road-acc-fin

March 31, 2023

0.1 Problem Statement

We are working with a dataset on accident severity, and our goal is to build a machine learning model that can accurately predict the severity of accidents based on a range of different features. Our main objective is to identify the most important features for predicting accident severity and to develop a model that can generalize well to new data.

The dataset contains information about road traffic accidents in the UK, from the year 2005 to 2021. The columns in the dataset contain the following information:

- accident_index: Unique identifier for each accident
- accident_year: Year in which the accident occurred
- accident_reference: Reference number for the accident
- location_easting_osgr: Easting (X-coordinate) of accident location in British National Grid
- location_northing_osgr: Northing (Y-coordinate) of accident location in British National Grid
- longitude: Longitude of accident location
- latitude: Latitude of accident location
- police_force: Police force that attended the accident
- accident_severity: Severity of the accident, ranging from 1 (fatal) to 3 (slight)
- number_of_vehicles: Number of vehicles involved in the accident
- number_of_casualties: Number of casualties in the accident
- date: Date of the accident
- day_of_week: Day of the week on which the accident occurred
- time: Time of the accident
- local_authority_district: Local authority district in which the accident occurred
- local_authority_ons_district: ONS code for the local authority district in which the accident occurred
- local_authority_highway: Highway authority responsible for the road on which the accident occurred
- first_road_class: Class of the first road involved in the accident, ranging from 1 (motorway) to 6 (other)
- first_road_number: Number of the first road involved in the accident
- road_type: Type of road on which the accident occurred, ranging from 1 (motorway) to 6 (footpath or bridleway)
- speed_limit: Speed limit on the road on which the accident occurred
- junction_detail: Type of junction at which the accident occurred
- junction_control: Control of the junction at which the accident occurred
- second_road_class: Class of the second road involved in the accident, ranging from 0 (not

at junction or within 20 metres) to 6 (other)
- second_road_number: Number of the second road involved in the accident
- pedestrian_crossing_human_control: Whether a pedestrian crossing with a human controller was present at the accident location
- pedestrian_crossing_physical_facilities: Whether a pedestrian crossing with physical facilities was present at the accident location
- light_conditions: Light conditions at the time of the accident
- weather_conditions: Weather conditions at the time of the accident
- road_surface_conditions: Road surface conditions at the time of the accident
- special_conditions_at_site: Whether any special conditions were present at the accident location
- carriageway_hazards: Whether any carriageway hazards were present at the accident location
- urban_or_rural_area: Whether the accident occurred in an urban or rural area
- did_police_officer_attend_scene_of_accident: Whether a police officer attended the scene of the accident
- trunk_road_flag: Whether the road on which the accident occurred is a trunk road
- lsoa_of_accident_location: Lower Super Output Area in which the accident occurred

This information can be used to understand the factors that contribute to road traffic accidents and to develop strategies to prevent such accidents in the future.

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[3]:
```
!unzip -q "/content/drive/MyDrive/code_files/Sri/road_acc_4/Data.zip"
```

[4]:
```
# Libraries to help with reading and manipulating data
import pandas as pd
import numpy as np

# Libaries to help with data visualization
import matplotlib.pyplot as plt
import seaborn as sns

# To tune model, get different metric scores, and split data
from sklearn.metrics import (
    f1_score,
    accuracy_score,
    recall_score,
    precision_score,
    confusion_matrix,
    roc_auc_score
)
from sklearn.model_selection import train_test_split, StratifiedKFold,␣
 ↪cross_val_score
```

```python
# To be used for data scaling and one hot encoding
from sklearn.preprocessing import StandardScaler, MinMaxScaler, OneHotEncoder
from sklearn import metrics

# To impute missing values
from sklearn.impute import SimpleImputer

# To oversample and undersample data
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler

# To do hyperparameter tuning
from sklearn.model_selection import RandomizedSearchCV



# To define maximum number of columns to be displayed in a dataframe
pd.set_option("display.max_columns", None)

# To help with model building
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

# To supress warnings
import warnings

warnings.filterwarnings("ignore")
```

[5]:
```python
data = pd.read_csv("/content/Data/Accidents-2021.csv")
```

[6]:
```python
data.head()
```

[6]:
```
   accident_index  accident_year accident_reference  location_easting_osgr  \
0  2021010000000.0           2021           10287148               521508.0
1  2021010000000.0           2021           10287149               535379.0
2  2021010000000.0           2021           10287151               529701.0
3  2021010000000.0           2021           10287155               525312.0
4  2021010000000.0           2021           10287157               512144.0

   location_northing_osgr  longitude   latitude  police_force  \
0                193079.0  -0.246102  51.623425             1
1                180783.0  -0.050574  51.509767             1
2                170398.0  -0.136152  51.417769             1
3                178385.0  -0.196411  51.490536             1
4                171526.0  -0.388169  51.431649             1
```

```
   accident_severity  number_of_vehicles  number_of_casualties      date  \
0                  3                   3                     1  1/1/2021
1                  2                   2                     3  1/1/2021
2                  2                   2                     4  1/1/2021
3                  1                   1                     1  1/1/2021
4                  3                   4                     1  1/1/2021

   day_of_week  time  local_authority_district local_authority_ons_district  \
0            6  2:05                        -1                    E09000003
1            6  3:30                        -1                    E09000030
2            6  4:07                        -1                    E09000022
3            6  4:26                        -1                    E09000020
4            6  3:10                        -1                    E09000018

  local_authority_highway  first_road_class  first_road_number  road_type  \
0               E09000003                 6                  0          6
1               E09000030                 3               1203          3
2               E09000022                 4                272          6
3               E09000020                 3               3220          2
4               E09000018                 5                  0          6

   speed_limit  junction_detail  junction_control  second_road_class  \
0           30                9                 4                  6
1           30                7                 2                  3
2           30                9                 2                  5
3           30                9                 4                  6
4           20                3                 4                  6

   second_road_number  pedestrian_crossing_human_control  \
0                   0                                  0
1                1204                                  0
2                   0                                  0
3                   0                                  0
4                   0                                  0

   pedestrian_crossing_physical_facilities  light_conditions  \
0                                        0                 4
1                                        5                 4
2                                        5                 4
3                                        4                 4
4                                        0                 4

   weather_conditions  road_surface_conditions  special_conditions_at_site  \
0                   7                        4                           1
1                   1                        1                           0
2                   1                        1                           0
3                   1                        1                           0
```

```
4                   1                        1                            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  trunk_road_flag  \
0                                            1                2
1                                            1                2
2                                            1                2
3                                            1                2
4                                            1                2

  lsoa_of_accident_location
0                 E01000263
1                 E01004303
2                 E01003146
3                 E01002847
4                 E01002608
```

