TITLE - BIG DATA COURSEWORK - ROAD SAFETY DATASET

Table of Content

1. Introduction: Business Objective and Problem Context
2. Data Source
3. Data Exploration and Validation
4. Data Preperation
5. Exploratory Data Analysis
6. Missing value and Outlier Treatment
7. Dummy Variable creation
8. Exporting Data
9. Conclusion

1.INTRODUCTION : BUSINESS OBJECTIVE & PROBLEM CONTEXT

There is no doubt that the top cause of winter car accidents is ice and snow on the roadways. When the roads are icy and slick, the traction on your tires is less effective. Therefore impacting a huge loss for the Insurance companies.

The more the accidents the higher the claims raised by the insurer, therefore insurance companies are in a stage to introduce new policies from keeping their revenue and profit intact.

Therefore we aim to predict the severity of accident within the United Kingdom during the snow season and suggest "Forever Live" Insurance Company with preplanned policies that take winter prone accidents into consideration.

We are going to use the Machine Learning Methods to solve this classification problem keeping Accident Severity as our Target variable.

In [3]:

2.DATA SOURCE

We have the ROAD SAFETY statistical data provided by the government of UK in the below mentioned website. We will be using the links from that to access our Dataset. For our Prediction we will be using 3 different datasets namely Accidents, Casualty & Vehicles from the year 2021 and merge them into one file.

We will be selecting 24 features from the total of 81 columns and predict our final output.

3.DATA EXTRACTION AND VALIDATION

3.1 IMPORTING LIBRARIES

In [ ]:
```
import numpy as np
import pandas as pd
import re

import gdown
```

In [ ]:
```
import datetime
from datetime import datetime as dt
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
import seaborn as sns
import matplotlib.pyplot as plt
```

3.2 LOADING DATASET

In [ ]:
```
#Importing data
#Accidents
url='https://drive.google.com/uc?id=1vRYoBJ09S7k0n6gqJqJVVRzni7zlkx-x'
output='Accidents.csv'
gdown.download(url, output, quiet=False)
accidents=pd.read_csv(output)
#https://drive.google.com/file/d/1vRYoBJ09S7k0n6gqJqJVVRzni7zlkx-x/view?usp=sharing
#casuality
url1='https://drive.google.com/uc?id=1mKitbeHHU_nVoFlecM-38xcdAsHkNvQ4'
output1='Casuality.csv'
gdown.download(url1, output1, quiet=False)
casuality=pd.read_csv(output1)
#Vehicles
url2='https://drive.google.com/uc?id=1nCGcrEVPZUtxr3t-CXJHe5jCKeaRNPRU'
output2='Vehicles.csv'
gdown.download(url2, output2, quiet=False)
vehicles=pd.read_csv(output2)
```

```
Downloading...
From: https://drive.google.com/uc?id=1vRYoBJ09S7k0n6gqJqJVVRzni7zlkx-x
To: /content/Accidents.csv
100%|██████████| 16.5M/16.5M [00:00<00:00, 65.0MB/s]
Downloading...
From: https://drive.google.com/uc?id=1mKitbeHHU_nVoFlecM-38xcdAsHkNvQ4
To: /content/Casuality.csv
100%|██████████| 9.04M/9.04M [00:00<00:00, 56.8MB/s]
Downloading...
From: https://drive.google.com/uc?id=1nCGcrEVPZUtxr3t-CXJHe5jCKeaRNPRU
To: /content/Vehicles.csv
100%|██████████| 18.7M/18.7M [00:00<00:00, 53.3MB/s]
```

Veryfing if all three datasets are uploaded or not using the Head function.

In [ ]:
```
accidents.head(3)
#validating the accidents dataframe
```

Out[ ]:

	accident_index	accident_year	accident_reference	location_easting_osgr	location_northing_osgr	longitude	latitude	police_
0	2021010287148	2021	10287148	521508.0	193079.0	-0.246102	51.623425	
1	2021010287149	2021	10287149	535379.0	180783.0	-0.050574	51.509767	
2	2021010287151	2021	10287151	529701.0	170398.0	-0.136152	51.417769	

3 rows × 36 columns

In [ ]:

In [ ]:
```
vehicles.head()
# validating the vehicles dataframe
```

Out[ ]:

	accident_index	accident_year	accident_reference	vehicle_reference	vehicle_type	towing_and_articulation	vehicle_manoeuvre
0	2021010287148	2021	10287148	1	9	0	17
1	2021010287148	2021	10287148	2	9	0	2
2	2021010287148	2021	10287148	3	9	0	2
3	2021010287149	2021	10287149	1	9	0	18
4	2021010287149	2021	10287149	2	9	0	18

5 rows × 28 columns

In [ ]:
```
casuality.head()
#validating the cauality dataframe.
```

Out[ ]:

	accident_index	accident_year	accident_reference	vehicle_reference	casualty_reference	casualty_class	sex_of_casualty	age_of
0	2021010287148	2021	10287148	1	1	1	1	
1	2021010287149	2021	10287149	1	1	2	1	
2	2021010287149	2021	10287149	2	2	1	1	
3	2021010287149	2021	10287149	2	3	2	1	
4	2021010287151	2021	10287151	1	1	1	1	

3.3 MERGING DATASET

In [ ]:
```
merged_df1 = accidents.merge(vehicles,on='accident_index', how="inner").merge(casuality,on='accident_index',
merged_df1.shape
```

Out[ ]:
```
(155457, 81)
```

Verfying our merged data through shape we can find we have 155457 rows and 81 columns

In [ ]:
```
merged_df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 155457 entries, 0 to 155456
Data columns (total 81 columns):
 #   Column                                       Non-Null Count   Dtype  
---  ------                                       --------------   -----  
 0   accident_index                               155457 non-null  object 
 1   accident_year_x                              155457 non-null  int64  
 2   accident_reference_x                         155457 non-null  object 
 3   location_easting_osgr                        155418 non-null  float64
 4   location_northing_osgr                       155418 non-null  float64
 5   longitude                                    155418 non-null  float64
 6   latitude                                     155418 non-null  float64
 7   police_force                                 155457 non-null  int64  
 8   accident_severity                            155457 non-null  int64  
 9   number_of_vehicles                           155457 non-null  int64  
 10  number_of_casualties                         155457 non-null  int64  
 11  date                                         155457 non-null  object 
 12  day_of_week                                  155457 non-null  int64  
 13  time                                         155457 non-null  object 
 14  local_authority_district                     155457 non-null  int64  
 15  local_authority_ons_district                 155457 non-null  object 
 16  local_authority_highway                      155457 non-null  object 
 17  first_road_class                             155457 non-null  int64  
 18  first_road_number                            155457 non-null  int64  
 19  road_type                                    155457 non-null  int64  
 20  speed_limit                                  155457 non-null  int64  
 21  junction_detail                              155457 non-null  int64  
 22  junction_control                             155457 non-null  int64  
 23  second_road_class                            155457 non-null  int64  
 24  second_road_number                           155457 non-null  int64  
 25  pedestrian_crossing_human_control            155457 non-null  int64  
 26  pedestrian_crossing_physical_facilities      155457 non-null  int64  
 27  light_conditions                             155457 non-null  int64  
 28  weather_conditions                           155457 non-null  int64  
 29  road_surface_conditions                      155457 non-null  int64  
 30  special_conditions_at_site                   155457 non-null  int64  
 31  carriageway_hazards                          155457 non-null  int64  
 32  urban_or_rural_area                          155457 non-null  int64  
 33  did_police_officer_attend_scene_of_accident  155457 non-null  int64  
 34  trunk_road_flag                              155457 non-null  int64  
 35  lsoa_of_accident_location                    155457 non-null  object 
 36  accident_year_y                              155457 non-null  int64  
 37  accident_reference_y                         155457 non-null  object 
 38  vehicle_reference_x                          155457 non-null  int64  
 39  vehicle_type                                 155457 non-null  int64  
 40  towing_and_articulation                      155457 non-null  int64  
 41  vehicle_manoeuvre                            155457 non-null  int64  
 42  vehicle_direction_from                       155457 non-null  int64  
 43  vehicle_direction_to                         155457 non-null  int64  
 44  vehicle_location_restricted_lane             155457 non-null  int64  
 45  junction_location                            155457 non-null  int64  
 46  skidding_and_overturning                     155457 non-null  int64  
 47  hit_object_in_carriageway                    155457 non-null  int64  
 48  vehicle_leaving_carriageway                  155457 non-null  int64  
 49  hit_object_off_carriageway                   155457 non-null  int64  
 50  first_point_of_impact                        155457 non-null  int64  
 51  vehicle_left_hand_drive                      155457 non-null  int64  
 52  journey_purpose_of_driver                    155457 non-null  int64  
 53  sex_of_driver                                155457 non-null  int64  
 54  age_of_driver                                155457 non-null  int64  
 55  age_band_of_driver                           155457 non-null  int64  
 56  engine_capacity_cc                           155457 non-null  int64  
 57  propulsion_code                              155457 non-null  int64  
 58  age_of_vehicle                               155457 non-null  int64  
 59  generic_make_model                           155457 non-null  object 
 60  driver_imd_decile                            155457 non-null  int64  
 61  driver_home_area_type                        155457 non-null  int64  
 62  lsoa_of_driver                               155457 non-null  object 
 63  accident_year                                155457 non-null  int64  
 64  accident_reference                           155457 non-null  object 
 65  vehicle_reference_y                          155457 non-null  int64  
 66  casualty_reference                           155457 non-null  int64  
 67  casualty_class                               155457 non-null  int64  
 68  sex_of_casualty                              155457 non-null  int64  
 69  age_of_casualty                              155457 non-null  int64  
 70  age_band_of_casualty                         155457 non-null  int64  
 71  casualty_severity                            155457 non-null  int64  
 72  pedestrian_location                          155457 non-null  int64  
 73  pedestrian_movement                          155457 non-null  int64  
 74  car_passenger                                155457 non-null  int64  
```

```
 75  bus_or_coach_passenger                    155457 non-null  int64
 76  pedestrian_road_maintenance_worker        155457 non-null  int64
 77  casualty_type                             155457 non-null  int64
 78  casualty_home_area_type                   155457 non-null  int64
 79  casualty_imd_decile                       155457 non-null  int64
 80  lsoa_of_casualty                          155457 non-null  object
dtypes: float64(4), int64(65), object(12)
memory usage: 97.3+ MB
```

Converting our date column to datetime varible

In [ ]:
```
merged_df1['date'] = pd.to_datetime(merged_df1['date'])
```

In [ ]:
```
merged_df1['date'].head()
```

Out[ ]:
```
0   2021-01-01
1   2021-01-01
2   2021-01-01
3   2021-01-01
4   2021-01-01
Name: date, dtype: datetime64[ns]
```

Accidents are more in winter in UK due high snow fall and hence we are focusing on data of winter

In [ ]:
```
winter_data = merged_df1[((merged_df1['date'].dt.month == 1) | (merged_df1['date'].dt.month == 12))]
```

3.4 SHORTLISTING OUR FEATURES

In [ ]:
```
Selected_columns=['accident_index','accident_severity','number_of_vehicles','number_of_casualties','date','t
```

Selected Features are being updated to the new dataframe

In [ ]:
```
final_data=winter_data[Selected_columns]
```

In [ ]:
```
final_data.isnull().sum()
```

Out[ ]:
```
accident_index             0
accident_severity          0
number_of_vehicles         0
number_of_casualties       0
date                       0
time                       0
road_type                  0
light_conditions           0
weather_conditions         0
road_surface_conditions    0
sex_of_driver              0
age_of_driver              0
engine_capacity_cc         0
age_of_vehicle             0
casualty_severity          0
dtype: int64
```

