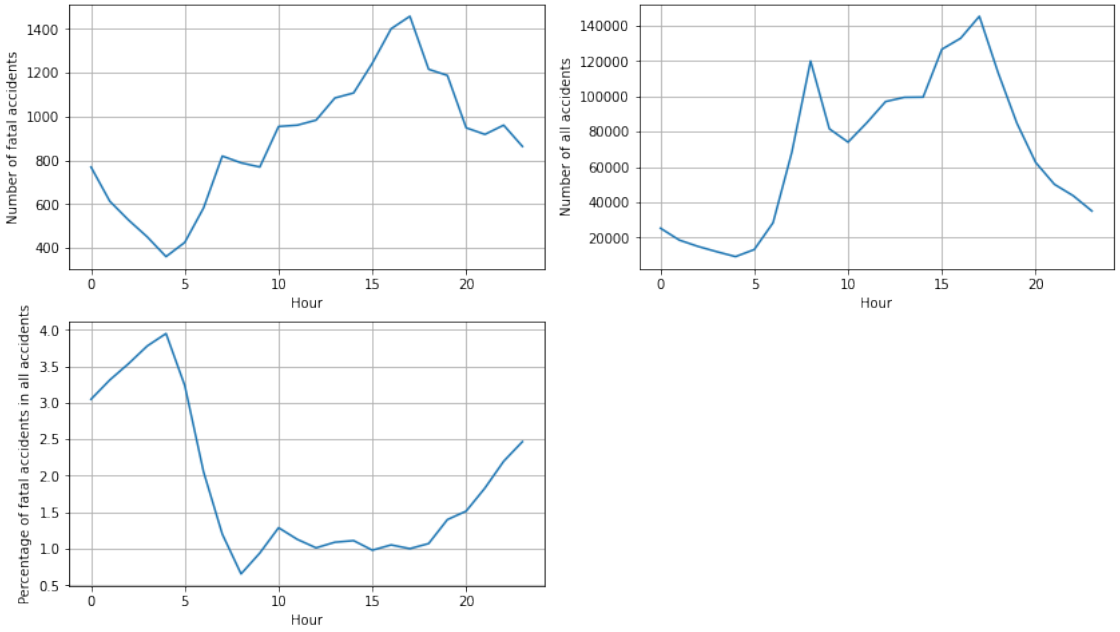


Q2. When is the most dangerous time to drive?

Data Visualization done on individual datasets

[16]:
```
accident_df['Hour'] = accident_df['Time'].map(lambda x: str(x).split(':')[0])
accident_df['Hour'] = accident_df['Hour'].apply(pd.to_numeric, errors='coerce')
hour = []
num_of_fatal_acci = []
num_of_acci = []
for i in range(24):
    hour.append(i)
    num_of_fatal_acci_hour = len(accident_df[(accident_df['Accident_Severity']␣
 ↪== 1) & (accident_df['Hour'] == i)])
    num_of_acci_hour = len(accident_df[accident_df['Hour'] == i])
    num_of_fatal_acci.append(num_of_fatal_acci_hour)
    num_of_acci.append(num_of_acci_hour)
    normalized_num_of_fatal_acci = list(np.array(num_of_fatal_acci) / np.
 ↪array(num_of_acci) *100)
fig = plt.figure(figsize=(14,8))
ax1 = fig.add_subplot(221)
ax1.plot(hour, num_of_fatal_acci)
ax1.set_ylabel('Number of fatal accidents')
ax1.set_xlabel('Hour')
ax1.grid(True)
ax2 = fig.add_subplot(222)
```

```
ax2.plot(hour, num_of_acci)
ax2.set_ylabel('Number of all accidents')
ax2.set_xlabel('Hour')
ax2.grid(True)
ax3 = fig.add_subplot(223)
ax3.plot(hour, normalized_num_of_fatal_acci)
ax3.set_ylabel('Percentage of fatal accidents in all accidents')
ax3.set_xlabel('Hour')
ax3.grid(True)
plt.show()
print("The most dangerous hour to drive, when most fatal accidents happend in␣
 ↪all accidents,is {} o'clock".format(normalized_num_of_fatal_acci.
 ↪index(max(normalized_num_of_fatal_acci))))
```



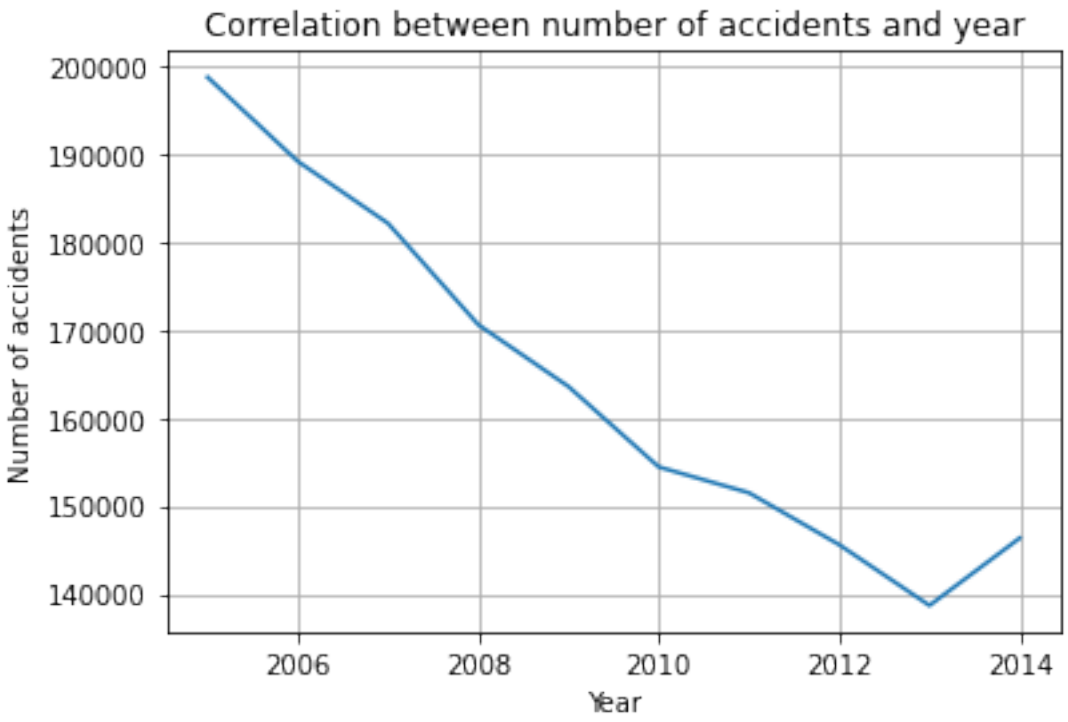

```
The most dangerous hour to drive, when most fatal accidents happend in all
accidents,is 4 o'clock
```

Q3. What is the trend in the number of accidents that occur each year?