[7]:
```
data.describe().T
```

[7]:
```
                                                count           mean  \
accident_year                                101087.0    2021.000000
location_easting_osgr                        101070.0  455370.590195
location_northing_osgr                       101070.0  274267.396596
longitude                                    101070.0      -1.204951
latitude                                     101070.0      52.355819
police_force                                 101087.0      27.055833
accident_severity                            101087.0       2.760286
number_of_vehicles                           101087.0       1.844382
number_of_casualties                         101087.0       1.268304
day_of_week                                  101087.0       4.138692
local_authority_district                     101087.0      -0.351756
first_road_class                             101087.0       4.204725
first_road_number                            101087.0     789.634899
road_type                                    101087.0       5.256769
speed_limit                                  101087.0      36.003146
junction_detail                              101087.0       4.385757
junction_control                             101087.0       1.771385
second_road_class                            101087.0       3.087697
second_road_number                           101087.0     223.660847
pedestrian_crossing_human_control            101087.0       0.362064
pedestrian_crossing_physical_facilities      101087.0       1.167648
```

```
light_conditions                             101087.0       1.976822
weather_conditions                           101087.0       1.651419
road_surface_conditions                      101087.0       1.346612
special_conditions_at_site                   101087.0       0.253128
carriageway_hazards                          101087.0       0.196623
urban_or_rural_area                          101087.0       1.319695
did_police_officer_attend_scene_of_accident  101087.0       1.450533
trunk_road_flag                              101087.0       1.721547

                                                       std          min  \
accident_year                                     0.000000  2021.000000
location_easting_osgr                         92901.603757 67564.000000
location_northing_osgr                       145053.875052 13898.000000
longitude                                         1.357913    -7.486852
latitude                                          1.307057    49.980835
police_force                                     24.235890     1.000000
accident_severity                                 0.459800     1.000000
number_of_vehicles                                0.680355     1.000000
number_of_casualties                              0.674544     1.000000
day_of_week                                       1.941061     1.000000
local_authority_district                         15.603753    -1.000000
first_road_class                                  1.464199     1.000000
first_road_number                              1583.067339     0.000000
road_type                                         1.701544     1.000000
speed_limit                                      14.121083    20.000000
junction_detail                                  13.915899    -1.000000
junction_control                                  2.528051    -1.000000
second_road_class                                 2.751656    -1.000000
second_road_number                              931.658373    -1.000000
pedestrian_crossing_human_control                 1.717190    -1.000000
pedestrian_crossing_physical_facilities           2.420433    -1.000000
light_conditions                                  1.685636    -1.000000
weather_conditions                                1.848077    -1.000000
road_surface_conditions                           0.957330    -1.000000
special_conditions_at_site                        1.345088    -1.000000
carriageway_hazards                               1.209476    -1.000000
urban_or_rural_area                               0.466721     1.000000
did_police_officer_attend_scene_of_accident       0.729211     1.000000
trunk_road_flag                                   0.792752    -1.000000

                                                       25%           50%  \
accident_year                                  2021.000000   2021.000000
location_easting_osgr                        392252.500000 461832.500000
location_northing_osgr                       175175.250000 211641.500000
longitude                                        -2.116694     -1.087226
latitude                                         51.462397     51.789613
police_force                                      4.000000     21.000000
```

accident_severity	3.000000	3.000000
number_of_vehicles	1.000000	2.000000
number_of_casualties	1.000000	1.000000
day_of_week	2.000000	4.000000
local_authority_district	-1.000000	-1.000000
first_road_class	3.000000	4.000000
first_road_number	0.000000	34.000000
road_type	6.000000	6.000000
speed_limit	30.000000	30.000000
junction_detail	0.000000	2.000000
junction_control	-1.000000	2.000000
second_road_class	0.000000	3.000000
second_road_number	-1.000000	0.000000
pedestrian_crossing_human_control	0.000000	0.000000
pedestrian_crossing_physical_facilities	0.000000	0.000000
light_conditions	1.000000	1.000000
weather_conditions	1.000000	1.000000
road_surface_conditions	1.000000	1.000000
special_conditions_at_site	0.000000	0.000000
carriageway_hazards	0.000000	0.000000
urban_or_rural_area	1.000000	1.000000
did_police_officer_attend_scene_of_accident	1.000000	1.000000
trunk_road_flag	2.000000	2.000000

	75%	max
accident_year	2021.000000	2.021000e+03
location_easting_osgr	530038.000000	6.551400e+05
location_northing_osgr	382389.500000	1.179892e+06
longitude	-0.127483	1.755955e+00
latitude	53.334320	6.050001e+01
police_force	44.000000	9.900000e+01
accident_severity	3.000000	3.000000e+00
number_of_vehicles	2.000000	1.300000e+01
number_of_casualties	1.000000	2.200000e+01
day_of_week	6.000000	7.000000e+00
local_authority_district	-1.000000	4.800000e+02
first_road_class	6.000000	6.000000e+00
first_road_number	532.000000	9.480000e+03
road_type	6.000000	9.000000e+00
speed_limit	40.000000	7.000000e+01
junction_detail	3.000000	9.900000e+01
junction_control	4.000000	9.000000e+00
second_road_class	6.000000	6.000000e+00
second_road_number	0.000000	9.176000e+03
pedestrian_crossing_human_control	0.000000	9.000000e+00
pedestrian_crossing_physical_facilities	0.000000	9.000000e+00
light_conditions	4.000000	7.000000e+00

```
weather_conditions                               1.000000  9.000000e+00
road_surface_conditions                          2.000000  9.000000e+00
special_conditions_at_site                       0.000000  9.000000e+00
carriageway_hazards                              0.000000  9.000000e+00
urban_or_rural_area                              2.000000  3.000000e+00
did_police_officer_attend_scene_of_accident      2.000000  3.000000e+00
trunk_road_flag                                  2.000000  2.000000e+00
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 101087 entries, 0 to 101086
Data columns (total 36 columns):
 #   Column                                       Non-Null Count   Dtype
---  ------                                       --------------   -----
 0   accident_index                               101087 non-null  object
 1   accident_year                                101087 non-null  int64
 2   accident_reference                           101087 non-null  object
 3   location_easting_osgr                        101070 non-null  float64
 4   location_northing_osgr                       101070 non-null  float64
 5   longitude                                    101070 non-null  float64
 6   latitude                                     101070 non-null  float64
 7   police_force                                 101087 non-null  int64
 8   accident_severity                            101087 non-null  int64
 9   number_of_vehicles                           101087 non-null  int64
 10  number_of_casualties                         101087 non-null  int64
 11  date                                         101087 non-null  object
 12  day_of_week                                  101087 non-null  int64
 13  time                                         101087 non-null  object
 14  local_authority_district                     101087 non-null  int64
 15  local_authority_ons_district                 101087 non-null  object
 16  local_authority_highway                      101087 non-null  object
 17  first_road_class                             101087 non-null  int64
 18  first_road_number                            101087 non-null  int64
 19  road_type                                    101087 non-null  int64
 20  speed_limit                                  101087 non-null  int64
 21  junction_detail                              101087 non-null  int64
 22  junction_control                             101087 non-null  int64
 23  second_road_class                            101087 non-null  int64
 24  second_road_number                           101087 non-null  int64
 25  pedestrian_crossing_human_control            101087 non-null  int64
 26  pedestrian_crossing_physical_facilities      101087 non-null  int64
 27  light_conditions                             101087 non-null  int64
 28  weather_conditions                           101087 non-null  int64
 29  road_surface_conditions                      101087 non-null  int64
 30  special_conditions_at_site                   101087 non-null  int64
 31  carriageway_hazards                          101087 non-null  int64
```

```
 32   urban_or_rural_area                           101087 non-null  int64
 33   did_police_officer_attend_scene_of_accident   101087 non-null  int64
 34   trunk_road_flag                               101087 non-null  int64
 35   lsoa_of_accident_location                     101087 non-null  object
dtypes: float64(4), int64(25), object(7)
memory usage: 27.8+ MB
```