4.DATA PREPARATION

Checking the missing values in each row

In [ ]:
```
final_data.isnull().sum()
```

Out[ ]:
```
accident_index             0
accident_severity          0
number_of_vehicles         0
number_of_casualties       0
date                       0
time                       0
road_type                  0
light_conditions           0
weather_conditions         0
road_surface_conditions    0
sex_of_driver              0
age_of_driver              0
engine_capacity_cc         0
age_of_vehicle             0
casualty_severity          0
dtype: int64
```

In [ ]:
```
final_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21750 entries, 0 to 155254
Data columns (total 15 columns):
 #   Column                   Non-Null Count  Dtype         
---  ------                   --------------  -----         
 0   accident_index           21750 non-null  object        
 1   accident_severity        21750 non-null  int64         
 2   number_of_vehicles       21750 non-null  int64         
 3   number_of_casualties     21750 non-null  int64         
 4   date                     21750 non-null  datetime64[ns]
 5   time                     21750 non-null  object        
 6   road_type                21750 non-null  int64         
 7   light_conditions         21750 non-null  int64         
 8   weather_conditions       21750 non-null  int64         
 9   road_surface_conditions  21750 non-null  int64         
 10  sex_of_driver            21750 non-null  int64         
 11  age_of_driver            21750 non-null  int64         
 12  engine_capacity_cc       21750 non-null  int64         
 13  age_of_vehicle           21750 non-null  int64         
 14  casualty_severity        21750 non-null  int64         
dtypes: datetime64[ns](1), int64(12), object(2)
memory usage: 2.7+ MB
```

Some categorical variable are stored as numerical variable in the dataframe and we need to change the datatype of these variables

In [ ]:
```
final_data['accident_severity']=final_data['accident_severity'].astype(str)
final_data['road_type']=final_data['road_type'].astype(str)
final_data['light_conditions']=final_data['light_conditions'].astype(str)
final_data['weather_conditions']=final_data['weather_conditions'].astype(str)
final_data['road_surface_conditions']=final_data['road_surface_conditions'].astype(str)
final_data['sex_of_driver']=final_data['sex_of_driver'].astype(str)
final_data['casualty_severity']=final_data['casualty_severity'].astype(str)
```

4.2 REMOVING DUPLICATES

It shows that there are 4493 duplicate values in our data

In [ ]:
```
final_data.duplicated().sum()
```

Out[ ]:
```
4493
```

There are 4493 duplicates in this dataset.

Dropping the duplicates from the final dataframe Because duplicate rows (identical rows) will introduce bias when analysing the data, they must be deleted.

In [ ]:
```
df_final=final_data.drop_duplicates(inplace=False)
```

4.4 DATA PARTITIONING

Performing Train Test Split

In [ ]:
```
training_set ,testing_set = train_test_split(df_final,test_size=0.2,random_state=7)
```

In [ ]:
```
print(f' There are {training_set.shape[0]} rows in the training set ')
print(f' There are {testing_set.shape[0]} rows in the test set ')
print(f' There are {training_set.shape[1]} columns in the training set ')
print(f' There are {testing_set.shape[1]} columns in the test set ')
```

```
 There are 13805 rows in the training set
 There are 3452 rows in the test set
 There are 15 columns in the training set
 There are 15 columns in the test set
```

5.EXPLORATORY DATA ANALYSIS

5.1 DESCRIPTIVE STATISTICS

Splitting the category variable and the numerical variable from the dataset.

In [ ]:
```
cat_variable = training_set.select_dtypes(include='object')
num_variable = training_set.select_dtypes(include=[float,int])
```

Checking the descriptive statistics for numerical variables

In [ ]:
```
num_variable.describe()
```

Out[ ]:

	number_of_vehicles	number_of_casualties	age_of_driver	engine_capacity_cc	age_of_vehicle
count	13805.000000	13805.000000	13805.000000	13805.000000	13805.00000
mean	2.092865	1.377544	34.686708	1198.424411	5.27845
std	0.848702	0.774028	20.840840	1525.786869	6.34178
min	1.000000	1.000000	-1.000000	-1.000000	-1.00000
25%	2.000000	1.000000	22.000000	-1.000000	-1.00000
50%	2.000000	1.000000	34.000000	1242.000000	4.00000
75%	2.000000	2.000000	49.000000	1797.000000	10.00000
max	10.000000	8.000000	99.000000	29980.000000	48.00000

Checking the descriptive statistics for categorical variables

In [ ]:
```
cat_variable.describe()
```

Out[ ]:

	accident_index	accident_severity	time	road_type	light_conditions	weather_conditions	road_surface_conditions	sex_o
count	13805	13805	13805	13805	13805	13805	13805	
unique	8288	3	1245	6	6	10	7	
top	2021361098508	3	17:30	6	1	1	1	
freq	15	10430	142	9860	8431	10119	7891	

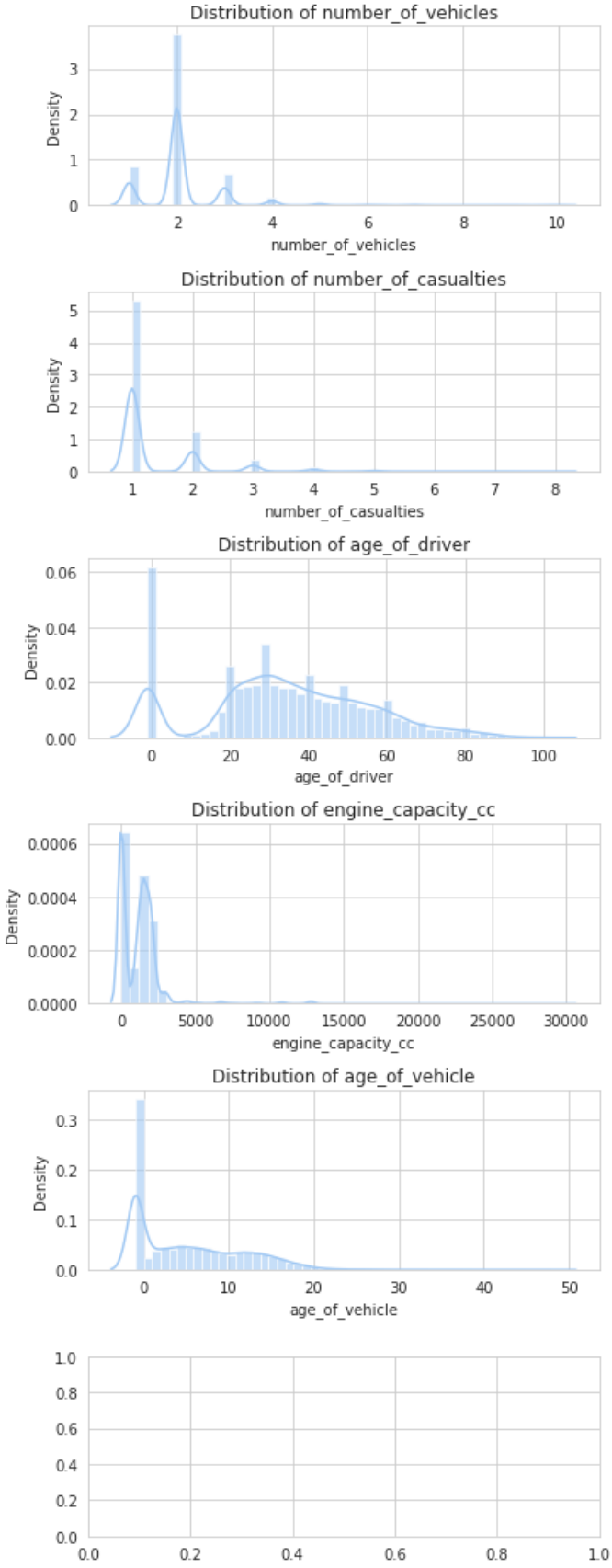
Checking the descriptive statistics for target variables

5.2 UNIVARIATE VISUALISATION

Data Visualisation for all numerical variables

In [ ]:
```
# Set the style and color palette
sns.set_style('whitegrid')
sns.set_palette('pastel')

# Get the numerical columns
num_cols = training_set.select_dtypes(include='number').columns
```

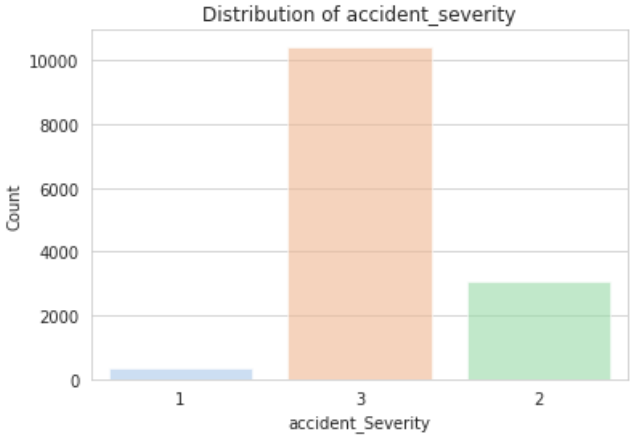
```
# Create a histogram of each numerical column in a separate subplot
fig, axs = plt.subplots(6, 1, figsize=(6, 15))
axs = axs.flatten()
for i, col in enumerate(num_cols):
    sns.distplot(training_set[col], ax=axs[i], kde=True, hist_kws={'alpha': 0.6})
    axs[i].set_title(f"Distribution of {col}")
    axs[i].set_xlabel(col)
    axs[i].set_ylabel("Density")

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



we can see the distribution of our variables in the below graphs.

Data Visualisation for target variable

In [ ]:
```
sns.countplot(data=training_set, x='accident_severity', alpha=0.6)
plt.title('Distribution of accident_severity')
plt.xlabel('accident_Severity')
plt.ylabel('Count')