Data Visualization done on individual datasets

[17]:
```
accident_df['Year'] = accident_df['Accident_Index'].map(lambda x: str(x)[:4])
accident_df['Year'] = accident_df['Year'].apply(pd.to_numeric, errors='coerce')
year = []
num_of_acci_year = []
for i in range(2005, 2015):
    year.append(i)
```

```
    num_of_acci_year.append(len(accident_df[accident_df['Year'] == i]))
plt.plot(year, num_of_acci_year)
plt.xlabel('Year')
plt.ylabel('Number of accidents')
plt.title('Correlation between number of accidents and year')
plt.grid(True)
plt.show()
```



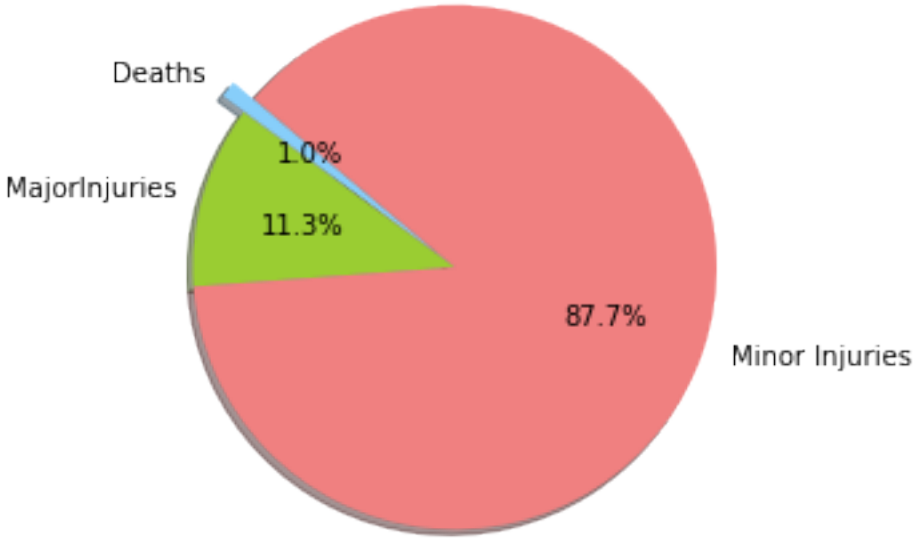
Q4. What fraction of accidents caused minor injuires, major injuries and deaths?

Data Visualization done on individual datasets

[18]:
```
Severity1 =len(casual_df[casual_df['Casualty_Severity']==1])
Severity2 =len(casual_df[casual_df['Casualty_Severity']==2])
Severity3 =len(casual_df[casual_df['Casualty_Severity']==3])
tot=Severity1+Severity2+Severity3;
s1=(Severity1/tot)*100;
s2=(Severity2/tot)*100;
s3=(Severity3/tot)*100;
print("Percentage of Deaths is {0:.0f}%".format(s1))
print("Percentage of Major Injuries is {0:.0f}%".format(s2))
print("Percentage of Minor Injuires is {0:.0f}%".format(s3))
labels = 'Deaths','MajorInjuries','Minor Injuries'
sizes = [s1, s2, s3]
```

```
colors = ['lightskyblue', 'yellowgreen', 'lightcoral']
explode = (0.1, 0, 0)
# Plot
plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.
↪1f%%', shadow=True, startangle=140)
plt.axis('equal')
plt.show()
```

```
Percentage of Deaths is 1%
Percentage of Major Injuries is 11%
Percentage of Minor Injuires is 88%
```

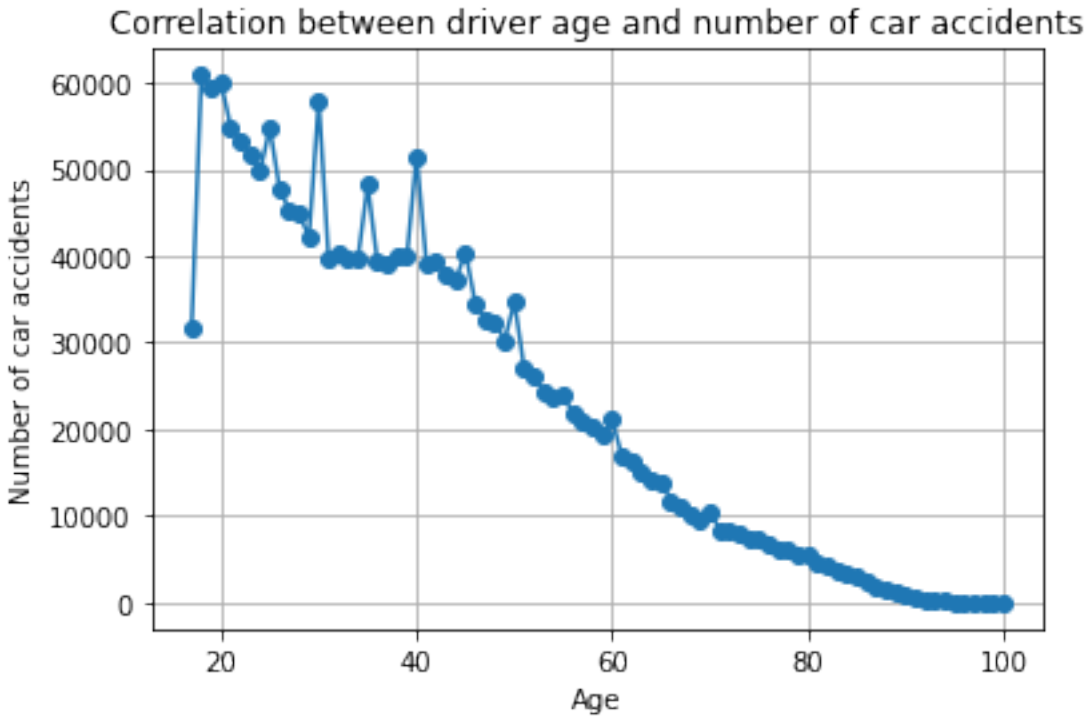


Q5. How fast do the number of car accidents drop off with age?

Data Visualization done on individual datasets

[19]:
```
age_acci = vehicle_df[['Accident_Index', 'Age_of_Driver', 'Vehicle_Type']]
age = []
num_of_acci = []
for i in range(17, max(age_acci['Age_of_Driver'])+1):
    age.append(i)
    num_of_acci.
↪append(len(age_acci[(age_acci['Age_of_Driver']==i)&(age_acci['Vehicle_Type']␣
↪== 9 )]))
plt.plot(age, num_of_acci, label = 'Data', marker = 'o')
plt.xlabel('Age')
plt.ylabel('Number of car accidents')
plt.title('Correlation between driver age and number of car accidents')
```

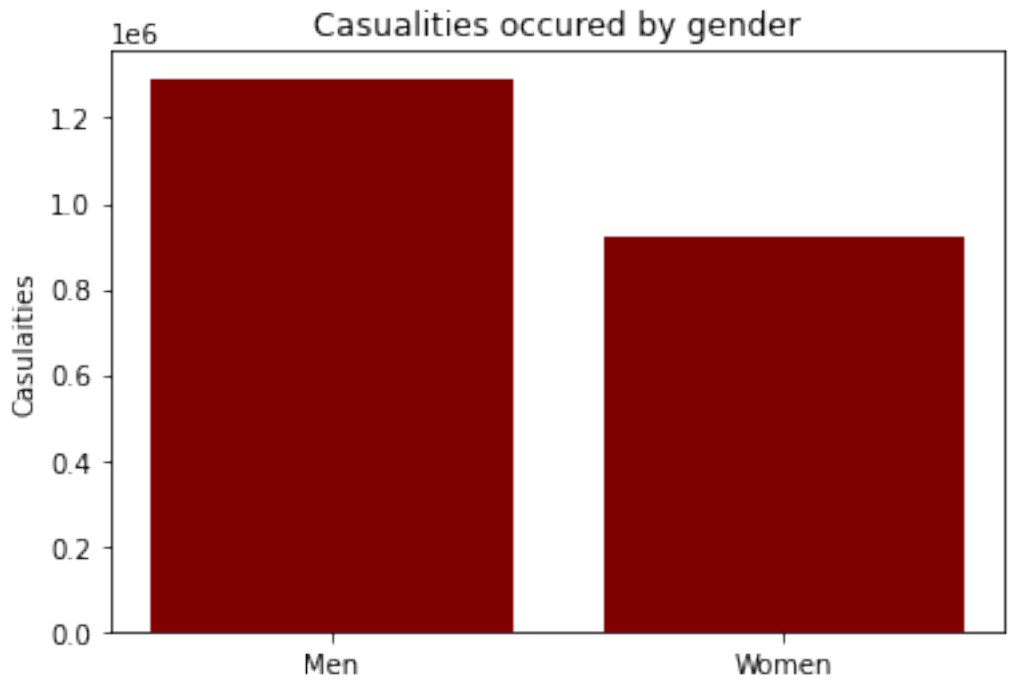
```
plt.grid(True)
plt.show()
```



Q6. What is the ratio of men and women who get injured in accidents?