[9]:
```python
# Checking for missing values in the data
data.isnull().sum()
```

[9]:
```
accident_index                                  0
accident_year                                   0
accident_reference                              0
location_easting_osgr                          17
location_northing_osgr                         17
longitude                                      17
latitude                                       17
police_force                                    0
accident_severity                               0
number_of_vehicles                              0
number_of_casualties                            0
date                                            0
day_of_week                                     0
time                                            0
local_authority_district                        0
local_authority_ons_district                    0
local_authority_highway                         0
first_road_class                                0
first_road_number                               0
road_type                                       0
speed_limit                                     0
junction_detail                                 0
junction_control                                0
second_road_class                               0
second_road_number                              0
pedestrian_crossing_human_control               0
pedestrian_crossing_physical_facilities         0
light_conditions                                0
weather_conditions                              0
road_surface_conditions                         0
special_conditions_at_site                      0
carriageway_hazards                             0
urban_or_rural_area                             0
did_police_officer_attend_scene_of_accident     0
trunk_road_flag                                 0
lsoa_of_accident_location                       0
dtype: int64
```

[31]:
```python
data.nunique()
```

[31]:
```
police_force                                     44
accident_severity                                 3
number_of_vehicles                               13
number_of_casualties                             12
day_of_week                                       7
local_authority_district                         15
first_road_class                                  6
first_road_number                              3099
road_type                                         6
speed_limit                                       6
junction_detail                                  11
junction_control                                  6
second_road_class                                 8
second_road_number                             2374
pedestrian_crossing_human_control                 5
pedestrian_crossing_physical_facilities           8
light_conditions                                  6
weather_conditions                               10
road_surface_conditions                           7
special_conditions_at_site                       10
carriageway_hazards                               8
urban_or_rural_area                               3
did_police_officer_attend_scene_of_accident       3
trunk_road_flag                                   3
dtype: int64
```

[12]:
```python
# deleting columns which are mostly unqiue or has a unique category
data = data.
 ↪drop(['accident_index','accident_year','accident_reference','location_easting_osgr',
                ␣
 ↪'location_northing_osgr','longitude','latitude','date','time','lsoa_of_accident_location',
                 'local_authority_ons_district','local_authority_highway'],␣
 ↪axis = 1)
```

[17]:
```python
data.duplicated().sum()
```

[17]: 0

[16]:
```python
# dropping duplicaates
data = data.drop_duplicates()
```

[21]:
```python
# converting the datatypes of categoraical variables to category
data['police_force'] = data['police_force'].astype('category')
data['day_of_week'] = data['day_of_week'].astype('category')
data['accident_severity'] = data['accident_severity'].astype('category')
```

```
data['first_road_class'] = data['first_road_class'].astype('category')
data['local_authority_district'] = data['local_authority_district'].
    ↪astype('category')
data['second_road_class'] = data['second_road_class'].astype('category')

data['pedestrian_crossing_human_control'] =␣
    ↪data['pedestrian_crossing_human_control'].astype('category')
data['pedestrian_crossing_physical_facilities'] =␣
    ↪data['pedestrian_crossing_physical_facilities'].astype('category')
data['light_conditions'] = data['light_conditions'].astype('category')
data['weather_conditions'] = data['weather_conditions'].astype('category')
data['special_conditions_at_site'] = data['special_conditions_at_site'].
    ↪astype('category')
data['carriageway_hazards'] = data['carriageway_hazards'].astype('category')
data['did_police_officer_attend_scene_of_accident'] =␣
    ↪data['did_police_officer_attend_scene_of_accident'].astype('category')

data['trunk_road_flag'] = data['trunk_road_flag'].astype('category')
data['road_surface_conditions'] = data['road_surface_conditions'].
    ↪astype('category')
data['urban_or_rural_area'] = data['urban_or_rural_area'].astype('category')
```

[22]:
```
num_cols =␣
    ↪['number_of_vehicles','number_of_casualties','first_road_number','speed_limit',␣
    ↪'junction_detail', 'junction_control']
cat_cols =␣
    ↪['police_force','day_of_week','accident_severity','first_road_class','local_authority_distr
            ␣
    ↪'second_road_class','pedestrian_crossing_human_control','pedestrian_crossing_physical_facil
            ␣
    ↪'weather_conditions','special_conditions_at_site','carriageway_hazards','did_police_officer
             'road_surface_conditions','trunk_road_flag','urban_or_rural_area']
```

[24]:
```
# let's check for missing values in the data
round(data.isnull().sum() / data.isnull().count() * 100, 2)
```

[24]:
```
police_force                                0.0
accident_severity                           0.0
number_of_vehicles                          0.0
number_of_casualties                        0.0
day_of_week                                 0.0
local_authority_district                    0.0
first_road_class                            0.0
first_road_number                           0.0
road_type                                   0.0
speed_limit                                 0.0
```

```
junction_detail                              0.0
junction_control                             0.0
second_road_class                            0.0
second_road_number                           0.0
pedestrian_crossing_human_control            0.0
pedestrian_crossing_physical_facilities      0.0
light_conditions                             0.0
weather_conditions                           0.0
road_surface_conditions                      0.0
special_conditions_at_site                   0.0
carriageway_hazards                          0.0
urban_or_rural_area                          0.0
did_police_officer_attend_scene_of_accident  0.0
trunk_road_flag                              0.0
dtype: float64
```