plt.show()
```



we can observe from the above graph that accident_Severity for category 3 is maximum at a little more than 10000 count which is followed by category 2 but the count for it is at around 2500 count, lastly, there is category one where the count for accident severity is below 500 count.

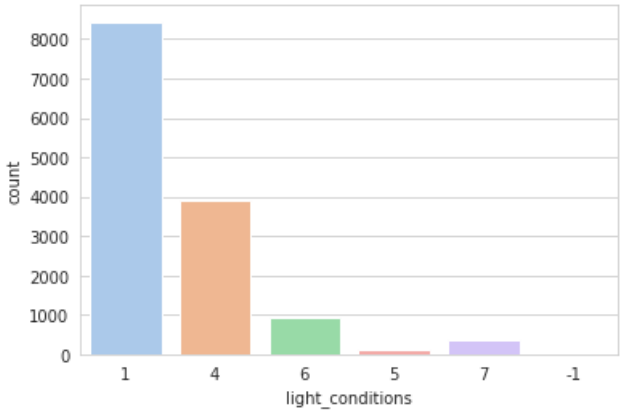
In [ ]:
```
training_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 13805 entries, 14579 to 152939
Data columns (total 15 columns):
 #   Column                   Non-Null Count  Dtype         
---  ------                   --------------  -----         
 0   accident_index           13805 non-null  object        
 1   accident_severity        13805 non-null  object        
 2   number_of_vehicles       13805 non-null  int64         
 3   number_of_casualties     13805 non-null  int64         
 4   date                     13805 non-null  datetime64[ns]
 5   time                     13805 non-null  object        
 6   road_type                13805 non-null  object        
 7   light_conditions         13805 non-null  object        
 8   weather_conditions       13805 non-null  object        
 9   road_surface_conditions  13805 non-null  object        
 10  sex_of_driver            13805 non-null  object        
 11  age_of_driver            13805 non-null  int64         
 12  engine_capacity_cc       13805 non-null  int64         
 13  age_of_vehicle           13805 non-null  int64         
 14  casualty_severity        13805 non-null  object        
dtypes: datetime64[ns](1), int64(5), object(9)
memory usage: 1.7+ MB
```

Data Visualisation for all categorical variables, Plotting the Graphs

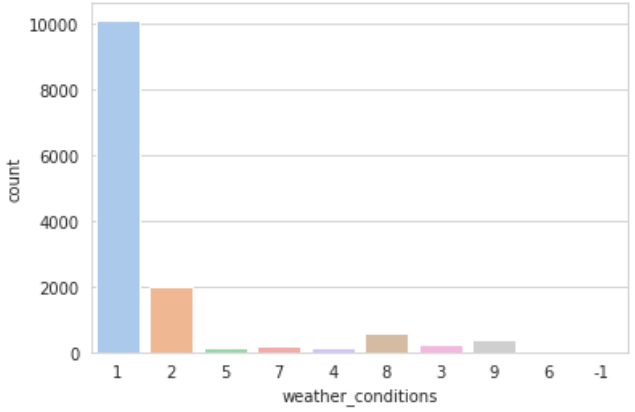
In [ ]:
```
sns.countplot(data=training_set, x='light_conditions')
```

Out[ ]:
```
<Axes: xlabel='light_conditions', ylabel='count'>
```



In [ ]:
```
sns.countplot(data=training_set, x='weather_conditions')
```

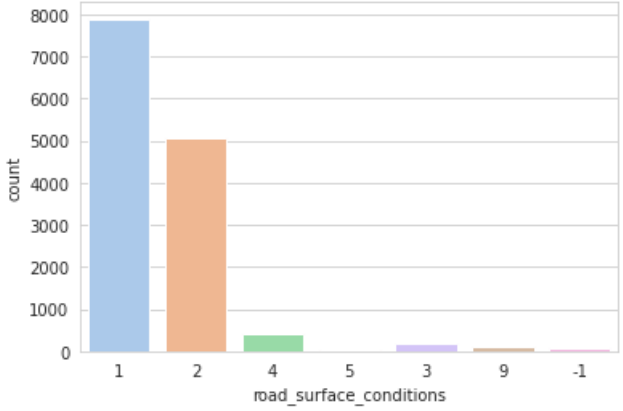
Out[ ]:
```
<Axes: xlabel='weather_conditions', ylabel='count'>
```



we can depict from the above graph that the count of accidents in the first type of weather_condition is maximum compared to all other categories at more than 10000 count, this is followed by category 2 of weather_condition where the number of count of accident is at 2000. all other weather condition category have count accident under 500.

In [ ]:
```
sns.countplot(data=training_set, x='road_surface_conditions')
```

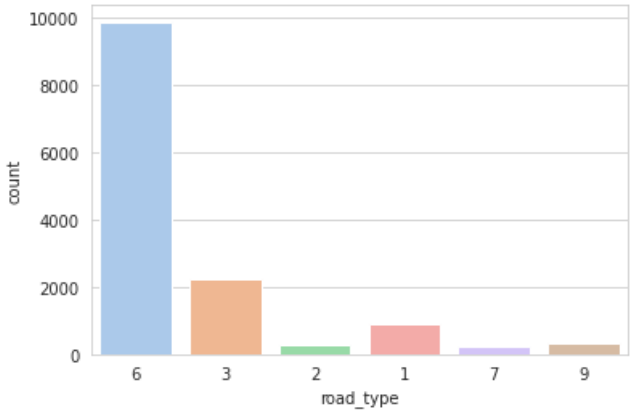
Out[ ]:
```
<Axes: xlabel='road_surface_conditions', ylabel='count'>
```



we can observe in the above graph that the count of accidents in the road_surface_condition of category 1 are the highes at almost 8000 which is followed by 5000 count in the category 2. the count are less than 500 for categories 4,3 and 9 while in road surface condition category 5 the count of accidents is zero.

In [ ]:
```
sns.countplot(data=training_set, x='road_type')
```

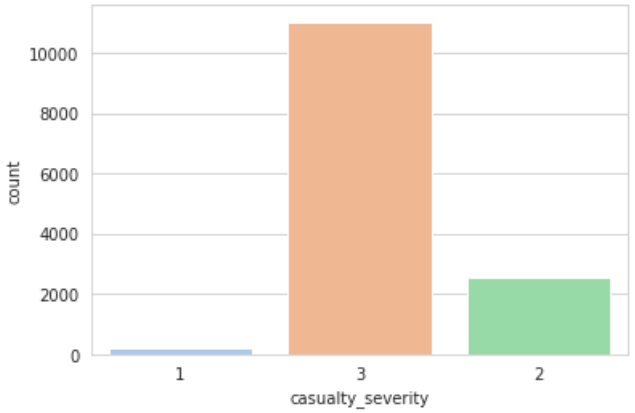
Out[ ]:
```
<Axes: xlabel='road_type', ylabel='count'>
```



it is evident from above graph that number of accidents in the road_type of category 6 are maximum at almost 10000 count, in all other categories the count is below 500 except for the road_type category 3 where the count is a little more than 2000.

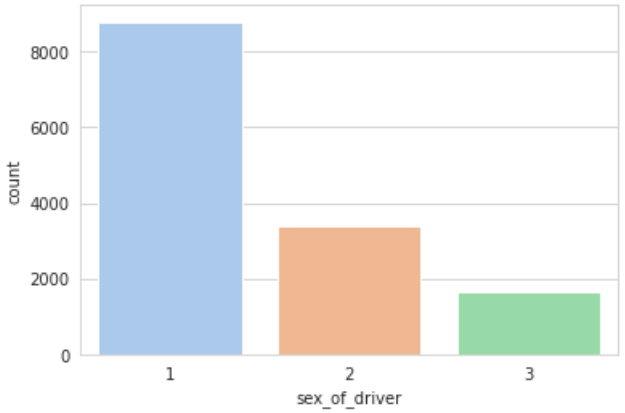
In [ ]:
```
sns.countplot(data=training_set, x='casualty_severity')
```

Out[ ]:
```
<Axes: xlabel='casualty_severity', ylabel='count'>
```



In [ ]:
```
sns.countplot(data=training_set, x='sex_of_driver')
```

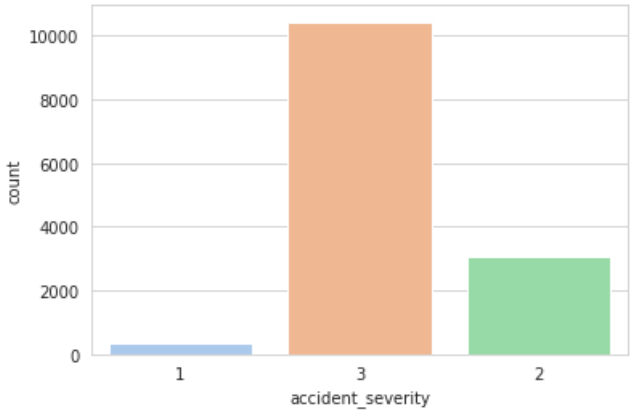
Out[ ]:
```
<Axes: xlabel='sex_of_driver', ylabel='count'>
```



we can observe in the above graph that the count of accident is maximum in the sex category 1 at almost 8500 which is followed by sex category 2 at count of 3500. lastly there is category 3 at little above 1500 count of accidents.

In [ ]:
```
sns.countplot(data=training_set, x='accident_severity')
```

Out[ ]:
```
<Axes: xlabel='accident_severity', ylabel='count'>
```



we can depict from the above graph that count of accidents is little above 10000 count for accident severity 3 while it is 2500 for accident severity 2 and less than 100 count for accident severity 1

EXPLORATORY DATA ANALYSIS

5.3 FEATURE ENGINEERING

In [ ]:
```
#Exploratory data analysis
sns.set_palette(palette=["#808282", "#C2CD23", "#918BC3"])
```

Checking for number of accidents on each day of the week in a month

In [ ]:
```
#Number of accidents on each day of the week
Monthly_accidents=training_set['date'].dt.month_name().value_counts().sort_index()

Monthly_accidents
```

Out[ ]:
```
December    6699
January     7106
Name: date, dtype: int64
```

From the above graph we can see that more accidents occur on friday followed by thursday, Tuesday being the lowest number of accidents followed by Sunday.

We create new columns for month, day of week, and hour of day to get a better understanding of the pattern of accidents being occurred though graphical presentation.

In [ ]:
```
training_set['Month'] =df_final['date'].dt.month_name()
training_set['Day'] = df_final['date'].dt.day_name()
training_set['HourOfDay'] = df_final['time'].str[:2].astype(int)
```

In [ ]:
```
testing_set['Month'] =testing_set['date'].dt.month_name()
testing_set['Day'] = testing_set['date'].dt.day_name()
testing_set['HourOfDay'] = testing_set['time'].str[:2].astype(int)
```

In [ ]:
```
print(training_set.shape)
print(testing_set.shape)
```