Data Visualization done on individual datasets

[20]:
```
men =len(casual_df[casual_df['Sex_of_Casualty']==1])
women =len(casual_df[casual_df['Sex_of_Casualty']==2])
x = ['Men', 'Women']
y = [men,women]
x_pos =list(range(len(x)))
plt.bar(x_pos, y,color ='maroon')
plt.ylabel('Casulaities')
plt.xticks(x_pos,x)
plt.title("Casualities occured by gender")
plt.show()
```



Data Visualization done on the merged dataframe from this point onwards

Q7. What is the relation between hour, day, week, month with number of fatal accident?

[21]:
```
#creating function to add month column to the dataset
def month(string):
    return int(string[3:5])
df['Month']=df['Date'].apply(lambda x: month(x))
```

[22]:
```
#creating function to add hour column to the dataset
def hour(string):
    s=string[0:2]
    return int(s)
df['Hour']=df['Time'].apply(lambda x: hour(x))
```

[23]:
```
#getting a dataframe suitable for Q6
q7df=pd.
 ↪DataFrame(data=df,columns=['Hour','Day_of_Week','Month','Accident_Severity'])
```

[24]:
```
q7df.head()
```

[24]:

	Hour	Day_of_Week	Month	Accident_Severity
0	17	3	1	2
1	17	4	1	3

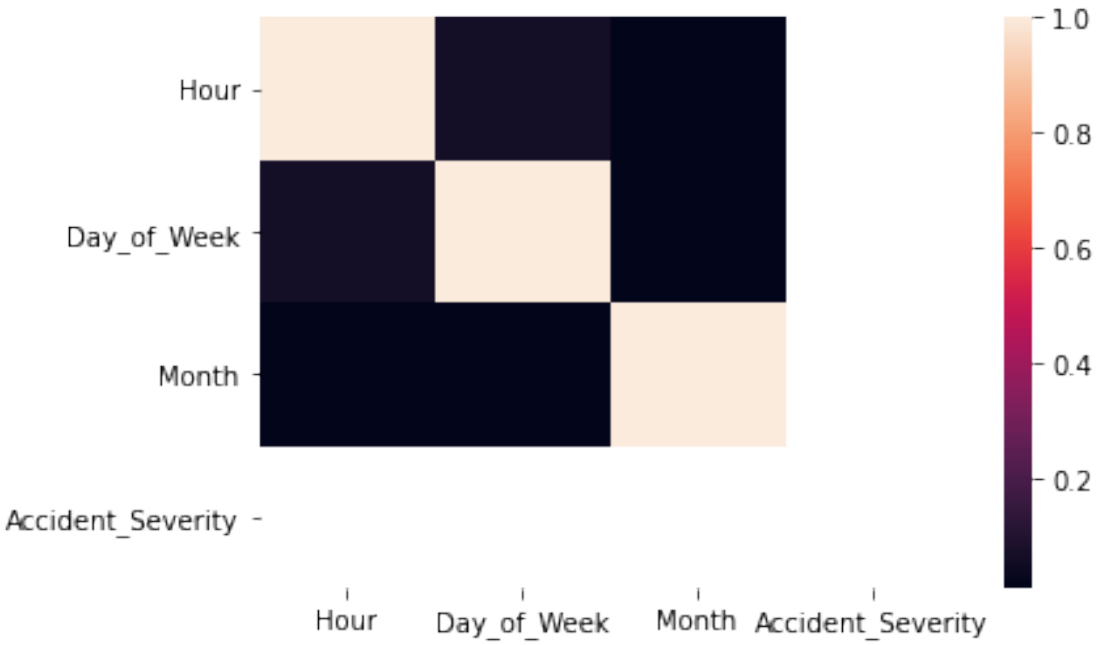
```
2     0            5       1                   3
3     0            5       1                   3
4    10            6       1                   3
```

[25]:
```
q7df=q7df[q7df.Accident_Severity ==1]
q7df.head()
```

[25]:
```
       Hour  Day_of_Week  Month  Accident_Severity
1646     18            4      5                  1
1647     18            4      5                  1
1648     18            4      5                  1
1649     18            4      5                  1
34637     9            4     11                  1
```

[26]:
```
sns.heatmap(q7df.corr())
```

[26]: <AxesSubplot:>



Q8: Does driver age has an effect on the number of accident?

[27]:
```
q8df=  pd.DataFrame(data=df, columns=['Journey_Purpose_of_Driver',␣
↪'Sex_of_Driver',␣
↪'Age_of_Driver','Age_Band_of_Driver','Driver_Home_Area_Type'])
```

[28]:
```
q8df=q8df[q8df.Sex_of_Driver !=-1]
q8df.head()
```

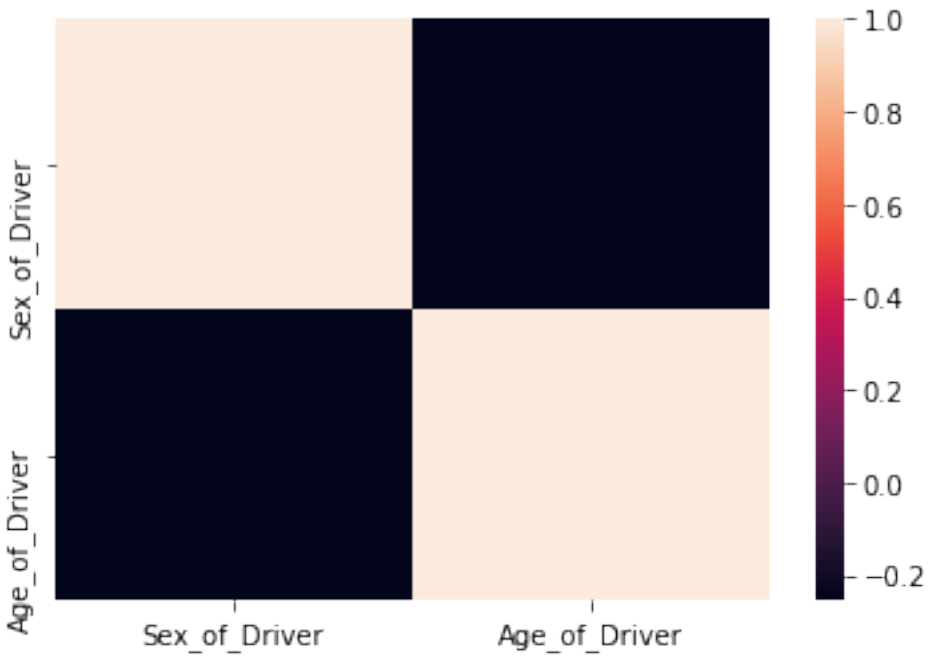
[28]:
```
   Journey_Purpose_of_Driver  Sex_of_Driver  Age_of_Driver  \
0                         15              2             74
1                          1              1             42
2                          1              1             35
3                         15              1             62
4                         15              2             49

   Age_Band_of_Driver  Driver_Home_Area_Type
0                  10                      1
1                   7                     -1
2                   6                      1
3                   9                      1
4                   8                      1
```