[25]:
```
# checking for value_counts in each nominal feature
columns = cat_cols
for i in columns:
    print("Unique values in", i, "are :")
    print(data[i].value_counts())
    print("*" * 50)
```

```
Unique values in police_force are :
1     21738
20     4094
99     3749
13     3532
46     3423
47     3006
6      2815
44     2808
43     2770
42     2601
4      2592
50     2570
45     2387
52     2168
14     1942
5      1804
31     1750
30     1706
10     1706
16     1683
7      1546
22     1525
41     1517
```

```
32     1490
36     1315
35     1282
12     1251
55     1232
54     1223
33     1086
40     1038
37     1017
34     1013
63      947
62      902
23      832
21      806
60      766
53      737
3       724
11      623
17      602
61      590
48      145
Name: police_force, dtype: int64
**************************************************
Unique values in day_of_week are :
6    15945
5    14149
4    13974
3    13680
7    13256
2    13117
1    10932
Name: day_of_week, dtype: int64
**************************************************
Unique values in accident_severity are :
3    73048
2    20533
1     1472
Name: accident_severity, dtype: int64
**************************************************
Unique values in first_road_class are :
3    44706
6    30379
4    12580
5     4123
1     2988
2      277
Name: first_road_class, dtype: int64
**************************************************
```

```
Unique values in local_authority_district are :
-1     94859
241       32
245       31
243       27
240       25
472       14
476       14
471       10
480        9
470        7
474        7
473        6
477        5
475        4
478        3
Name: local_authority_district, dtype: int64
**************************************************
Unique values in second_road_class are :
6     38843
0     38307
3     10774
4      3918
5      2767
1       337
-1       84
2        23
Name: second_road_class, dtype: int64
**************************************************
Unique values in pedestrian_crossing_human_control are :
0     89432
9      3720
2      1217
1       437
-1      247
Name: pedestrian_crossing_human_control, dtype: int64
**************************************************
Unique values in pedestrian_crossing_physical_facilities are :
0     70749
5      8022
4      5679
1      4459
9      3140
8      2508
7       264
-1      232
Name: pedestrian_crossing_physical_facilities, dtype: int64
**************************************************
```

```
Unique values in light_conditions are :
1     68118
4     19401
6      4840
7      1972
5       711
-1       11
Name: light_conditions, dtype: int64
**************************************************
Unique values in weather_conditions are :
1     75824
2     10290
8      3203
9      2698
5      1033
4       872
3       603
7       427
6        91
-1       12
Name: weather_conditions, dtype: int64
**************************************************
Unique values in special_conditions_at_site are :
0     90447
9      1742
4      1406
-1      351
1       318
5       248
7       203
3       157
6       130
2        51
Name: special_conditions_at_site, dtype: int64
**************************************************
Unique values in carriageway_hazards are :
0     91075
9      1455
2      1184
7       348
-1      342
1       262
6       202
3       185
Name: carriageway_hazards, dtype: int64
**************************************************
Unique values in did_police_officer_attend_scene_of_accident are :
1    65704
```

```
2    15612
3    13737
Name: did_police_officer_attend_scene_of_accident, dtype: int64
**************************************************
Unique values in road_surface_conditions are :
1     69488
2     22327
4      1202
9      1001
3       473
-1      428
5       134
Name: road_surface_conditions, dtype: int64
**************************************************
Unique values in trunk_road_flag are :
2     81736
-1     6954
1      6363
Name: trunk_road_flag, dtype: int64
**************************************************
Unique values in urban_or_rural_area are :
1    63633
2    31403
3       17
Name: urban_or_rural_area, dtype: int64
**************************************************
```

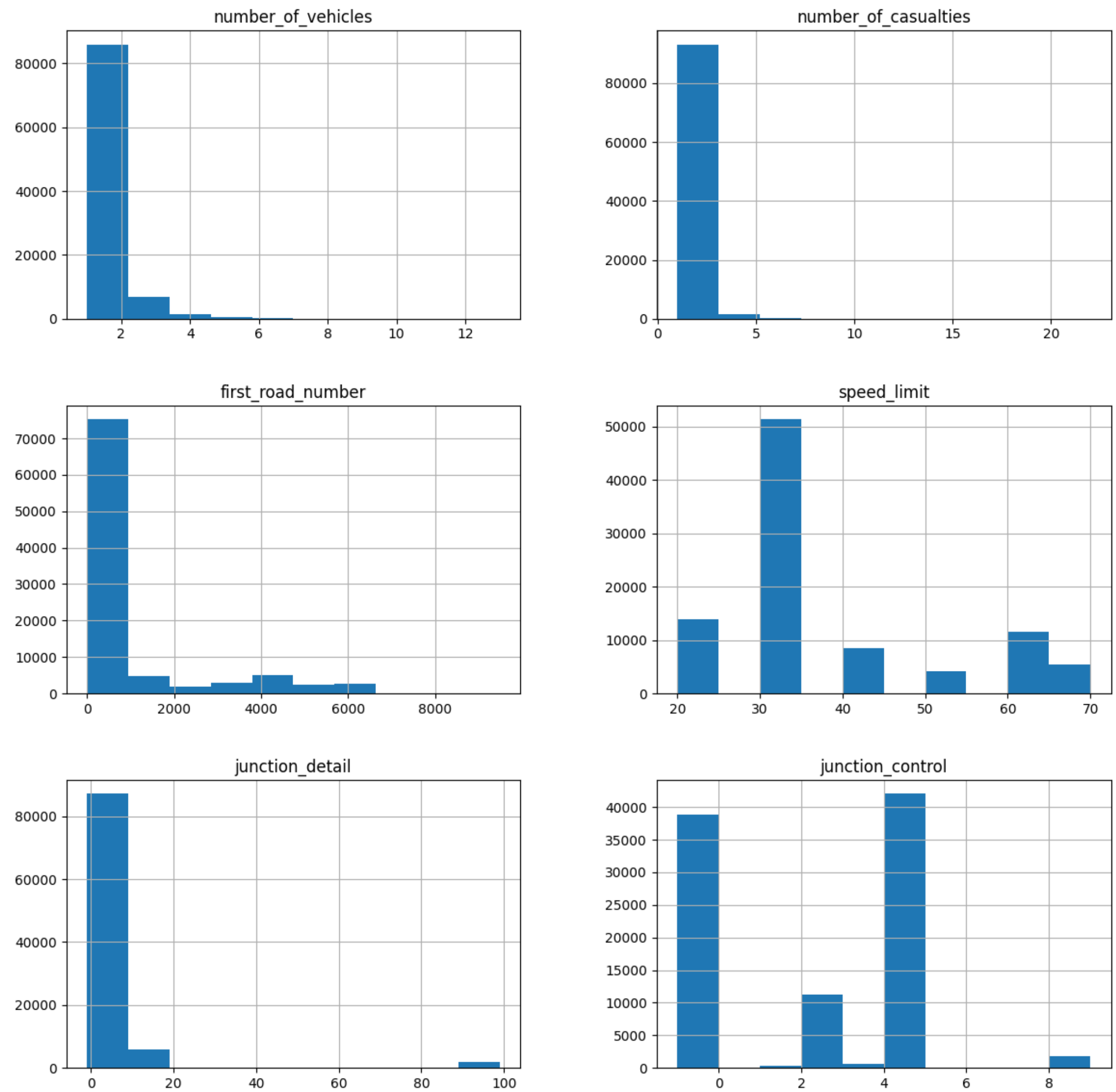
[26]:

```
# Checking summary statistics
data[num_cols].describe().T
```