```
(13805, 18)
(3452, 18)
```

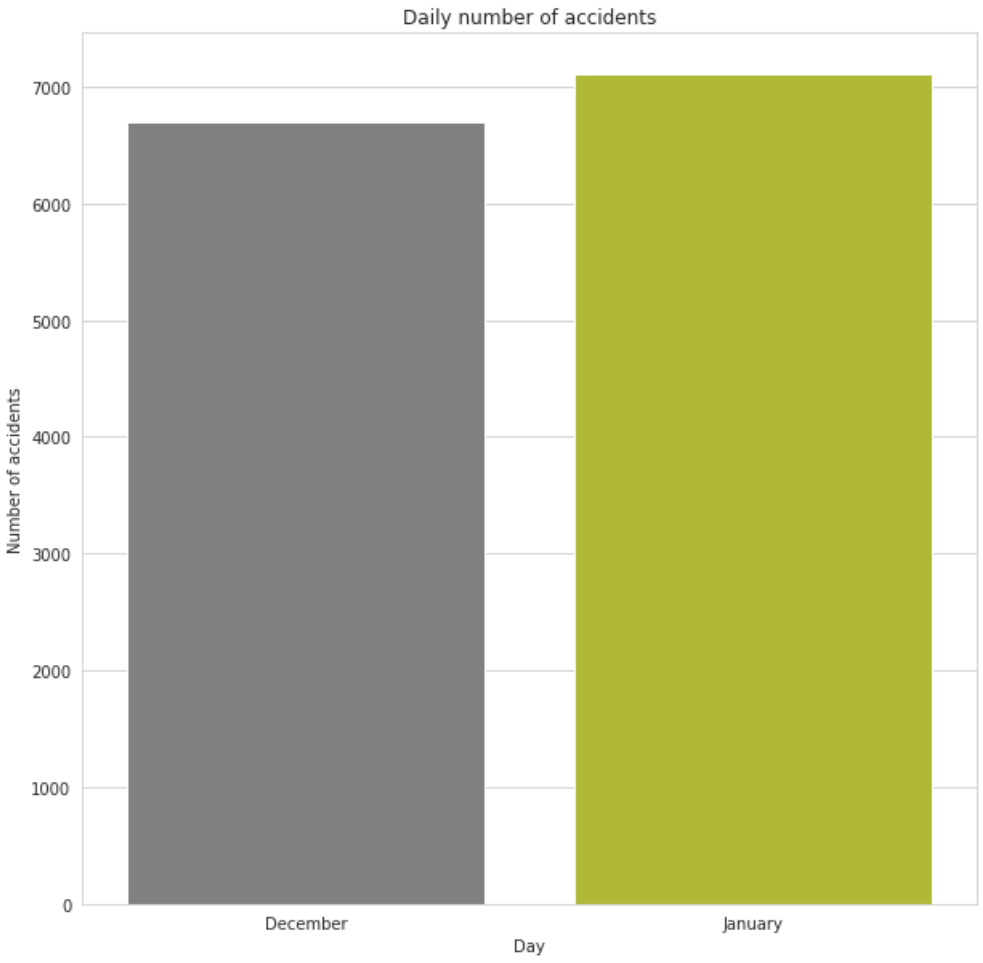
Counting the number of accidents per day on a hourly basis

In [ ]:
```
day_counts = training_set['Day'].value_counts().sort_index()
hour_counts = training_set['HourOfDay'].value_counts().sort_index()
```

5.4 BI-VARIATE VISUALISATION

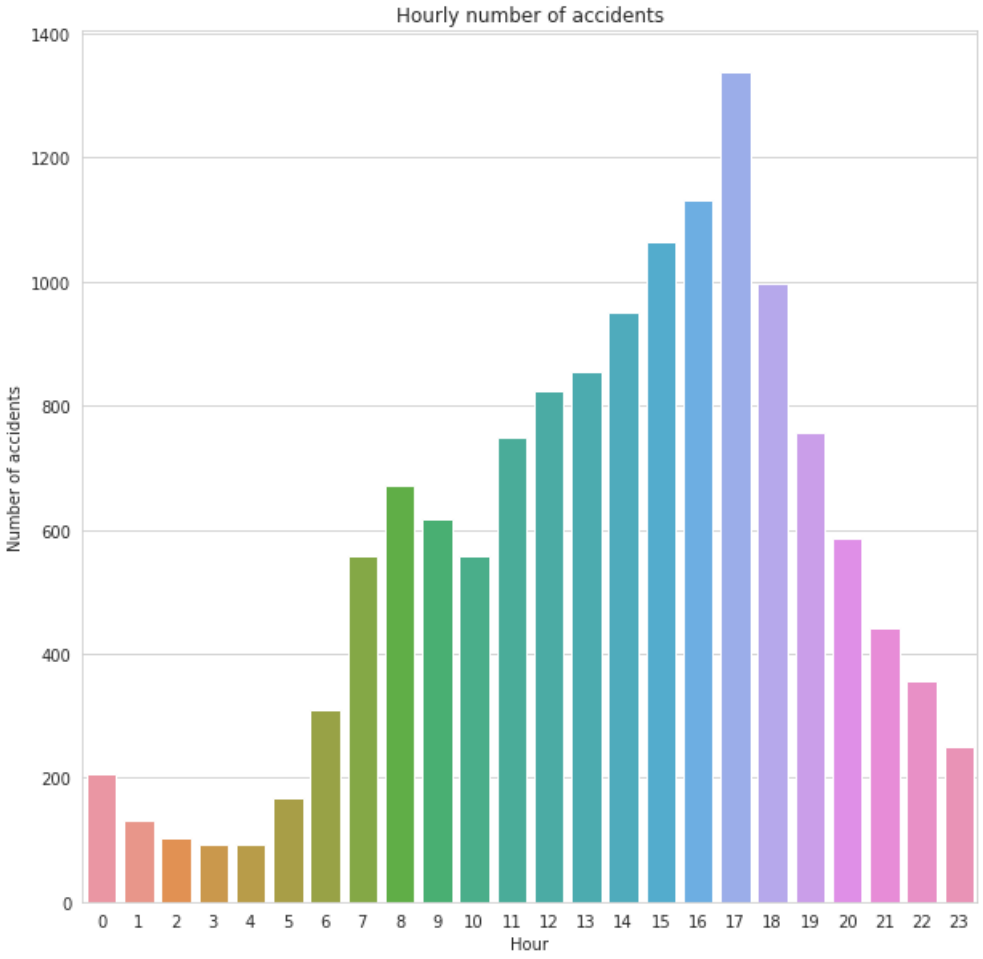
In [ ]:
```
fig, axs = plt.subplots(figsize=(10, 10))
sns.barplot(x=Monthly_accidents.index, y=Monthly_accidents.values)
plt.title('Daily number of accidents')
plt.xlabel('Day')
plt.ylabel('Number of accidents')
```

Out[ ]:
```
Text(0, 0.5, 'Number of accidents')
```



in the above graph we have data on number of accidents for the months of december and january. it is evident that the number of accidents in january is more than 7000 which is more compared to the number of accidents in december at around 6500.

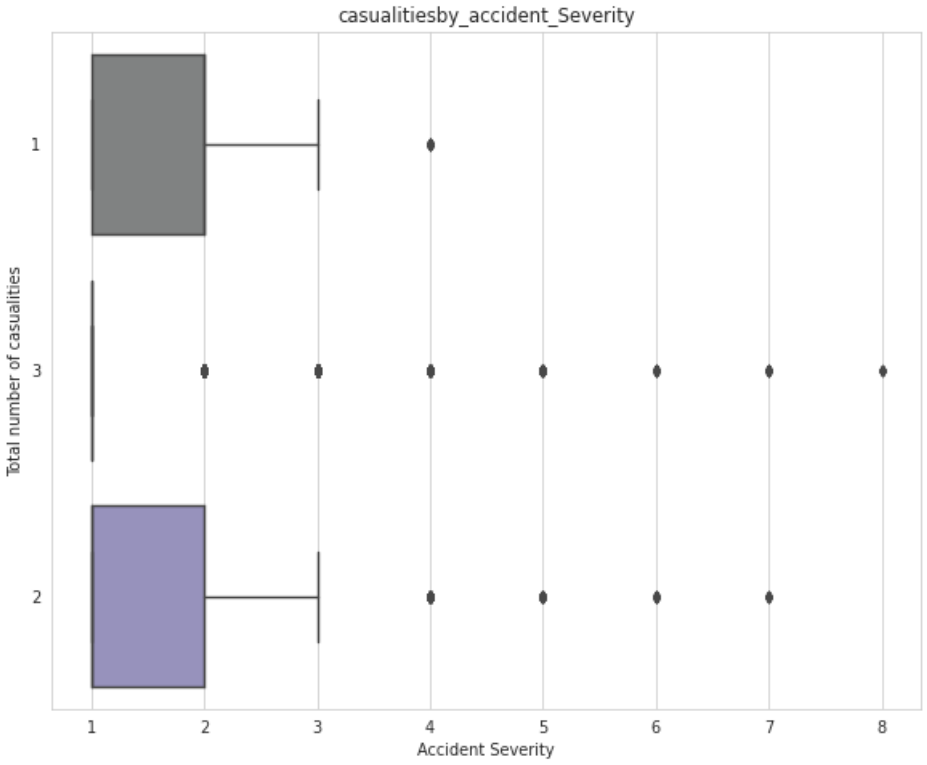
In [ ]:
```
# Plot the results
#sns.barplot(x=day_counts.index, y=day_counts.values)
fig, ax = plt.subplots(figsize=[10, 10])

# Plot barplot using seaborn
sns.barplot(x=hour_counts.index, y=hour_counts.values)
plt.title('Hourly number of accidents')
plt.xlabel('Hour')
plt.ylabel('Number of accidents')
plt.show()
```



The above is the graph for number of accidents on a hourly basis where it shows starting from 15:00 to 18:00 being the evening peak time with high accident count of which 17:00 alone with a accident count of above 2000.

In [ ]:
```
plt.figure(figsize=(10,8), dpi=70)



#Plotting boxplot using seaborn
ax1=sns.boxplot(y='accident_severity', x='number_of_casualties', data=training_set )
ax1.set_title('casualitiesby_accident_Severity')
ax1.set_xlabel('Accident Severity')
ax1.set_ylabel('Total number of casualities')
```

Out[ ]:
```
Text(0, 0.5, 'Total number of casualities')
```

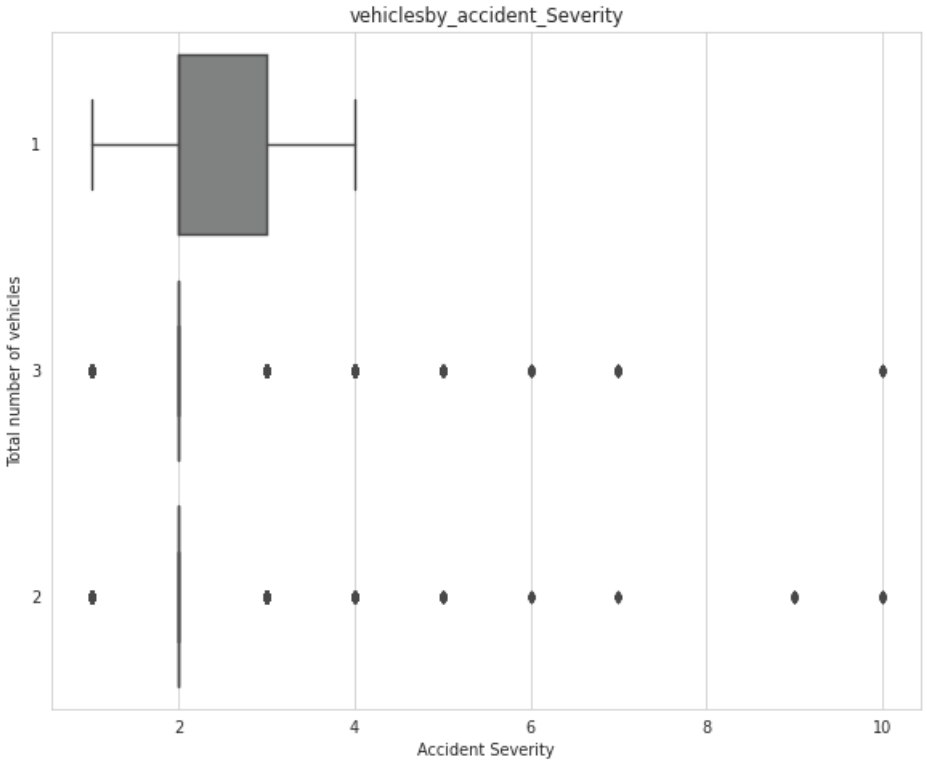


In [ ]:
```
training_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 13805 entries, 14579 to 152939
Data columns (total 18 columns):
 #   Column                   Non-Null Count  Dtype         
---  ------                   --------------  -----         
 0   accident_index           13805 non-null  object        
 1   accident_severity        13805 non-null  object        
 2   number_of_vehicles       13805 non-null  int64         
 3   number_of_casualties     13805 non-null  int64         
 4   date                     13805 non-null  datetime64[ns]
 5   time                     13805 non-null  object        
 6   road_type                13805 non-null  object        
 7   light_conditions         13805 non-null  object        
 8   weather_conditions       13805 non-null  object        
 9   road_surface_conditions  13805 non-null  object        
 10  sex_of_driver            13805 non-null  object        
 11  age_of_driver            13805 non-null  int64         
 12  engine_capacity_cc       13805 non-null  int64         
 13  age_of_vehicle           13805 non-null  int64         
 14  casualty_severity        13805 non-null  object        
 15  Month                    13805 non-null  object        
 16  Day                      13805 non-null  object        
 17  HourOfDay                13805 non-null  int64         
dtypes: datetime64[ns](1), int64(6), object(11)
memory usage: 2.0+ MB
```

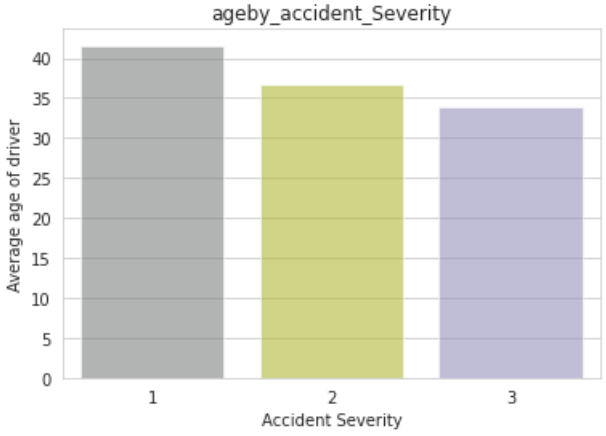
In [ ]:
```
plt.figure(figsize=(10,8), dpi=70)