[29]:
```
map_df={1:'Journey as part of work',2:'Commuting to/from work',3:'Taking pupil␣
↪to/from school',4:'Pupil riding to/from school',5:'Other',6:'Not known',15:
↪'Not known/Other'}
map_df_age={1:'0 - 5',2:'6 - 10',3:'11 - 15',4:'16 - 20',5:'21 - 25',6:'26 -␣
↪35',7:'36 - 45',8:'46 - 55',9:'56 - 65',10:'66 - 75',11:'Over 75'}
map_df_area={1:'Urban Area',2:'Small Town',3:'Rural'}
q8df.Age_Band_of_Driver=q8df.Age_Band_of_Driver.map(map_df_age)
q8df.Journey_Purpose_of_Driver=q8df.Journey_Purpose_of_Driver.map(map_df)
q8df.Driver_Home_Area_Type=q8df.Driver_Home_Area_Type.map(map_df_area)
q8df.head()
```

[29]:
```
  Journey_Purpose_of_Driver  Sex_of_Driver  Age_of_Driver Age_Band_of_Driver  \
0           Not known/Other              2             74            66 - 75
1   Journey as part of work              1             42            36 - 45
2   Journey as part of work              1             35            26 - 35
3           Not known/Other              1             62            56 - 65
4           Not known/Other              2             49            46 - 55

  Driver_Home_Area_Type
0            Urban Area
1                   NaN
2            Urban Area
3            Urban Area
4            Urban Area
```

[30]:
```
sns.heatmap(q8df.corr())
```

[30]:
```
<AxesSubplot:>
```

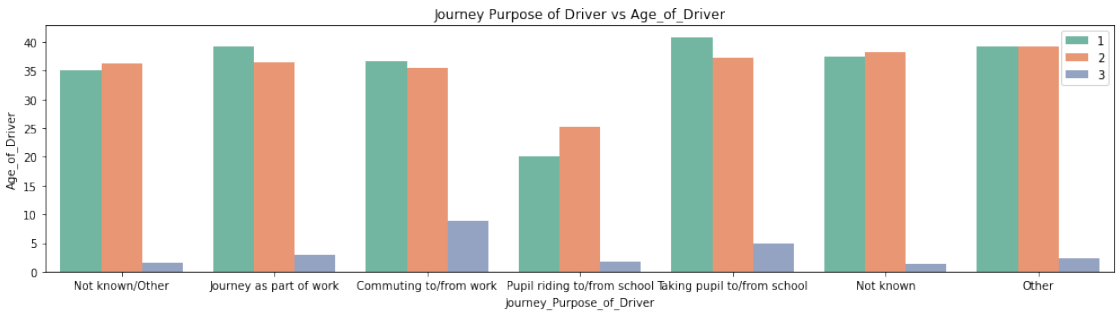


It is seen that the Drivers who met with an accident were in the age range of 30-40 years. * Usually, drivers who meet with an accident are males.

[31]:
```
plt.figure(figsize=(17,4))
sns.
 ↪barplot('Journey_Purpose_of_Driver','Age_of_Driver',hue='Sex_of_Driver',data=q8df,ci=None,␣
 ↪palette='Set2')
plt.legend(bbox_to_anchor=(1,1))
plt.title('Journey Purpose of Driver vs Age_of_Driver')
```

```
C:\Anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the
following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

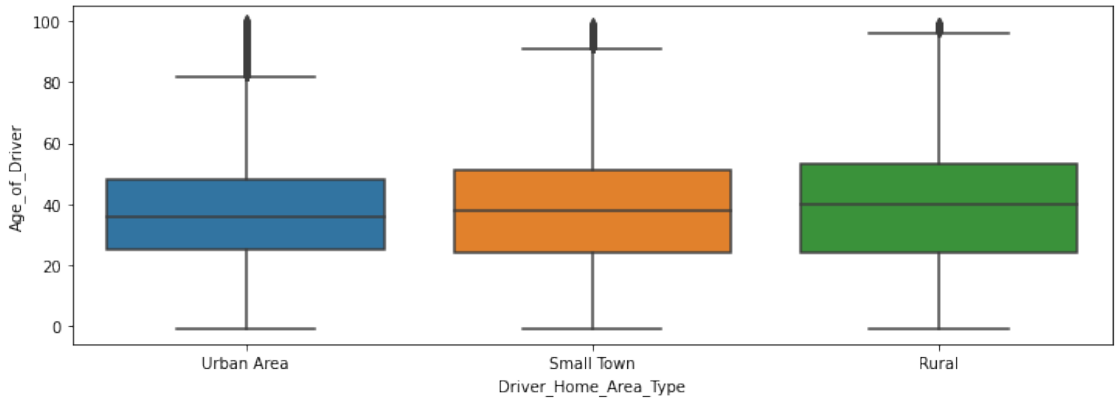
[31]:
```
Text(0.5, 1.0, 'Journey Purpose of Driver vs Age_of_Driver')
```



[32]:
```
plt.figure(figsize=(12,4))
sns.boxplot('Driver_Home_Area_Type','Age_of_Driver',data=q8df)
```

```
C:\Anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the
following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

[32]:
```
<AxesSubplot:xlabel='Driver_Home_Area_Type', ylabel='Age_of_Driver'>
```



Q9: How the weather impact the number or severity of an accident?

[33]:
```
q9df=pd.
 ↪DataFrame(data=df,columns=['Accident_Severity','Light_Conditions','Weather_Conditions','Hou
q9df.head()
```

[33]:

	Accident_Severity	Light_Conditions	Weather_Conditions	Hour
0	2	1	2	17
1	3	4	1	17
2	3	4	1	0
3	3	4	1	0
4	3	1	1	10

[34]:
```
#creating function to identify time of day: morning, afternoon, evening, night,␣
 ↪etc.
def time_of_day(n):
    if n in range(4,8):
        return 'Early Morning'
    elif n in range(8,12):
        return 'Morning'
```

```
    elif n in range(12,17):
        return 'Afternoon'
    elif n in range(17,20):
        return 'Evening'
    elif n in range(20,25) or n==0:
        return 'Night'
    elif n in range(1,4):
        return 'Late Night'
```