[26]:

```
                       count        mean          std   min   25%   50%  \
number_of_vehicles   95053.0    1.850462     0.690398   1.0   1.0   2.0
number_of_casualties 95053.0    1.282737     0.690961   1.0   1.0   1.0
first_road_number    95053.0  836.641505  1617.922890   0.0   0.0  46.0
speed_limit          95053.0   36.287334    14.262362  20.0  30.0  30.0
junction_detail      95053.0    4.464825    14.038640  -1.0   0.0   2.0
junction_control     95053.0    1.794167     2.519444  -1.0  -1.0   2.0

                       75%     max
number_of_vehicles     2.0    13.0
number_of_casualties   1.0    22.0
first_road_number    595.0  9480.0
speed_limit           40.0    70.0
junction_detail        3.0    99.0
junction_control       4.0     9.0
```

[28]:
```python
# Creating histograms
data[num_cols].hist(figsize = (14, 14))

plt.show()
```



[ ]:

[ ]:

[30]:
```python
data.head()
```

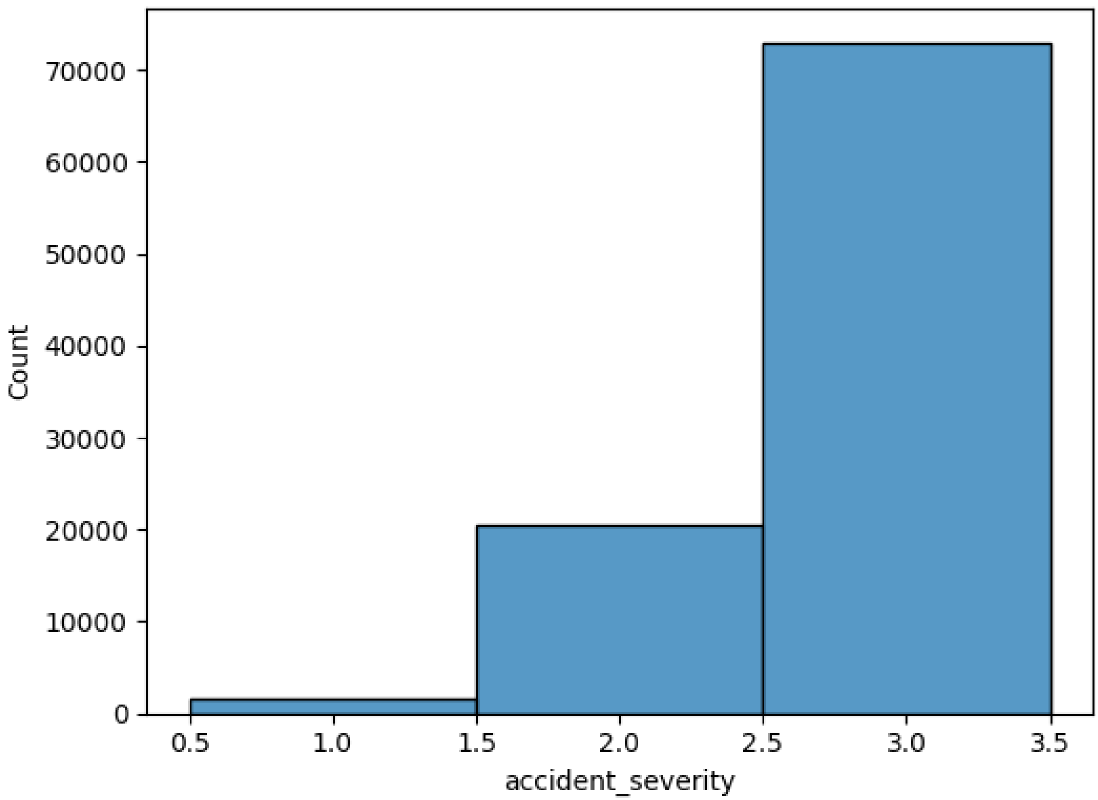
[30]:
```
   police_force accident_severity  number_of_vehicles  number_of_casualties  \
0             1                 3                   3                     1
```

```
1          1              2                2                    3
2          1              2                2                    4
3          1              1                1                    1
4          1              3                4                    1

  day_of_week local_authority_district first_road_class  first_road_number  \
0           6                       -1                6                  0
1           6                       -1                3               1203
2           6                       -1                4                272
3           6                       -1                3               3220
4           6                       -1                5                  0

   road_type  speed_limit  junction_detail  junction_control  \
0          6           30                9                 4
1          3           30                7                 2
2          6           30                9                 2
3          2           30                9                 4
4          6           20                3                 4

  second_road_class  second_road_number pedestrian_crossing_human_control  \
0                 6                   0                                 0
1                 3                1204                                 0
2                 5                   0                                 0
3                 6                   0                                 0
4                 6                   0                                 0

  pedestrian_crossing_physical_facilities light_conditions weather_conditions  \
0                                       0                4                  7
1                                       5                4                  1
2                                       5                4                  1
3                                       4                4                  1
4                                       0                4                  1

  road_surface_conditions special_conditions_at_site carriageway_hazards  \
0                       4                          1                   0
1                       1                          0                   0
2                       1                          0                   0
3                       1                          0                   0
4                       1                          0                   0

  urban_or_rural_area did_police_officer_attend_scene_of_accident  \
0                   1                                           1
1                   1                                           1
2                   1                                           1
3                   1                                           1
4                   1                                           1
```

```
   trunk_road_flag
0                2
1                2
2                2
3                2
4                2
```