#Plotting boxplot using seaborn
ax1=sns.boxplot(y='accident_severity', x='number_of_vehicles', data=training_set )
ax1.set_title('vehiclesby_accident_Severity')
ax1.set_xlabel('Accident Severity')
ax1.set_ylabel('Total number of vehicles')
```

Out[ ]:
```
Text(0, 0.5, 'Total number of vehicles')
```



we can observe from the above graph that as the severity of accident increases, the number of casualities also increases.

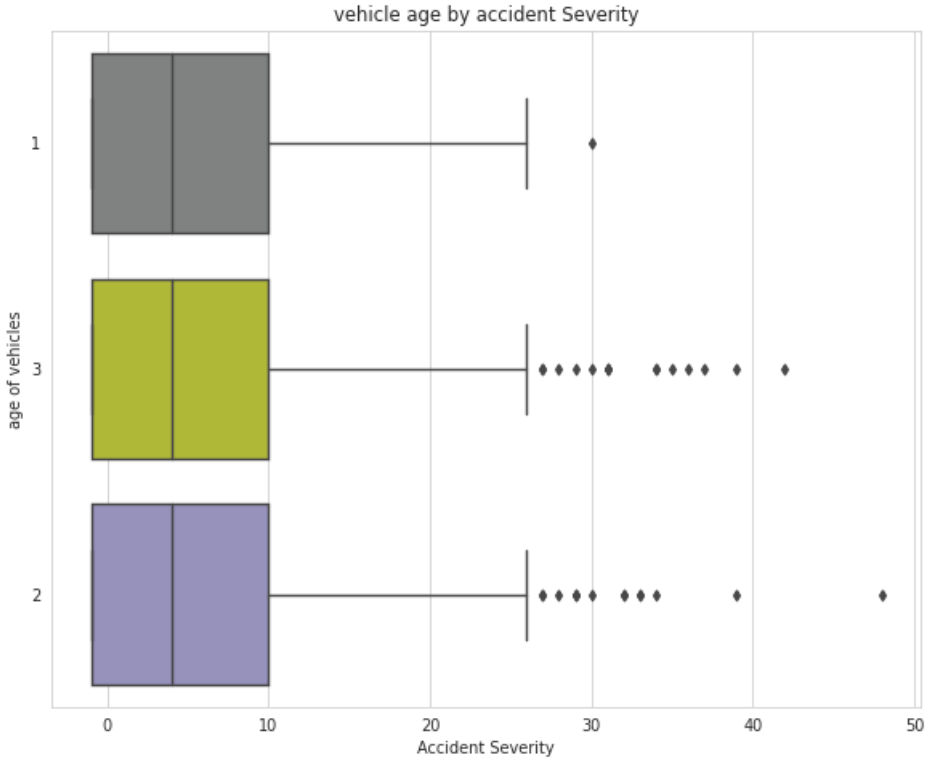
In [ ]:
```
ageby_accident_Severity = training_set.groupby('accident_severity')['age_of_driver'].mean()

# Plot the results
sns.barplot(x=ageby_accident_Severity.index, y=ageby_accident_Severity.values, alpha=0.6)

# Set the title and axis labels
plt.title('ageby_accident_Severity')
plt.xlabel('Accident Severity')
plt.ylabel('Average age of driver')

# Show the plot
plt.show()
```



The above graph shows us the data of accident severity to the average age of the driver, it can be depicted that accident severity of category 1 is met with people above 40 years of age while 2 and 3 category of accident severity happens with people having 40 years of age or below.

In [ ]:
```
plt.figure(figsize=(10,8), dpi=70)



#Plotting boxplot using seaborn
ax1=sns.boxplot(y='accident_severity', x='age_of_vehicle', data=training_set )
ax1.set_title('vehicle age by accident Severity')
ax1.set_xlabel('Accident Severity')
ax1.set_ylabel('age of vehicles')
```

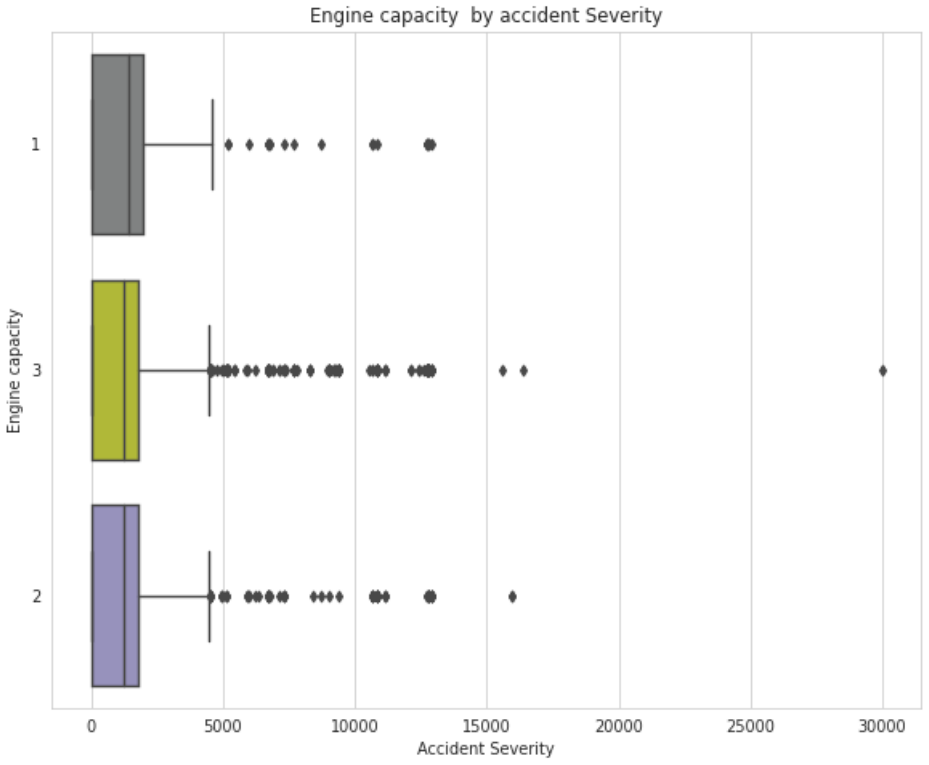
Out[ ]:
```
Text(0, 0.5, 'age of vehicles')
```



we can observe from the above graph that count of accident_Severity of category 2 is higest among all categories of accident severity, following that the 3rd age category of vehicle is not very far behind, however the accident severity for the 1st category of vehicle is less compared to the other two.

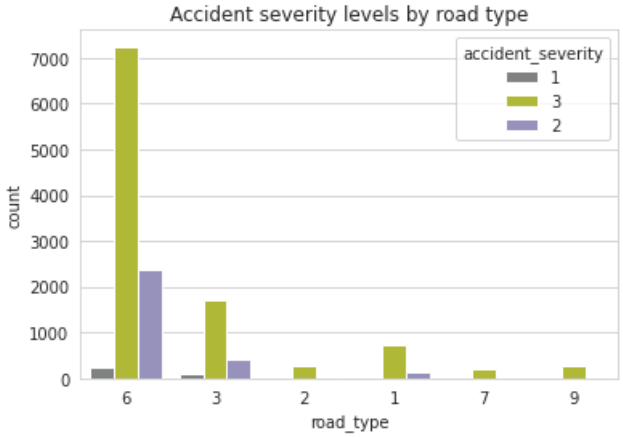
In [ ]:
```
plt.figure(figsize=(10,8), dpi=70)



#Plotting boxplot using seaborn
ax1=sns.boxplot(y='accident_severity', x='engine_capacity_cc', data=training_set )
ax1.set_title('Engine capacity  by accident Severity')
ax1.set_xlabel('Accident Severity')
ax1.set_ylabel('Engine capacity')
```

Out[ ]:
```
Text(0, 0.5, 'Engine capacity')
```



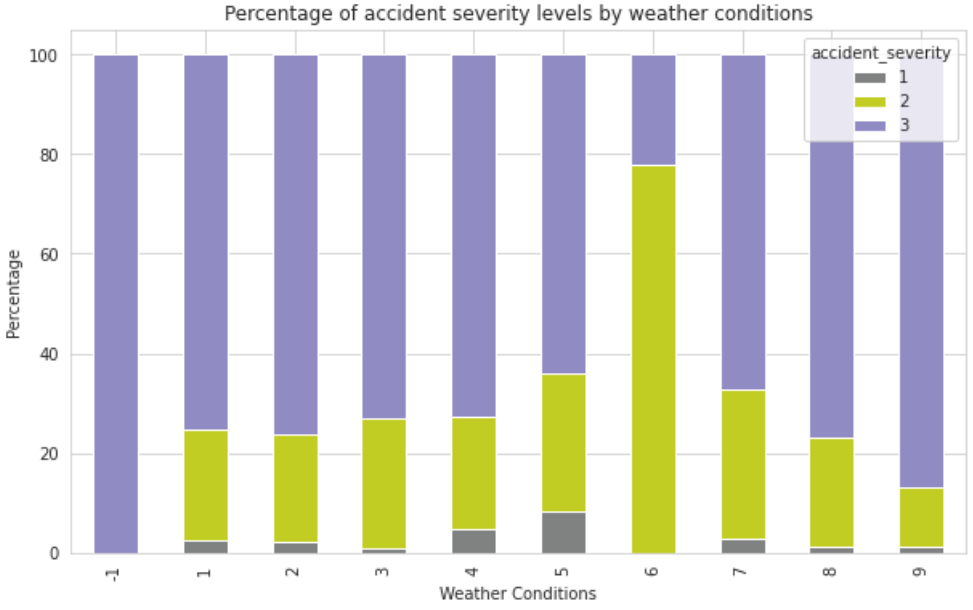
The above box plot explains us about the relation between engine capacity and accident severity where we can depict that accident severity between 0 to 3000 are happen almost equally in all 3 types of engine capacity. but, the 3rd type of engine capacity the accident severity has also reached 30000.

In [ ]:
```
sns.countplot(x='road_type', hue='accident_severity', data=training_set)
plt.title('Accident severity levels by road type')
plt.show()
```



we can observe from the above graph that count of accident_Severity of category 3 is higest among all categories of accident severity it has highes of 700 count in road type category 6 and for all the other road types its below 1800 count. for 2 category of accident severity the count is maximum for road type 6 and is below 300 count for rest of the road types . the accident severity of category 1 is below 200 count for all type of road types.

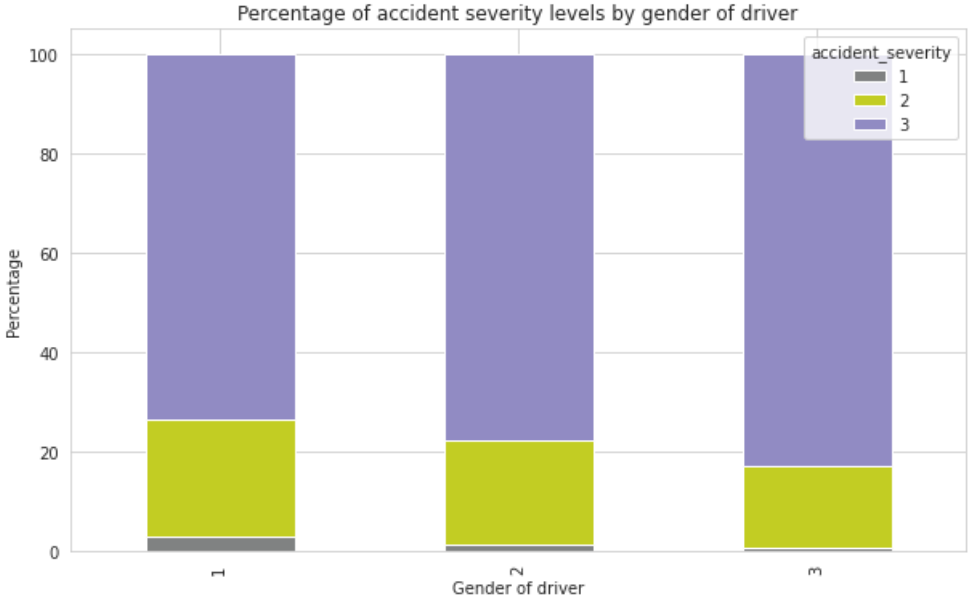
In [ ]:
```
# Calculate the percentages for each category combination
counts = training_set.groupby(['weather_conditions', 'accident_severity']).size()
percentages = counts.groupby(level=0).apply(lambda x: 100 * x / float(x.sum())).unstack()