[35]:
```
q9df['Time_of_Day']=q9df['Hour'].apply(lambda x: time_of_day(x))
```

[36]:
```
q9df.head()
```

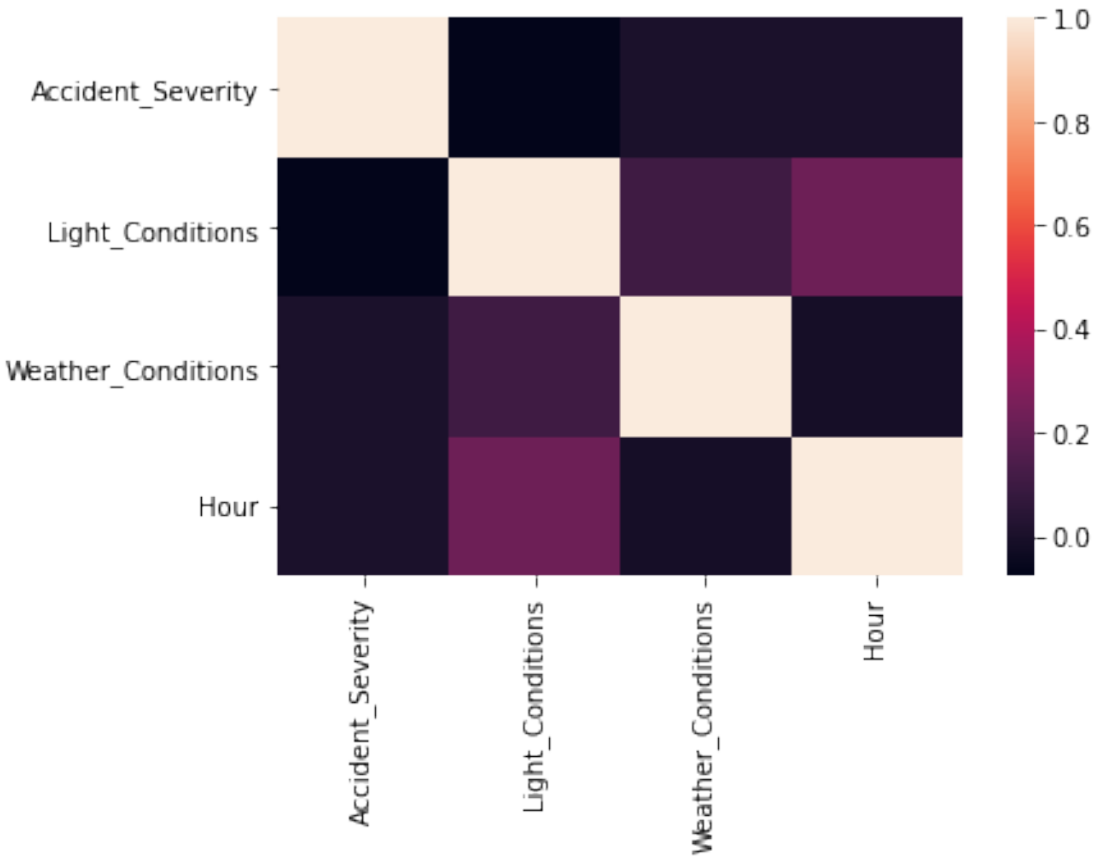
[36]:

	Accident_Severity	Light_Conditions	Weather_Conditions	Hour	Time_of_Day
0	2	1	2	17	Evening
1	3	4	1	17	Evening
2	3	4	1	0	Night
3	3	4	1	0	Night
4	3	1	1	10	Morning

[37]:
```
q9df=q9df[q9df.Weather_Conditions!=-1]
```

[38]:
```
sns.heatmap(q9df.corr())
```

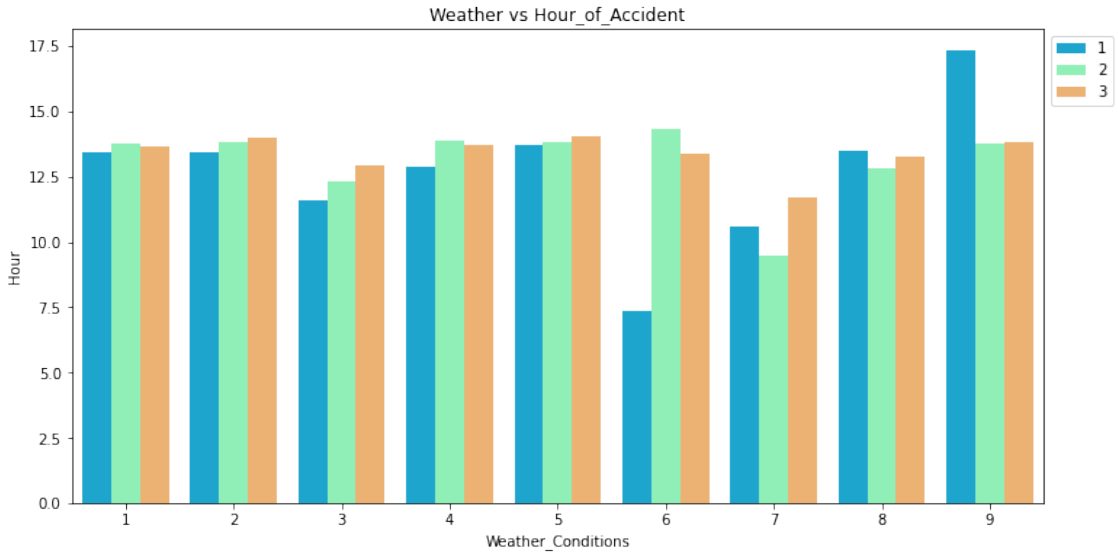
[38]:
```
<AxesSubplot:>
```



[39]:
```
plt.figure(figsize=(12,6))
sns.barplot('Weather_Conditions','Hour',data=q9df,␣
 ↪hue='Accident_Severity',ci=None, palette='rainbow')
plt.legend(bbox_to_anchor=(1,1))
plt.title('Weather vs Hour_of_Accident')
```

```
C:\Anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the
following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

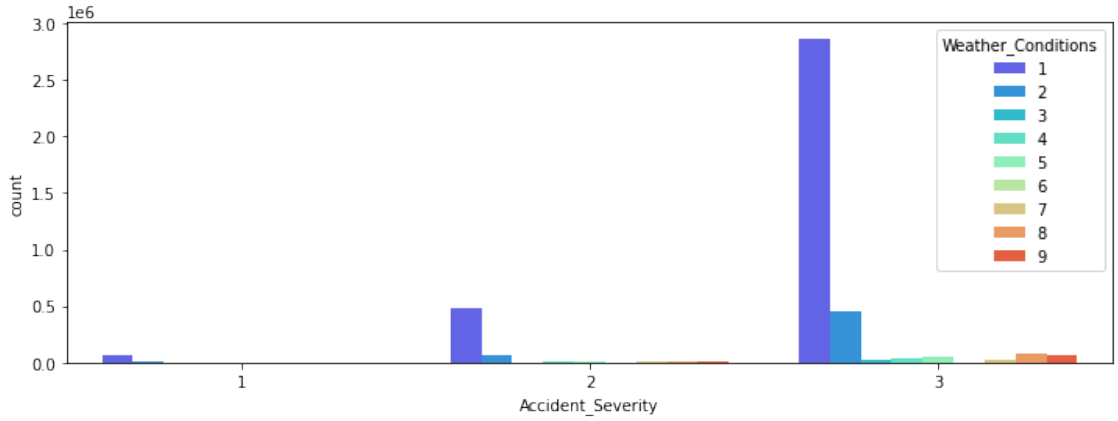
[39]:
```
Text(0.5, 1.0, 'Weather vs Hour_of_Accident')
```



- 1: Fatal
- 2: Serious
- 3: Slight

[40]:
```
plt.figure(figsize=(12,4))
sns.
 ↪countplot(x='Accident_Severity',data=q9df,hue='Weather_Conditions',palette='rainbow')
```

[40]:
```
<AxesSubplot:xlabel='Accident_Severity', ylabel='count'>
```



Weather Conditions * 1: Fine no high winds * 2: Raining no high winds * 3: Snowing no high winds * 4: Fine + high winds * 5: Raining + high winds * 6: Snowing + high winds * 7: Fog or mist * 8: Other * 9: Unknown

[41]:
```
df.Accident_Severity.value_counts()
```

[41]:
```
3    3606998
2     596470
1      83605
Name: Accident_Severity, dtype: int64
```

Results:

- Accidents usually take place in the afternoon: refer fig: Weather vs Hour_of_Accident
- Accidents with Slight severity occured the most
- **Accidents ususally took place when the Weather conditions were fine and also there weren't any high winds** : meaning which the weather conditions didn't effectively contribute to occurences of accidents.

Q10: Are certain car models safer than others?

[42]:
```
q10df=pd.
↪DataFrame(data=df,columns=['Vehicle_Type','Age_of_Vehicle','Was_Vehicle_Left_Hand_Drive?
↪'
                                    ,'Propulsion_Code','Engine_Capacity_(CC)'])
```

[43]:
```
q10df=q10df[q10df.Vehicle_Type!=-1]
q10df.head()
```

[43]:

	Vehicle_Type	Age_of_Vehicle	Was_Vehicle_Left_Hand_Drive?	Propulsion_Code	Engine_Capacity_(CC)
0	9	-1	1	-1	-1
1	11	3	1	2	8268
2	11	5	1	2	8300
3	9	6	1	1	1762
4	9	4	1	1	1769

[44]:
```
q10df=q10df[q10df.Age_of_Vehicle!=-1]
q10df=q10df[q10df.Propulsion_Code!=-1]
q10df=q10df[q10df['Engine_Capacity_(CC)']!=-1]
```

[45]:
```
map_vehicle_type={1:'Pedal cycle',
2:'Motorcycle 50cc and under',
3:'Motorcycle 125cc and under',
4:'Motorcycle over 125cc and up to 500cc',
5:'Motorcycle over 500cc',
8:'Taxi/Private hire car',
```