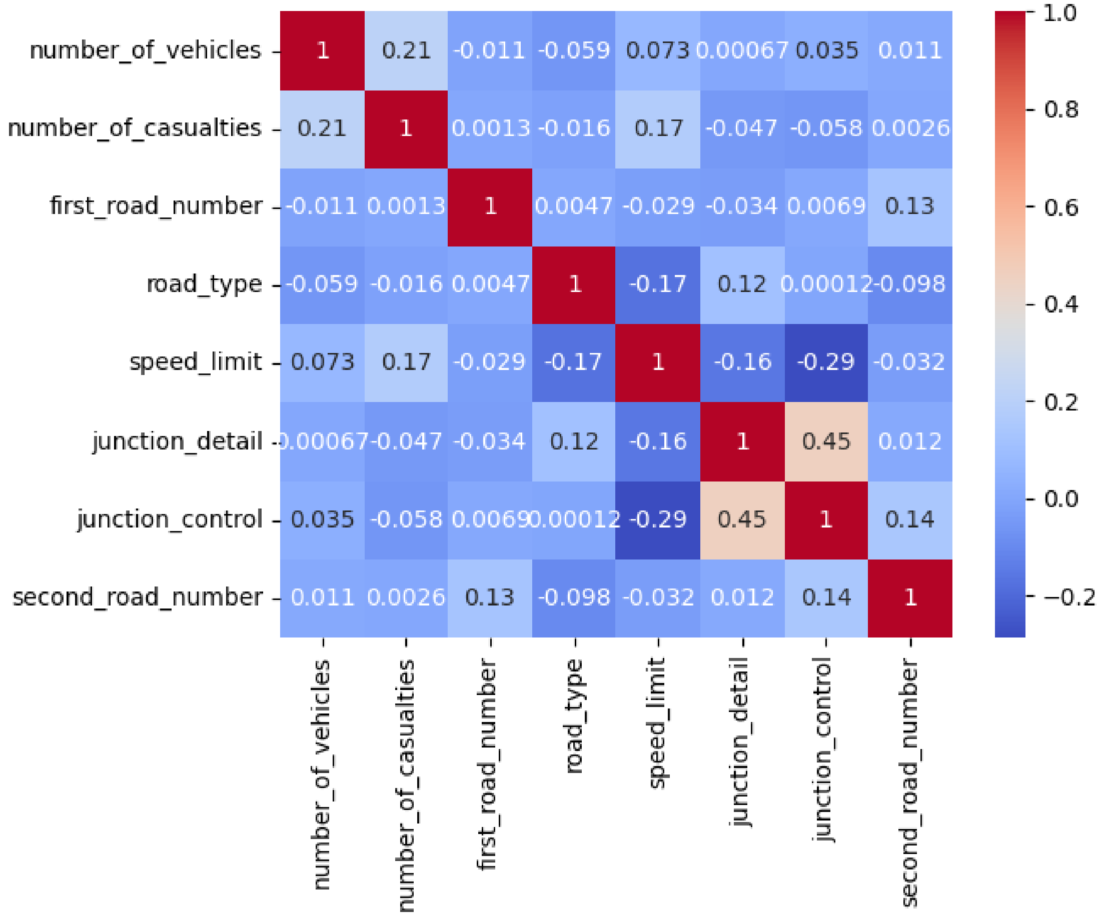
[33]:

```
# Distribution of target variable
sns.histplot(data["accident_severity"], kde=False)
plt.show()
```

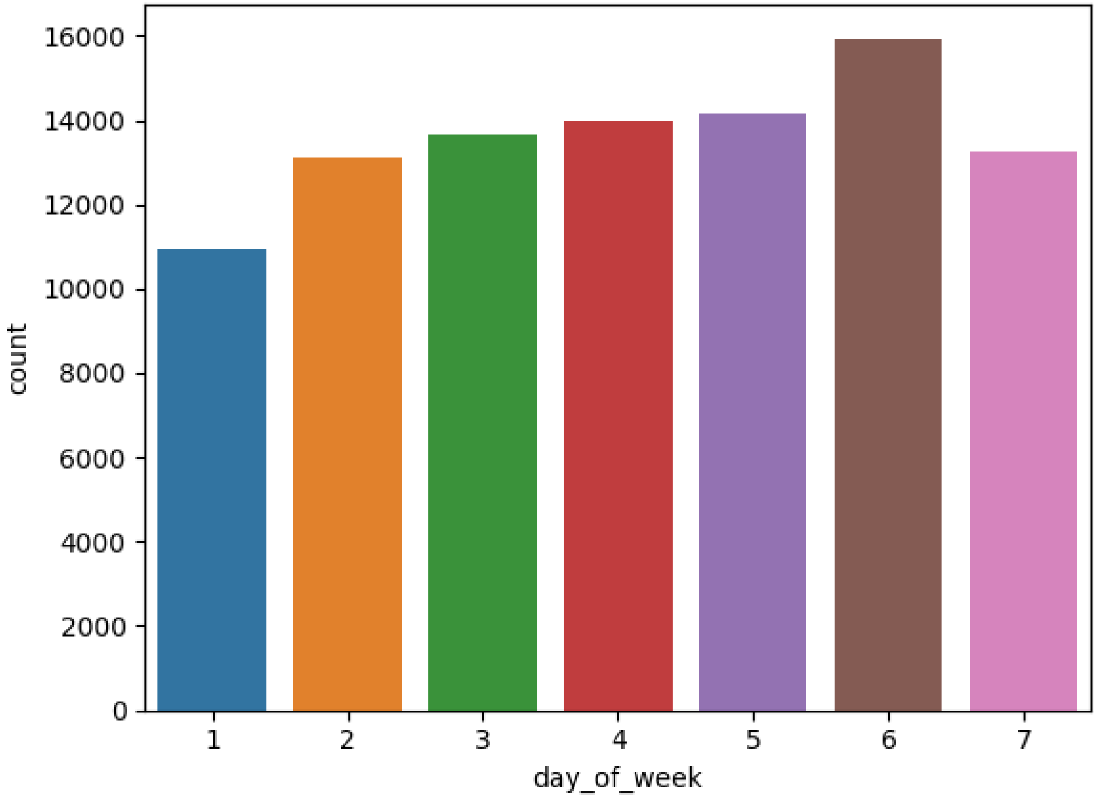


[35]:

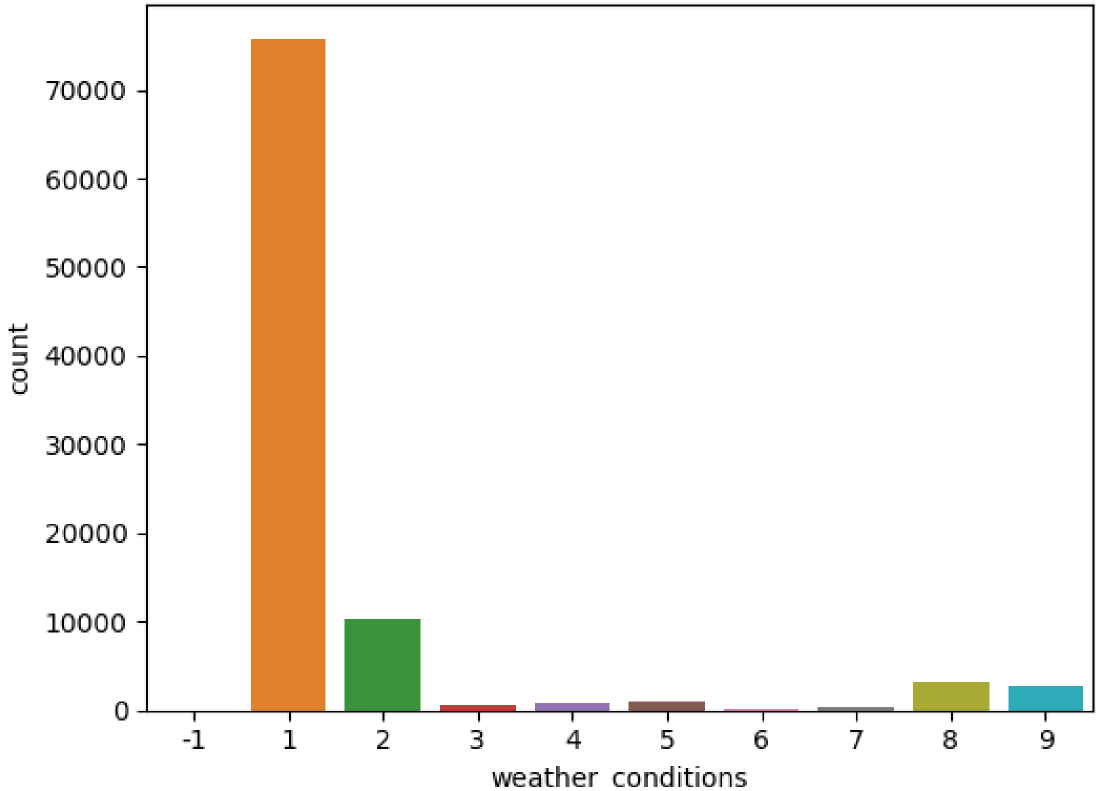
```
# Correlation analysis
corr = data.corr()
sns.heatmap(corr, cmap="coolwarm", annot=True)
plt.show()
```



[36]:
```
# Bar chart for categorical variables
sns.countplot(x="day_of_week", data=data)
plt.show()

sns.countplot(x="weather_conditions", data=data)
plt.show()
```



16000
14000
12000
10000
8000
6000
4000
2000
0
count
1
2
3
4
5
6
7
day_of_week



[38]:
```
for i in cat_cols:
    print(data[i].value_counts(normalize = True))

    print('*' * 40)
```

```
1     0.228693
20    0.043071
99    0.039441
13    0.037158
46    0.036011
47    0.031624
6     0.029615
44    0.029541
43    0.029142
42    0.027364
4     0.027269
50    0.027038
45    0.025112
52    0.022808
14    0.020431
5     0.018979
```

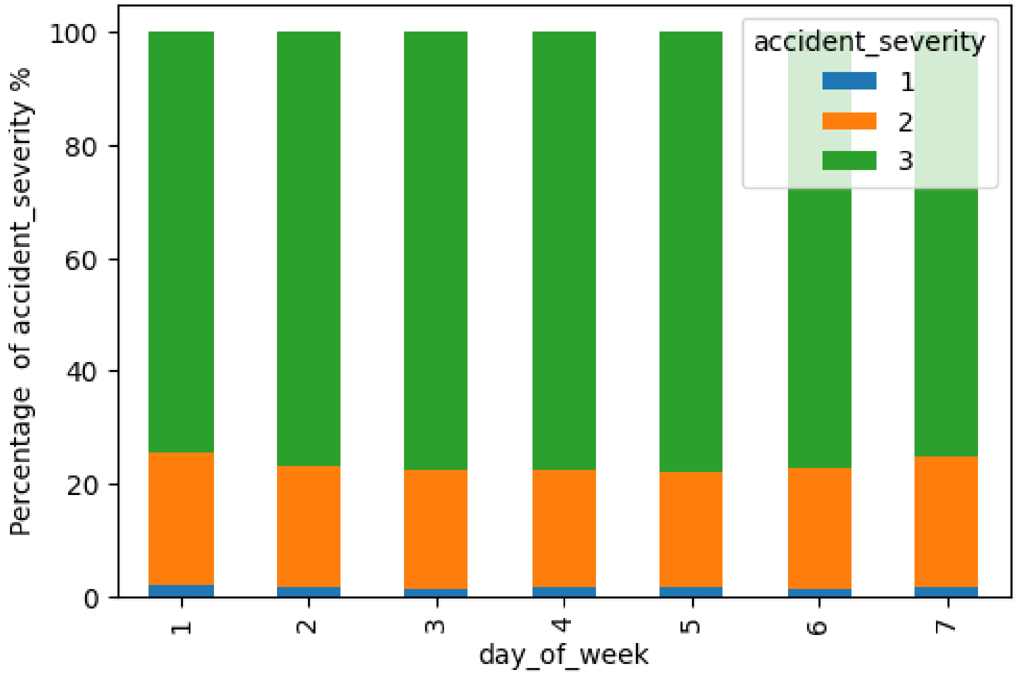
```
31    0.018411
30    0.017948
10    0.017948
16    0.017706
7     0.016265
22    0.016044
41    0.015960
32    0.015675
36    0.013834
35    0.013487
12    0.013161
55    0.012961
54    0.012867
33    0.011425
40    0.010920
37    0.010699
34    0.010657
63    0.009963
62    0.009489
23    0.008753
21    0.008479
60    0.008059
53    0.007754
3     0.007617
11    0.006554
17    0.006333
61    0.006207
48    0.001525
Name: police_force, dtype: float64
****************************************
6    0.167749
5    0.148854
4    0.147013
3    0.143920
7    0.139459
2    0.137997
1    0.115010
Name: day_of_week, dtype: float64
****************************************
3    0.768498
2    0.216016
1    0.015486
Name: accident_severity, dtype: float64
****************************************
3    0.470327
6    0.319601
4    0.132347
5    0.043376
```