# Plot the percentages as a stacked bar chart
ax = percentages.plot(kind='bar', stacked=True, figsize=(10, 6))

# Add labels and title
ax.set_xlabel('Weather Conditions')
ax.set_ylabel('Percentage')
ax.set_title('Percentage of accident severity levels by weather conditions')
plt.show()
```



From the above graph we can observe that accident_severity of category 3 is maximum in all type of weather conditions , accident severity of category 2 is slightly less than the 3rd category for all types of weather conditions. the severity of 1st category is minimum and below 10% for all types of weather conditions.

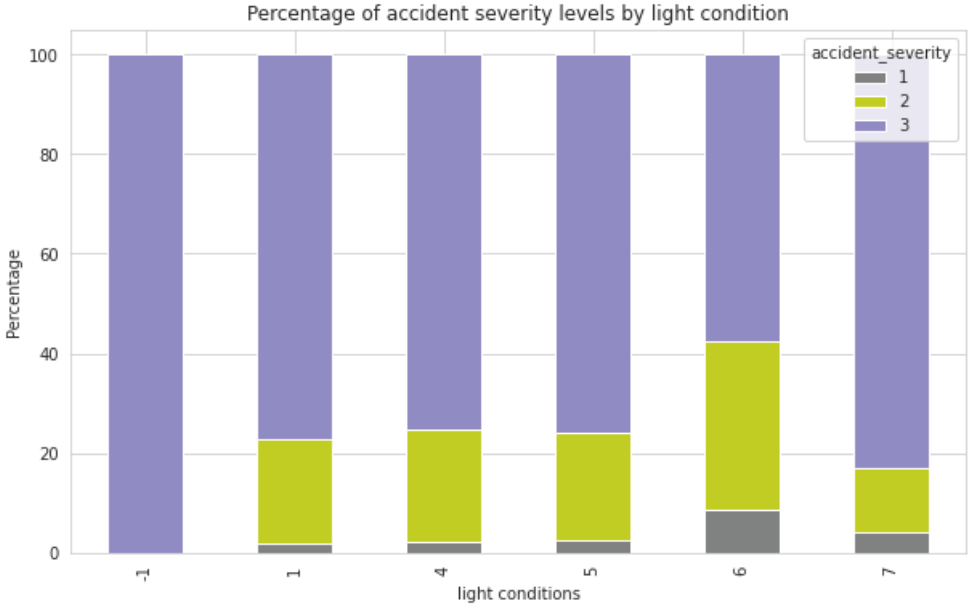
In [ ]:
```
# Calculate the percentages for each category combination
counts = training_set.groupby(['sex_of_driver', 'accident_severity']).size()
```

```
percentages = counts.groupby(level=0).apply(lambda x: 100 * x / float(x.sum())).unstack()

# Plot the percentages as a stacked bar chart
ax = percentages.plot(kind='bar', stacked=True, figsize=(10, 6))

# Add labels and title
ax.set_xlabel('Gender of driver')
ax.set_ylabel('Percentage')
ax.set_title('Percentage of accident severity levels by gender of driver')
plt.show()
```



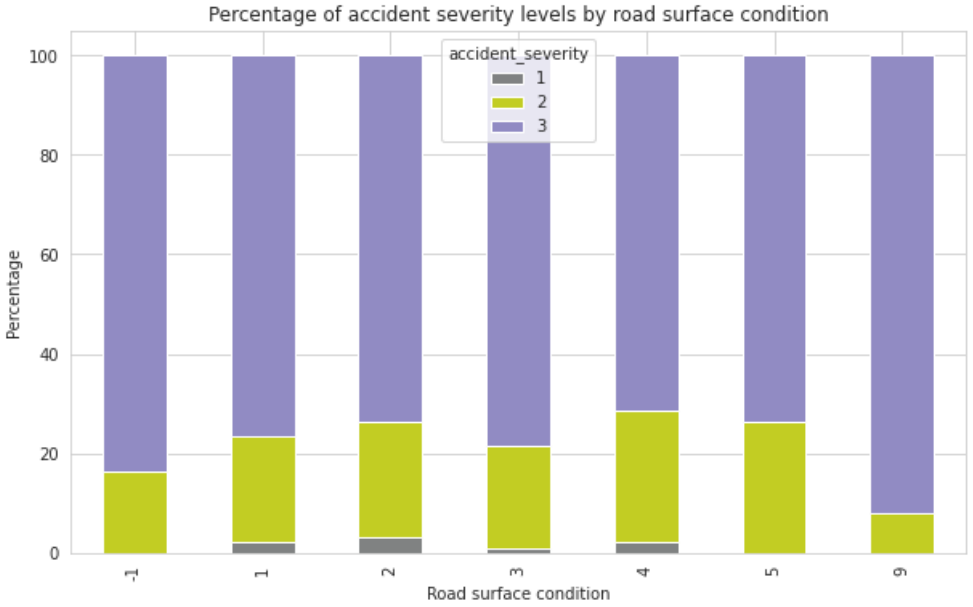
Through this grap we can find the majority of accidents commited by male are high.

In [ ]:
```
# Calculate the percentages for each category combination
counts = training_set.groupby(['light_conditions', 'accident_severity']).size()
percentages = counts.groupby(level=0).apply(lambda x: 100 * x / float(x.sum())).unstack()

# Plot the percentages as a stacked bar chart
ax = percentages.plot(kind='bar', stacked=True, figsize=(10, 6))

# Add labels and title
ax.set_xlabel('light conditions')
ax.set_ylabel('Percentage')
ax.set_title('Percentage of accident severity levels by light condition')
plt.show()
```



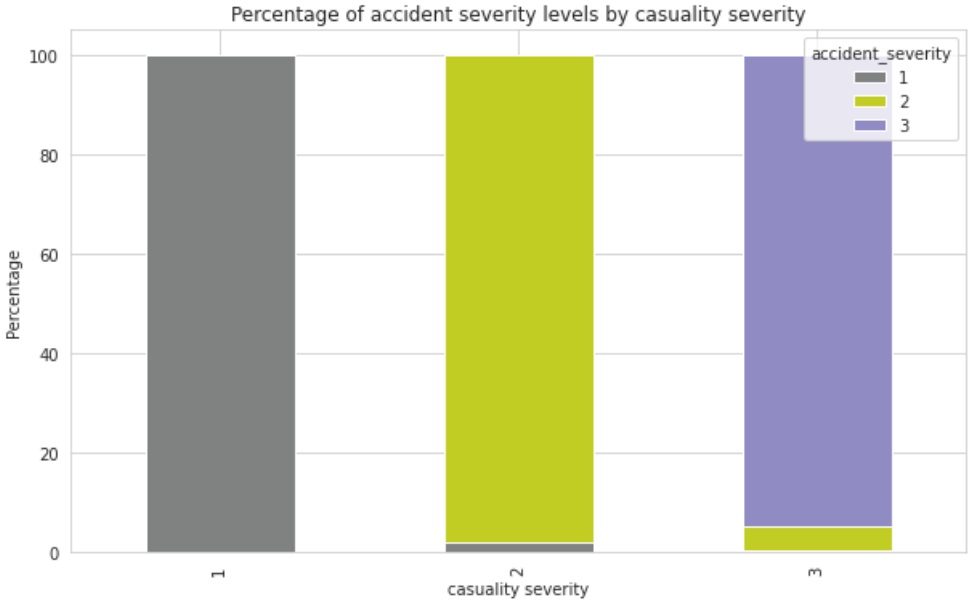
we can depict from the above graph that accident severity of category 3 is maximum in all light conditions which is followed by 2nd category of accident severity at around 20 percentage for all light condition except 1. lastly, accident severity of category 1 is less than 10 percentage in all light conditions.

In [ ]:
```
# Calculate the percentages for each category combination
counts = training_set.groupby(['road_surface_conditions', 'accident_severity']).size()
percentages = counts.groupby(level=0).apply(lambda x: 100 * x / float(x.sum())).unstack()

# Plot the percentages as a stacked bar chart
ax = percentages.plot(kind='bar', stacked=True, figsize=(10, 6))

# Add labels and title
ax.set_xlabel('Road surface condition')
ax.set_ylabel('Percentage')
ax.set_title('Percentage of accident severity levels by road surface condition')
plt.show()
```



we can depict from the above graph that accident severity of category 3 has maximum percentage in all Road Surface condition. which is followed by 2nd category of accident severity at around 20 percentage for all Road Surface condition expect 9. lastly, accident severity of category 1 is less than 10 percentage in all Road Surface conditions.

In [ ]:
```
# Calculate the percentages for each category combination
counts = training_set.groupby(['casualty_severity', 'accident_severity']).size()
percentages = counts.groupby(level=0).apply(lambda x: 100 * x / float(x.sum())).unstack()