```
9:'Car',
10:'Minibus (8 - 16 passenger seats)',
11:'Bus or coach (17 or more pass seats)',
16:'Ridden horse',
17:'Agricultural vehicle',
18:'Tram',
19:'Van / Goods 3.5 tonnes mgw or under',
20:'Goods over 3.5t. and under 7.5t',
21:'Goods 7.5 tonnes mgw and over',
22:'Mobility scooter',
23:'Electric motorcycle',
90:'Other vehicle',
97:'Motorcycle - unknown cc',
98:'Goods vehicle - unknown weight'
}
q10df['Vehicle_Type']=q10df.Vehicle_Type.map(map_vehicle_type)
```

[46]:
```
map_prop={1:'Petrol',
2:'Heavy oil',
3:'Electric',
4:'Steam',
5:'Gas',
6:'Petrol/Gas (LPG)',
7:'Gas/Bi-fuel',
8:'Hybrid electric',
9:'Gas Diesel',
10:'New fuel technology',
11:'Fuel cells',
12:'Electric diesel'
}
q10df['Propulsion_Code']=q10df.Propulsion_Code.map(map_prop)
```

[47]:
```
q10df=q10df[q10df['Was_Vehicle_Left_Hand_Drive?']!=-1]
q10df.head()
```

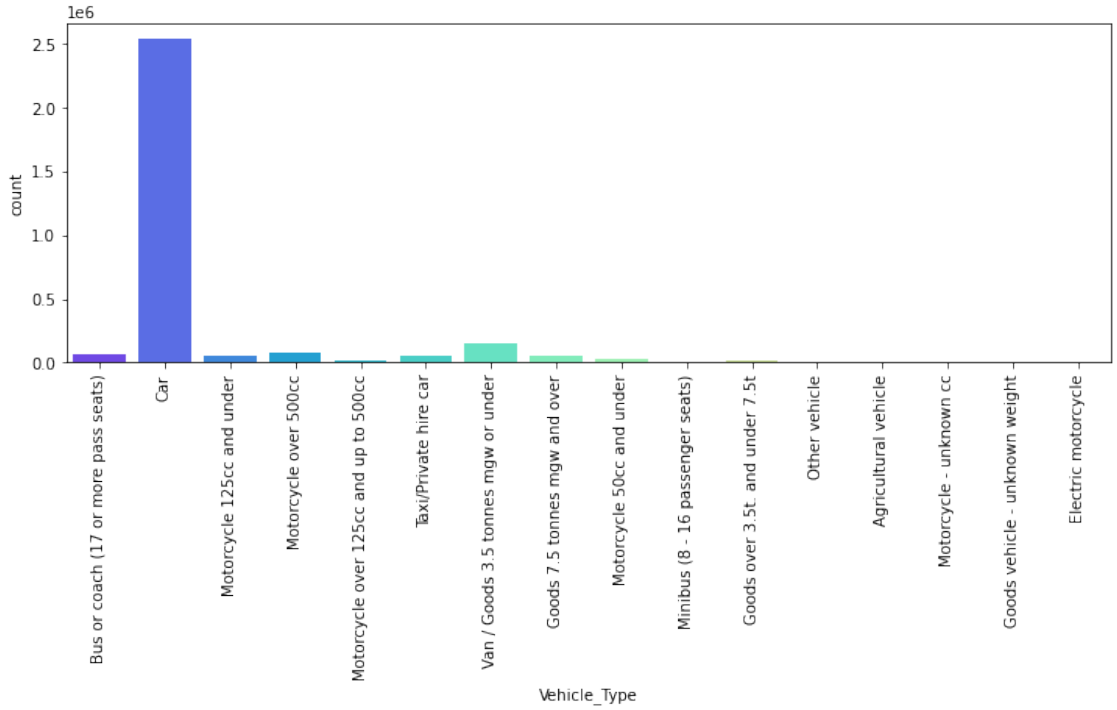
[47]:
```
                          Vehicle_Type  Age_of_Vehicle  \
1  Bus or coach (17 or more pass seats)               3
2  Bus or coach (17 or more pass seats)               5
3                                   Car               6
4                                   Car               4
5           Motorcycle 125cc and under              10

   Was_Vehicle_Left_Hand_Drive? Propulsion_Code  Engine_Capacity_(CC)
1                             1       Heavy oil                  8268
2                             1       Heavy oil                  8300
3                             1          Petrol                  1762
4                             1          Petrol                  1769
```

```
5                              1          Petrol                    85
```

[48]:
```
plt.figure(figsize=(12,4))
sns.countplot('Vehicle_Type',data=q10df, palette='rainbow')
plt.xticks(rotation=90)
```

```
C:\Anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the
following variable as a keyword arg: x. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

[48]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15]),
 [Text(0, 0, 'Bus or coach (17 or more pass seats)'),
  Text(1, 0, 'Car'),
  Text(2, 0, 'Motorcycle 125cc and under'),
  Text(3, 0, 'Motorcycle over 500cc'),
  Text(4, 0, 'Motorcycle over 125cc and up to 500cc'),
  Text(5, 0, 'Taxi/Private hire car'),
  Text(6, 0, 'Van / Goods 3.5 tonnes mgw or under'),
  Text(7, 0, 'Goods 7.5 tonnes mgw and over'),
  Text(8, 0, 'Motorcycle 50cc and under'),
  Text(9, 0, 'Minibus (8 - 16 passenger seats)'),
  Text(10, 0, 'Goods over 3.5t. and under 7.5t'),
  Text(11, 0, 'Other vehicle'),
  Text(12, 0, 'Agricultural vehicle'),
  Text(13, 0, 'Motorcycle - unknown cc'),
  Text(14, 0, 'Goods vehicle - unknown weight'),
  Text(15, 0, 'Electric motorcycle')])
```



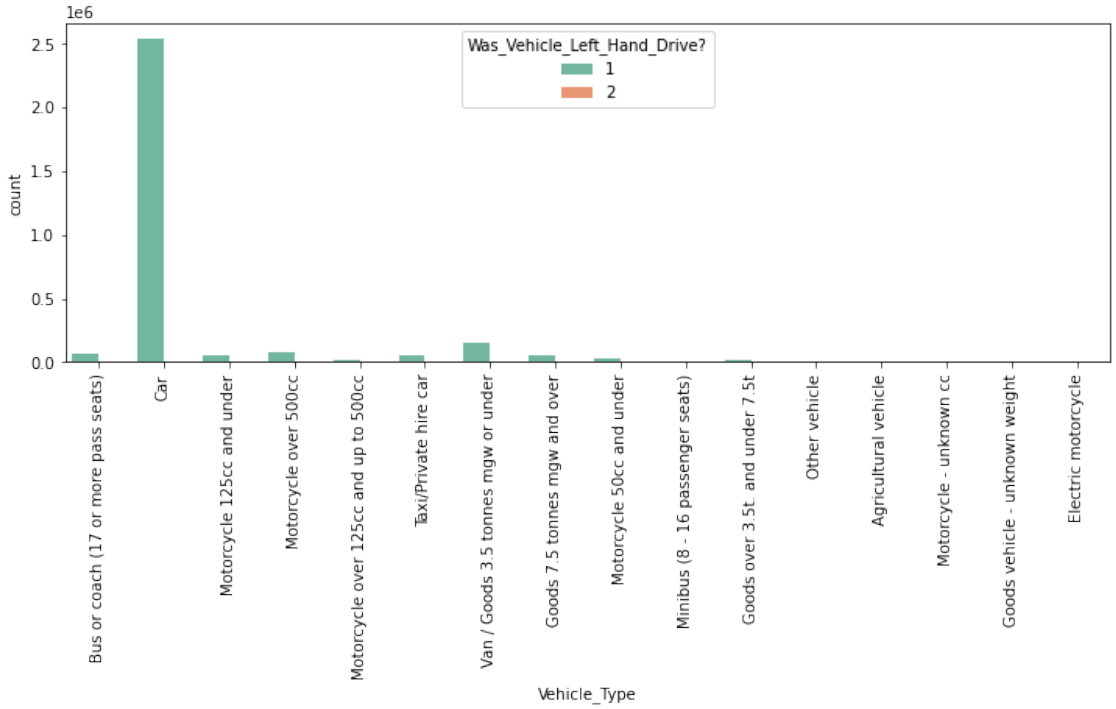
Number of accidents taking place with other vehicles are almost negligible as comapred to those with Cars.

[49]:
```
plt.figure(figsize=(12,4))
sns.countplot('Vehicle_Type',data=q10df, hue='Was_Vehicle_Left_Hand_Drive?',␣
 ↪palette='Set2')
plt.xticks(rotation=90)
```

```
C:\Anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the
following variable as a keyword arg: x. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

[49]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15]),
 [Text(0, 0, 'Bus or coach (17 or more pass seats)'),
  Text(1, 0, 'Car'),
  Text(2, 0, 'Motorcycle 125cc and under'),
  Text(3, 0, 'Motorcycle over 500cc'),
  Text(4, 0, 'Motorcycle over 125cc and up to 500cc'),
  Text(5, 0, 'Taxi/Private hire car'),
  Text(6, 0, 'Van / Goods 3.5 tonnes mgw or under'),
  Text(7, 0, 'Goods 7.5 tonnes mgw and over'),
  Text(8, 0, 'Motorcycle 50cc and under'),
  Text(9, 0, 'Minibus (8 - 16 passenger seats)'),
```

```
Text(10, 0, 'Goods over 3.5t. and under 7.5t'),
Text(11, 0, 'Other vehicle'),
Text(12, 0, 'Agricultural vehicle'),
Text(13, 0, 'Motorcycle - unknown cc'),
Text(14, 0, 'Goods vehicle - unknown weight'),
Text(15, 0, 'Electric motorcycle')])
```

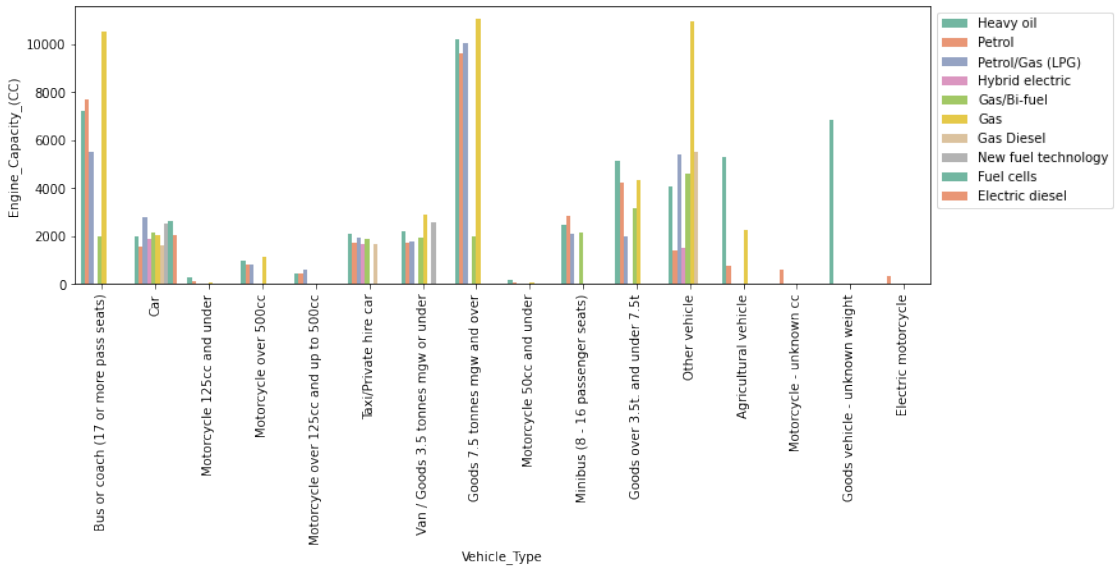


The vehicles which met with an accident were primarily Left-Hand-Driven type of vehicles.

[50]:
```
plt.figure(figsize=(12,4))
sns.barplot('Vehicle_Type','Engine_Capacity_(CC)',data=q10df,␣
 ↪hue='Propulsion_Code', palette='Set2',ci=None)
plt.xticks(rotation=90)
plt.legend(bbox_to_anchor=(1,1))
```

```
C:\Anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the
following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

[50]:
```
<matplotlib.legend.Legend at 0x1e64675f1f0>
```



Cars had low Engine Capacity with all types of fuel, which could be a possible reason for accidents.

1.3 Forecasting and predictions

As per the final objective of this project, we want to forecast which attribute of fatal accidents we can predict using machine learning techniques.

[51]:
```
fatal_df=pd.
 ↪DataFrame(data=df,columns=['Sex_of_Driver','Age_of_Driver','Vehicle_Type','Month','Accident
fatal_df=fatal_df[(fatal_df.Sex_of_Driver!=-1) & (fatal_df.Vehicle_Type!=-1) &␣
 ↪(fatal_df.Sex_of_Driver!=-1) & (fatal_df.Sex_of_Driver!=3)]
fatal_df.head()
```

[51]:

	Sex_of_Driver	Age_of_Driver	Vehicle_Type	Month	Accident_Severity
0	2	74	9	1	2
1	1	42	11	1	3
2	1	35	11	1	3
3	1	62	9	1	3
4	2	49	9	1	3

[52]:
```
acc=pd.get_dummies(data=fatal_df,columns=['Accident_Severity'])
sex=pd.get_dummies(data=fatal_df,columns=['Sex_of_Driver'])
sex.head()
```

[52]:

	Age_of_Driver	Vehicle_Type	Month	Accident_Severity	Sex_of_Driver_1	\
0	74	9	1	2	0	
1	42	11	1	3	1	
2	35	11	1	3	1	

```
3              62              9      1                    3                  1
4              49              9      1                    3                  0

   Sex_of_Driver_2
0                1
1                0
2                0
3                0
4                1
```