# Plot the percentages as a stacked bar chart
ax = percentages.plot(kind='bar', stacked=True, figsize=(10, 6))

# Add labels and title
ax.set_xlabel('casuality severity')
ax.set_ylabel('Percentage')
ax.set_title('Percentage of accident severity levels by casuality severity')
plt.show()
```



we can interpret from the above graph that for all type of accident severity category have equal percentage of accident pecentage.

In [ ]:
```
pd.crosstab(index=training_set['casualty_severity'], columns=training_set['accident_severity'])
```

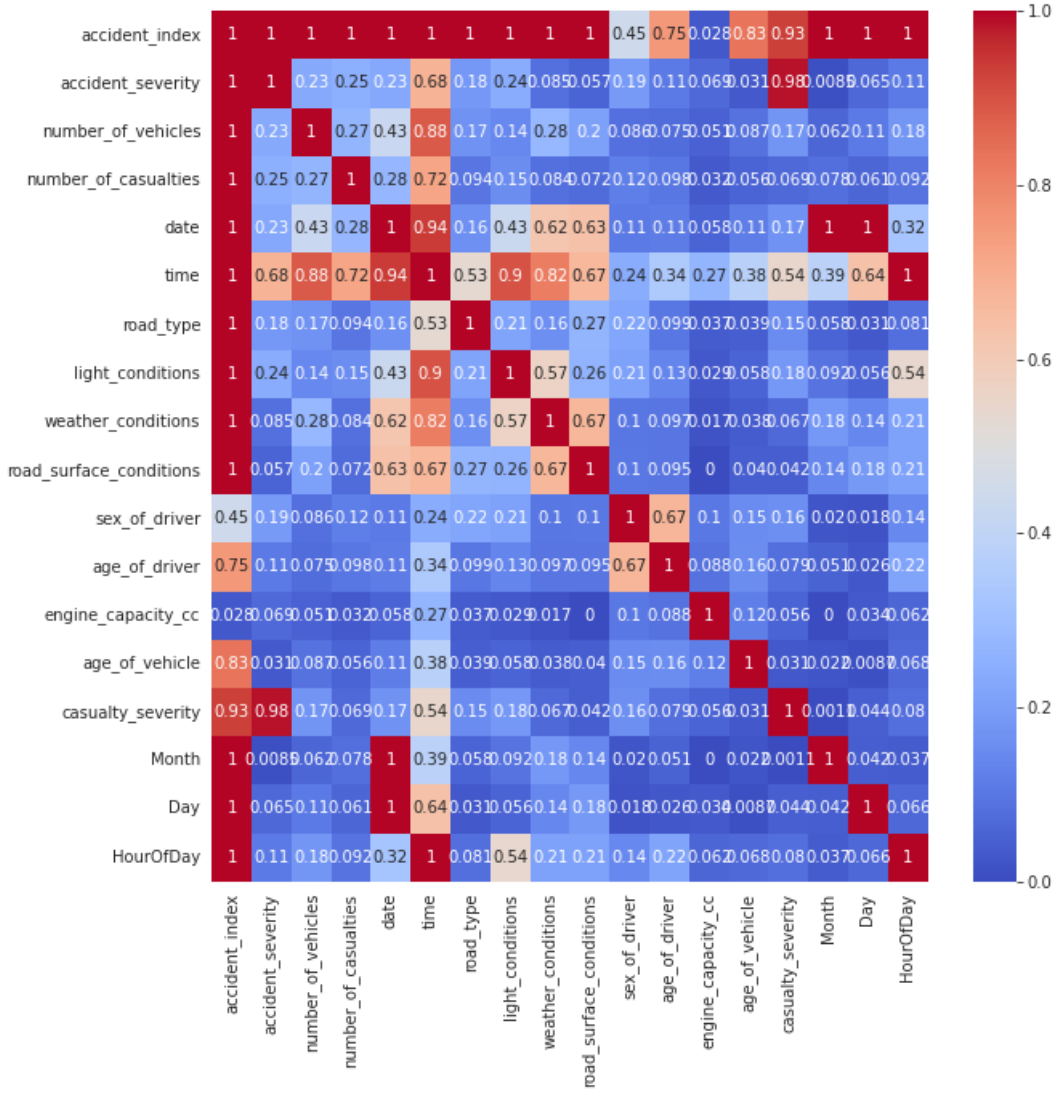
Out[ ]:

accident_severity	1	2	3
casualty_severity			
1	220	0	0
2	55	2513	0
3	55	532	10430

In [ ]:
```
import phik
```

In [ ]:
```
# create the correlation matrix
corr_matrix = training_set.phik_matrix()

# create a heatmap of the correlation matrix
fig, ax = plt.subplots(figsize=(10, 10))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')

# show the plot
plt.show()
```

```
interval columns not set, guessing: ['number_of_vehicles', 'number_of_casualties', 'age_of_driver', 'engine
_capacity_cc', 'age_of_vehicle', 'HourOfDay']
```



from the above heat map of correlation matrix we can interpret that:

- time and number of vehicles are highly correlated (0.88),

- time and weather condition are correlated (0.82) as weather is also related to what time of the day it is, the weather can differ from morning to afternoon and evening as it can get colder in evening and early morning but gets hotter in afternoon.
- the correlation between light condition and time is also very high at (0.9) as it is obvious that visiblity during the night time is very low and therefore the number of accidents increases but visiblity is good during the morning and afternoon time which corresponds to less number of accidents.
- Road surface and weather condition have correlation of (0.67) which is also significant. The road gets wear and tear in rainy season while it gets slippery during snow therefore all this condition affects the driving of vehicles and results into accidents. -age of the driver and the sex of the driver have correlation of (0.67) as older men tend to drive more as compared to older women.

6.MISSING VALUES AND OUTLIERS TREATMENT

6.1 MISSING VALUE TREATMENT

In [ ]:
```
training_set1= training_set.replace(-1,None)
testing_set1= testing_set.replace(-1,None)
```

In [ ]:
```
training_set1.isna().sum()
# determining the sum of na for every column in training set.
```

Out[ ]:
```
accident_index                0
accident_severity             0
number_of_vehicles            0
number_of_casualties          0
date                          0
time                          0
road_type                     0
light_conditions              0
weather_conditions            0
road_surface_conditions       0
sex_of_driver                 0
age_of_driver              1903
engine_capacity_cc         4696
age_of_vehicle             4630
casualty_severity             0
Month                         0
Day                           0
HourOfDay                     0
dtype: int64
```

In [ ]:
```
testing_set1.isna().sum()
# determining the sum of na for every column in testing set.
```

Out[ ]:
```
accident_index                0
accident_severity             0
number_of_vehicles            0
number_of_casualties          0
date                          0
time                          0
road_type                     0
light_conditions              0
weather_conditions            0
road_surface_conditions       0
sex_of_driver                 0
age_of_driver               513
engine_capacity_cc         1163
age_of_vehicle             1138
casualty_severity             0
Month                         0
Day                           0
HourOfDay                     0
dtype: int64
```

In [ ]:
```
final_trainset=training_set1.dropna()
# dropping na from the training dataset.
```

In [ ]:
```
final_testset=testing_set1.dropna()
#dropping na from the testing dataset.
```

In [ ]:
```
final_trainset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8159 entries, 14579 to 99881
Data columns (total 18 columns):
 #   Column                   Non-Null Count  Dtype         
---  ------                   --------------  -----         
 0   accident_index           8159 non-null   object        
 1   accident_severity        8159 non-null   object        
 2   number_of_vehicles       8159 non-null   int64         
 3   number_of_casualties     8159 non-null   int64         
 4   date                     8159 non-null   datetime64[ns]
 5   time                     8159 non-null   object        
 6   road_type                8159 non-null   object        
 7   light_conditions         8159 non-null   object        
 8   weather_conditions       8159 non-null   object        
 9   road_surface_conditions  8159 non-null   object        
 10  sex_of_driver            8159 non-null   object        
 11  age_of_driver            8159 non-null   object        
 12  engine_capacity_cc       8159 non-null   object        
 13  age_of_vehicle           8159 non-null   object        
 14  casualty_severity        8159 non-null   object        
 15  Month                    8159 non-null   object        
 16  Day                      8159 non-null   object        
 17  HourOfDay                8159 non-null   int64         
dtypes: datetime64[ns](1), int64(3), object(14)
memory usage: 1.2+ MB
```

In [ ]:
```
final_trainset[['age_of_driver','engine_capacity_cc','age_of_vehicle']]=final_trainset[['age_of_driver','eng
final_testset[['age_of_driver','engine_capacity_cc','age_of_vehicle']]=final_testset[['age_of_driver','engir
```

In [ ]:
```
final_trainset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8159 entries, 14579 to 99881
Data columns (total 18 columns):
 #   Column                   Non-Null Count  Dtype         
---  ------                   --------------  -----         
 0   accident_index           8159 non-null   object        
 1   accident_severity        8159 non-null   object        
 2   number_of_vehicles       8159 non-null   int64         
 3   number_of_casualties     8159 non-null   int64         
 4   date                     8159 non-null   datetime64[ns]
 5   time                     8159 non-null   object        
 6   road_type                8159 non-null   object        
 7   light_conditions         8159 non-null   object        
 8   weather_conditions       8159 non-null   object        
 9   road_surface_conditions  8159 non-null   object        
 10  sex_of_driver            8159 non-null   object        
 11  age_of_driver            8159 non-null   int64         
 12  engine_capacity_cc       8159 non-null   int64         
 13  age_of_vehicle           8159 non-null   int64         
 14  casualty_severity        8159 non-null   object        
 15  Month                    8159 non-null   object        
 16  Day                      8159 non-null   object        
 17  HourOfDay                8159 non-null   int64         
dtypes: datetime64[ns](1), int64(6), object(11)
memory usage: 1.2+ MB
```

6.2 OUTLIER DETECTION AND TREATMENT

In [ ]:
```
def outlier_treat(data, numerical, factor):

    for col in numerical:

        Q1 = data[col].quantile(0.25)
        Q3 = data[col].quantile(0.75)

        iqr = Q3 - Q1

        upper_whisk = Q3 + (factor*iqr)
        lower_whisk = Q1 - (factor*iqr)

        data[col] = np.where(data[col]>upper_whisk, upper_whisk,
                 np.where(data[col]<lower_whisk, lower_whisk, data[col]))
```

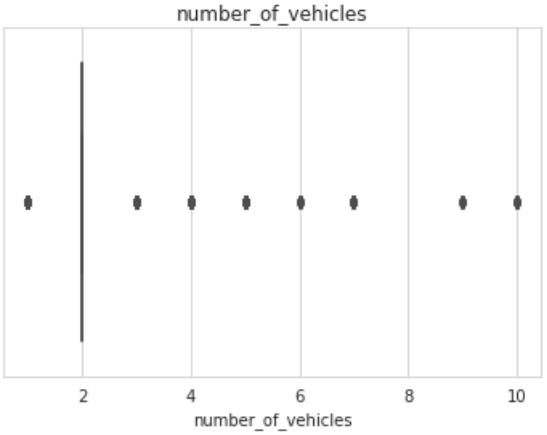
As outlier can affect the statistical analysis of the data and can result into data getting skewed we have detected and treated the outliers.

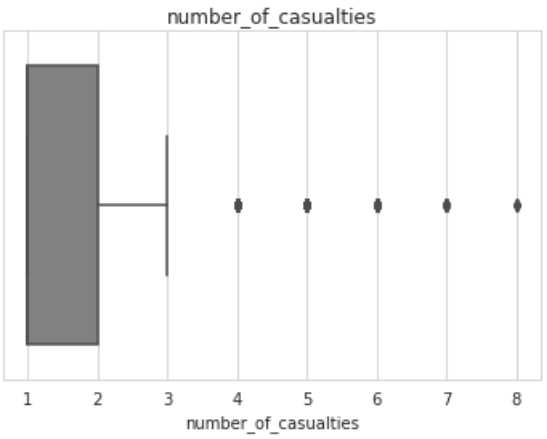
In [ ]:
```
numerical_column_1=final_trainset.select_dtypes(include=['int', 'float']).columns.tolist()
numerical_column_1
```

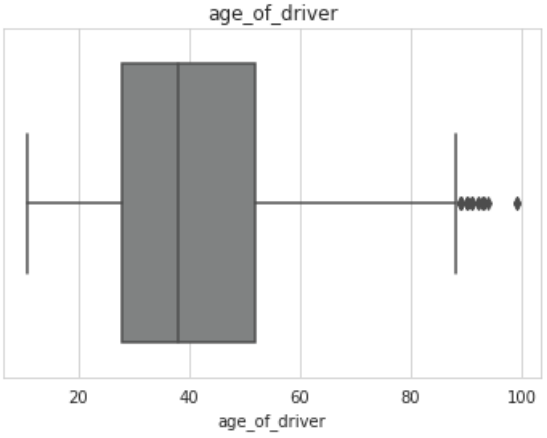
Out[ ]:
```
['number_of_vehicles',
 'number_of_casualties',
 'age_of_driver',
 'engine_capacity_cc',
 'age_of_vehicle',
 'HourOfDay']
```

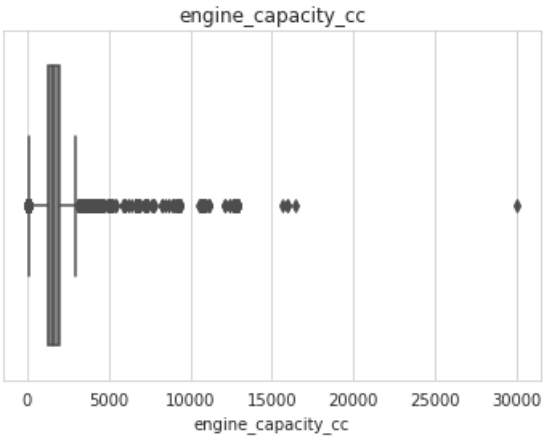
In [ ]:
```
#We use box plot to detect the outliers in the data
```

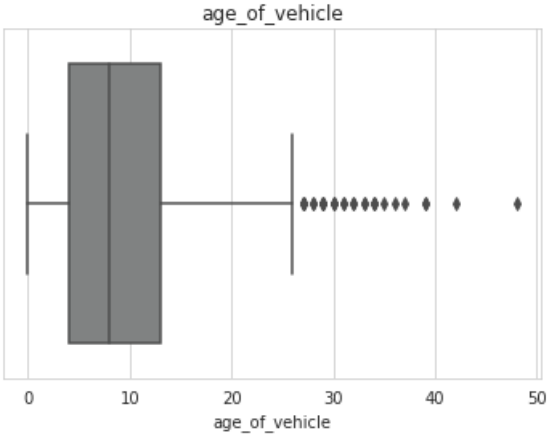
In [ ]:
```
for col in numerical_column_1:

  plt.figure(figsize=(6,4))
  sns.boxplot(x=final_trainset[col])
  plt.title(col)
  plt.show()
```

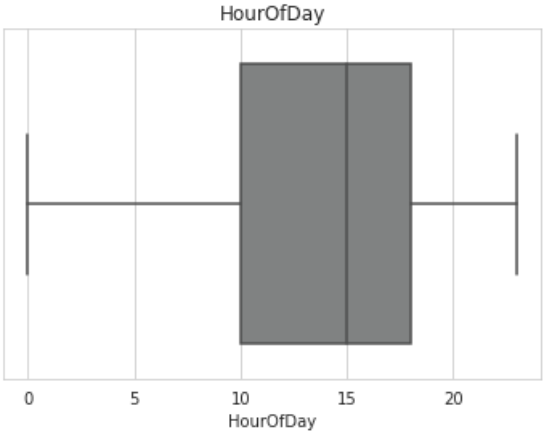












the above box plots gives us information on our data about the frequency of every variable,

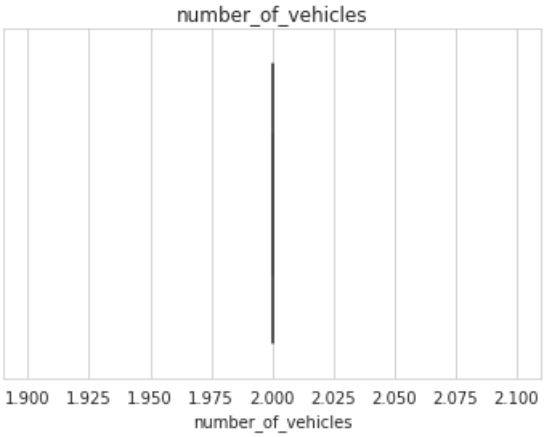
- we can interpret from the 2nd box plot that majority of casualities are 2.
- from the 3rd box plot we can observe that majority of drivers are in age category between 25 to 45.
- the majority of vehicles in our data has engine capacity between 1000 to 2000cc.
- The majority of vehicles have age between 5 to 12 years.

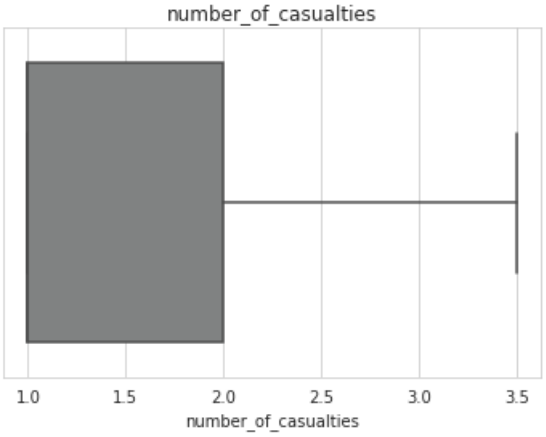
In [ ]:
```
numerical_column=['number_of_vehicles',
 'number_of_casualties',
 'age_of_driver',
 'engine_capacity_cc',
 'age_of_vehicle','HourOfDay'
 ]
```

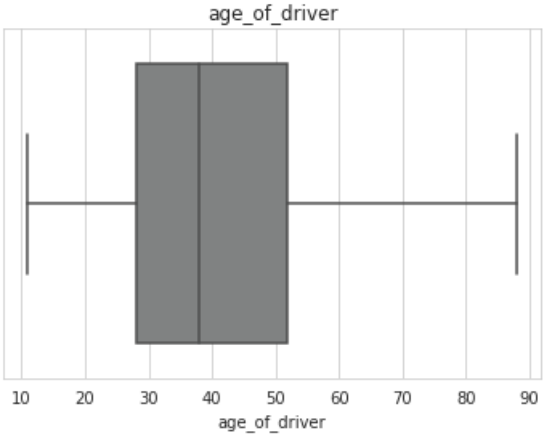
In [ ]:
```
#Outlier treatment is done using Inter Quartile Range capping
```

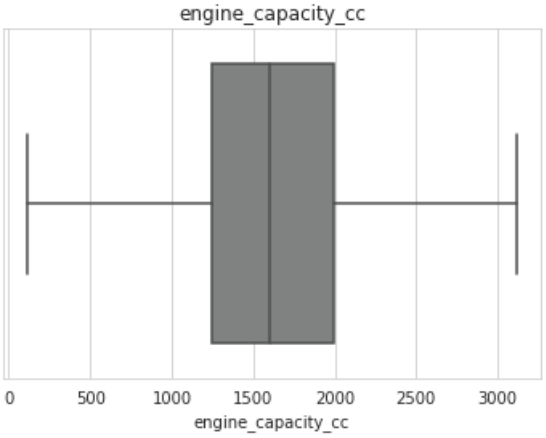
In [ ]:
```
outlier_treat(final_trainset,numerical_column, 1.5)
```

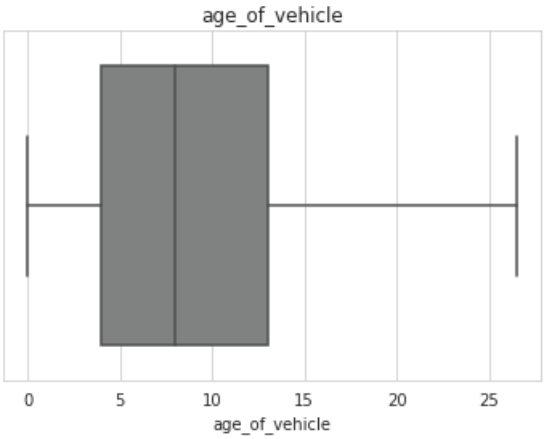
In [ ]:
```
for col in numerical_column_1:

  plt.figure(figsize=(6,4))
  sns.boxplot(x=final_trainset[col])
  plt.title(col)
  plt.show()
```

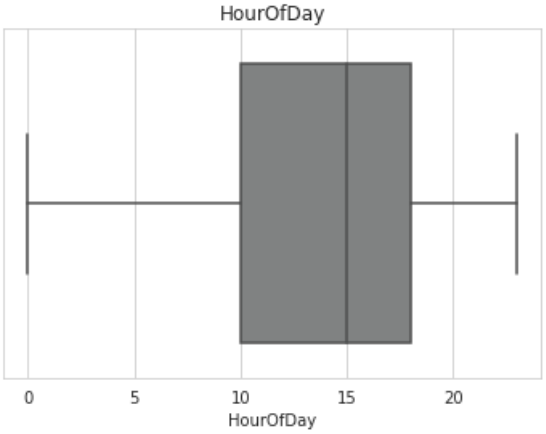

number_of_vehicles
1.900
1.925
1.950
1.975
2.000
2.025
2.050
2.075
2.100
number_of_vehicles


number_of_casualties
1.0
1.5
2.0
2.5
3.0
3.5
number_of_casualties


age_of_driver
10
20
30
40
50
60
70
80
90
age_of_driver


engine_capacity_cc
0
500
1000
1500
2000
2500
3000
engine_capacity_cc





we have managed the outliers in the above box plot so now we can interpret the exact frequencies of the variables.

In [ ]:
```
final_trainset.drop('number_of_vehicles',axis=1,inplace=True)
```

Dropping the column 'number_of_vehicles' from the final_trainset dataset.

In [ ]:
```
final_testset.drop('number_of_vehicles',axis=1,inplace=True)
```

Dropping the column 'number_of_vehicles' from the final_trainset dataset.

In [ ]:
```
training_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 13805 entries, 14579 to 152939
Data columns (total 18 columns):
 #   Column                   Non-Null Count  Dtype         
---  ------                   --------------  -----         
 0   accident_index           13805 non-null  object        
 1   accident_severity        13805 non-null  object        
 2   number_of_vehicles       13805 non-null  int64         
 3   number_of_casualties     13805 non-null  int64         
 4   date                     13805 non-null  datetime64[ns]
 5   time                     13805 non-null  object        
 6   road_type                13805 non-null  object        
 7   light_conditions         13805 non-null  object        
 8   weather_conditions       13805 non-null  object        
 9   road_surface_conditions  13805 non-null  object        
 10  sex_of_driver            13805 non-null  object        
 11  age_of_driver            13805 non-null  int64         
 12  engine_capacity_cc       13805 non-null  int64         
 13  age_of_vehicle           13805 non-null  int64         
 14  casualty_severity        13805 non-null  object        
 15  Month                    13805 non-null  object        
 16  Day                      13805 non-null  object        
 17  HourOfDay                13805 non-null  int64         
dtypes: datetime64[ns](1), int64(6), object(11)
memory usage: 2.0+ MB
```

7.DUMMY VARIABLE CREATION

We are doing label encoding for casuality severity as it is an Ordinal Variable

In [ ]:
```
nominal=['road_type','light_conditions','weather_conditions','road_surface_conditions','sex_of_driver','Mont
ordinal = ['casuality_severity']
```

In [ ]:
```
nomi_dummy=pd.get_dummies(final_trainset[nominal])
```

In [ ]:
```
nomi_dummy_test=pd.get_dummies(final_testset[nominal])
```

In [ ]:
```
from sklearn.preprocessing import LabelEncoder
```

In [ ]:
```
encoder = LabelEncoder()
```

In [ ]:
```
encoder.fit(final_trainset['casualty_severity'])
encoder.fit(final_testset['casualty_severity'])
```

Out[ ]:
```
▾ LabelEncoder
LabelEncoder()
```

In [ ]:
```
final_trainset['casualty_severity'] = encoder.transform(final_trainset['casualty_severity'])
```

In [ ]:
```
final_testset['casualty_severity'] = encoder.transform(final_testset['casualty_severity'])
```

In [ ]:
```
print(final_trainset.shape)
print(final_testset.shape)
```

```
(8159, 17)
(2027, 17)
```

In [ ]:
```
training_set_end = pd.concat([final_trainset, nomi_dummy], axis=1)
```

In [ ]:
```
testing_set_end = pd.concat([final_testset, nomi_dummy_test], axis=1)
```

8.EXPORTING THE DATA

In [ ]:
```
testing_set_end.to_csv('sample_data/trainset.csv')

training_set_end.to_csv('sample_data/testset.csv')
```

9.CONCLUSION

In this group assignment, we successfully created a data frame for the casuality and severity of accidents. we can now move forward to build our model.