Forecasting first considering the male gender

[53]:
```
fatal_df_male=pd.
 ↪concat([fatal_df,acc['Accident_Severity_1'],sex['Sex_of_Driver_1']],axis=1)
fatal_df_male.head()
```

[53]:
```
   Sex_of_Driver  Age_of_Driver  Vehicle_Type  Month  Accident_Severity  \
0              2             74             9      1                  2
1              1             42            11      1                  3
2              1             35            11      1                  3
3              1             62             9      1                  3
4              2             49             9      1                  3

   Accident_Severity_1  Sex_of_Driver_1
0                    0                0
1                    0                1
2                    0                1
3                    0                1
4                    0                0
```

[54]:
```
fatal_df_male.drop(['Accident_Severity','Sex_of_Driver'],axis=1,inplace=True)
fatal_df_male.head()
```

[54]:
```
   Age_of_Driver  Vehicle_Type  Month  Accident_Severity_1  Sex_of_Driver_1
0             74             9      1                    0                0
1             42            11      1                    0                1
2             35            11      1                    0                1
3             62             9      1                    0                1
4             49             9      1                    0                0
```

Note: Accident_Severity_1 corresponds to fatal accident and Sex_of_Driver_1 corresponds to male driver

[55]:
```
X=fatal_df_male.drop('Accident_Severity_1',axis=1)
y=fatal_df_male['Accident_Severity_1']
```

[56]:
```
X_train, X_test, y_train, y_test= train_test_split(X,y)
```

Using Decision Tree

[57]:
```
dtree= DecisionTreeClassifier()
dtree.fit(X_train,y_train)
predictions= dtree.predict(X_test)
print("\nClassification Report--> \n\n")
print(classification_report(y_test,predictions))
print("\nConfusion Matrix--> \n\n")
print(confusion_matrix(y_test,predictions))
```

```
Classification Report-->


              precision    recall  f1-score   support

           0       0.98      1.00      0.99    998270
           1       0.46      0.00      0.01     20282

    accuracy                           0.98   1018552
   macro avg       0.72      0.50      0.50   1018552
weighted avg       0.97      0.98      0.97   1018552


Confusion Matrix-->


[[998180     90]
 [ 20205     77]]
```

Results:

It seems like the model didn't do well * Though the precision is good, it is noticed that the model had better predictions for only case: 0 * Also, checking the recall, it is noticed that case: 1 is neglected.

[65]:
```
#Decision Tree Visualization
clf = DecisionTreeClassifier(criterion = "entropy", max_depth=10)
clf.fit(X_train,y_train)
dot_data = StringIO()
export_graphviz(clf, out_file=dot_data, filled=True, rounded=True,␣
 ↪special_characters=True,feature_names = X.columns.
 ↪tolist(),class_names=['0','1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('data.png')
#Image(graph.create_png())
```

```
dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.289459 to fit
```

[65]:
```
True
```

Forecasting second considering the female gender

[59]:
```
fatal_df_female=pd.
 ↪concat([fatal_df,acc['Accident_Severity_1'],sex['Sex_of_Driver_2']],axis=1)
fatal_df_female.head()
```

[59]:

	Sex_of_Driver	Age_of_Driver	Vehicle_Type	Month	Accident_Severity	\
0	2	74	9	1	2	
1	1	42	11	1	3	
2	1	35	11	1	3	
3	1	62	9	1	3	
4	2	49	9	1	3	

	Accident_Severity_1	Sex_of_Driver_2
0	0	1
1	0	0
2	0	0
3	0	0
4	0	1

[60]:
```
fatal_df_female.drop(['Accident_Severity','Sex_of_Driver'],axis=1,inplace=True)
fatal_df_female.head()
```

[60]:

	Age_of_Driver	Vehicle_Type	Month	Accident_Severity_1	Sex_of_Driver_2
0	74	9	1	0	1
1	42	11	1	0	0
2	35	11	1	0	0
3	62	9	1	0	0
4	49	9	1	0	1

Note: Accident_Severity_1 corresponds to fatal accident and Sex_of_Driver_2 corresponds to female drivers

[61]:
```
XX=fatal_df_female.drop('Accident_Severity_1',axis=1)
yy=fatal_df_female['Accident_Severity_1']
```

[62]:
```
XX_train, XX_test, yy_train, yy_test= train_test_split(XX,yy)
```

Using Decision Tree

[63]:
```
dtree2= DecisionTreeClassifier()
dtree2.fit(XX_train,yy_train)
predictions2= dtree.predict(XX_test)
print("\nClassification Report--> \n\n")
print(classification_report(yy_test,predictions2))
print("\nConfusion Matrix--> \n\n")
print(confusion_matrix(yy_test,predictions2))
```

```
Classification Report-->


              precision    recall  f1-score   support

           0       0.98      1.00      0.99    998162
           1       0.06      0.00      0.01     20390

    accuracy                           0.98   1018552
   macro avg       0.52      0.50      0.50   1018552
weighted avg       0.96      0.98      0.97   1018552


Confusion Matrix-->


[[996851   1311]
 [ 20306     84]]
```

Results:

It seems like the model didn't do well * Though the precision is good, it is noticed that the model had better predictions for only case: 0 * Also, checking the recall, it is noticed that case: 1 is neglected.

[66]:
```
#Decision Tree Visualization
clf2 = DecisionTreeClassifier(criterion = "entropy", max_depth=10)
clf2.fit(XX_train,yy_train)
dot_data2 = StringIO()
export_graphviz(clf2, out_file=dot_data2, filled=True, rounded=True,␣
↪special_characters=True,feature_names = X.columns.
↪tolist(),class_names=['0','1'])
graph2 = pydotplus.graph_from_dot_data(dot_data2.getvalue())
graph2.write_png('data2.png')
#Image(graph2.create_png())
```

```
dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.295846 to fit
```

[66]:
```
True
```

1.5 Conclusion

From our **Analysis of Road Traffic Casualties** project, we can draw out a couple of conclusions:

1. Accident Region Fraction:
    - Percentage of accidents occur in urban areas is 64%
    - Percentage of accidents occur in rural areas is 36%
    - Percentage of accidents occur in other areas is 0%
2. The most dangerous hour to drive, when most fatal accidents happened in all accidents, is 4 PM
3. The trend of road accidents has been decreasing consistently from 2004 to 2013, but began increasing again in 2014.
4. Fatality Fraction:
    - Percentage of Deaths is 1%
    - Percentage of Major Injuries is 11%
    - Percentage of Minor Injuries is 88%
5. Most accidents involve new drivers (18/19-year-olds).
6. The age class for the highest accident rate is 20-30.
7. Women are involved in only about ⅔ of the number of accidents that men are involved in.
8. Accident Severity based on weather conditions:

9. Number of accidents taking place with other vehicles are almost negligible as compared to those with Cars.

10. The vehicles which met with an accident were primarily Left-Hand-Driven type of vehicles.

11. Cars had low Engine Capacity with all types of fuel, which could be a possible reason for accidents.

Using our conclusions, we can design some **Prevention Methods** that could possibly mitigate the number of road accidents occurring each year:

1. Since 4 PM is the most fatal hour for accidents, authorities could incorporate certain speed limits from 3PM to 6PM.
2. Young drivers, around the age of 20, seem to be involved in the highest number of road accidents. Curfews could be enforced on at certain peak accident hours.

3. Using our predictive forecasting model, accident severity conditions can be analyzed and counter-measures can be implemented.

Decision Tree (Target attribute - Accident severity)

1. For male gender class - based on feature attributes of the above-mentioned dataset

2. For female gender class - based on feature attributes of the above-mentioned dataset

Appendix - List of figures