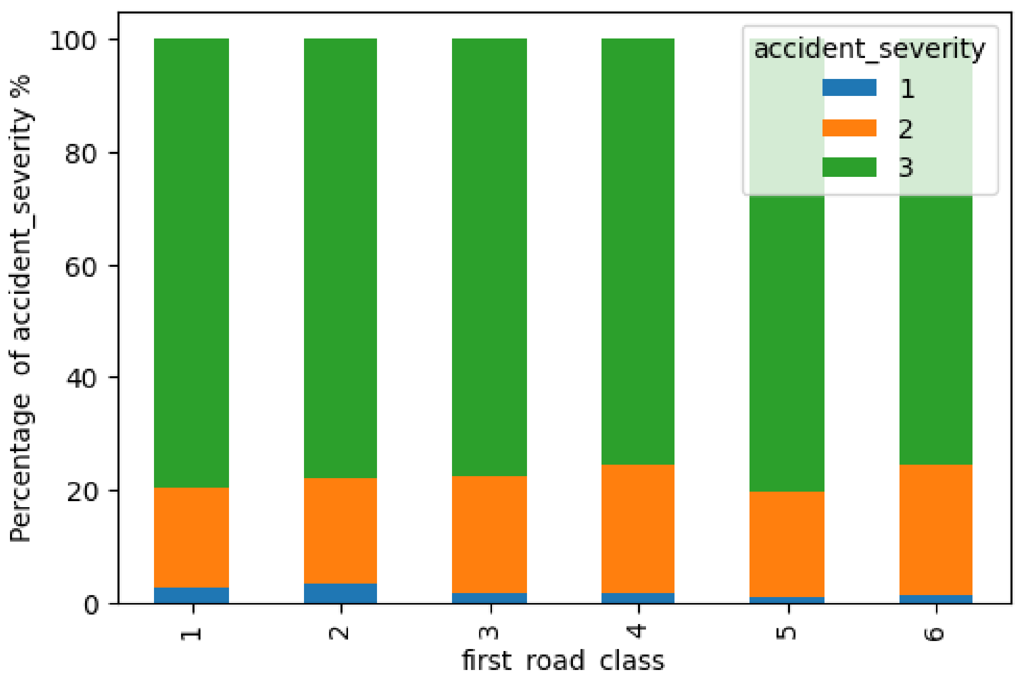
```
1    0.031435
2    0.002914
Name: first_road_class, dtype: float64
****************************************
-1     0.997959
241    0.000337
245    0.000326
243    0.000284
240    0.000263
472    0.000147
476    0.000147
471    0.000105
480    0.000095
470    0.000074
474    0.000074
473    0.000063
477    0.000053
475    0.000042
478    0.000032
Name: local_authority_district, dtype: float64
****************************************
6     0.408646
0     0.403007
3     0.113347
4     0.041219
5     0.029110
1     0.003545
-1    0.000884
2     0.000242
Name: second_road_class, dtype: float64
****************************************
0     0.940865
9     0.039136
2     0.012803
1     0.004597
-1    0.002599
Name: pedestrian_crossing_human_control, dtype: float64
****************************************
0     0.744311
5     0.084395
4     0.059746
1     0.046911
9     0.033034
8     0.026385
7     0.002777
-1    0.002441
Name: pedestrian_crossing_physical_facilities, dtype: float64
****************************************
```

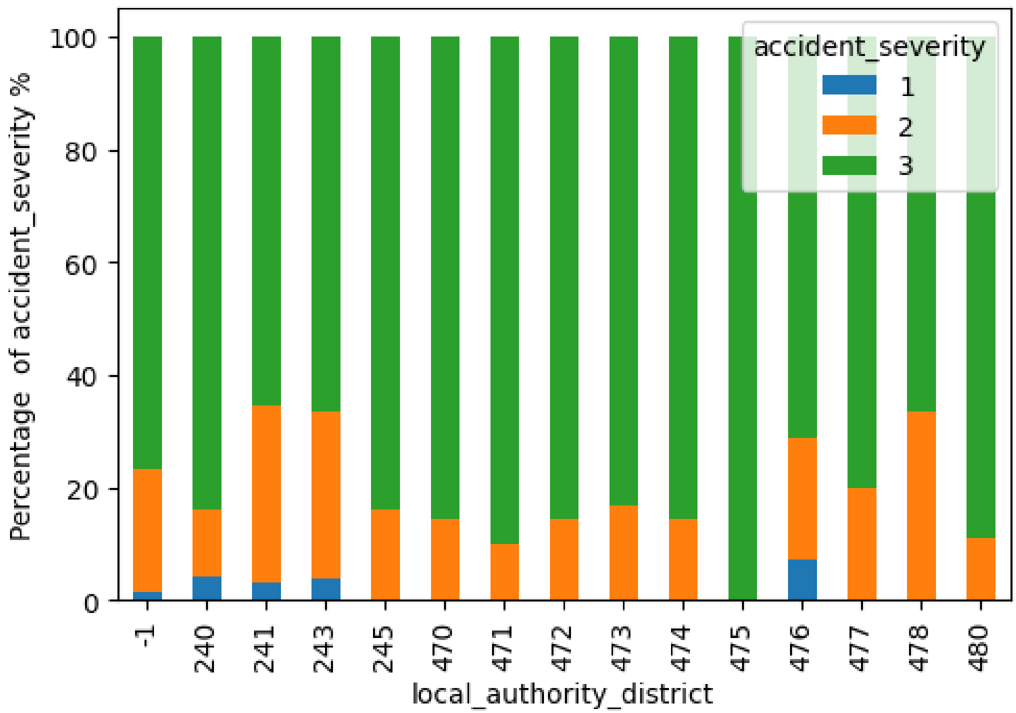
```
1     0.716632
4     0.204107
6     0.050919
7     0.020746
5     0.007480
-1    0.000116
Name: light_conditions, dtype: float64
****************************************
1     0.797702
2     0.108255
8     0.033697
9     0.028384
5     0.010868
4     0.009174
3     0.006344
7     0.004492
6     0.000957
-1    0.000126
Name: weather_conditions, dtype: float64
****************************************
0     0.951543
9     0.018327
4     0.014792
-1    0.003693
1     0.003346
5     0.002609
7     0.002136
3     0.001652
6     0.001368
2     0.000537
Name: special_conditions_at_site, dtype: float64
****************************************
0     0.958150
9     0.015307
2     0.012456
7     0.003661
-1    0.003598
1     0.002756
6     0.002125
3     0.001946
Name: carriageway_hazards, dtype: float64
****************************************
1    0.691235
2    0.164245
3    0.144519
Name: did_police_officer_attend_scene_of_accident, dtype: float64
****************************************
1     0.731045
```

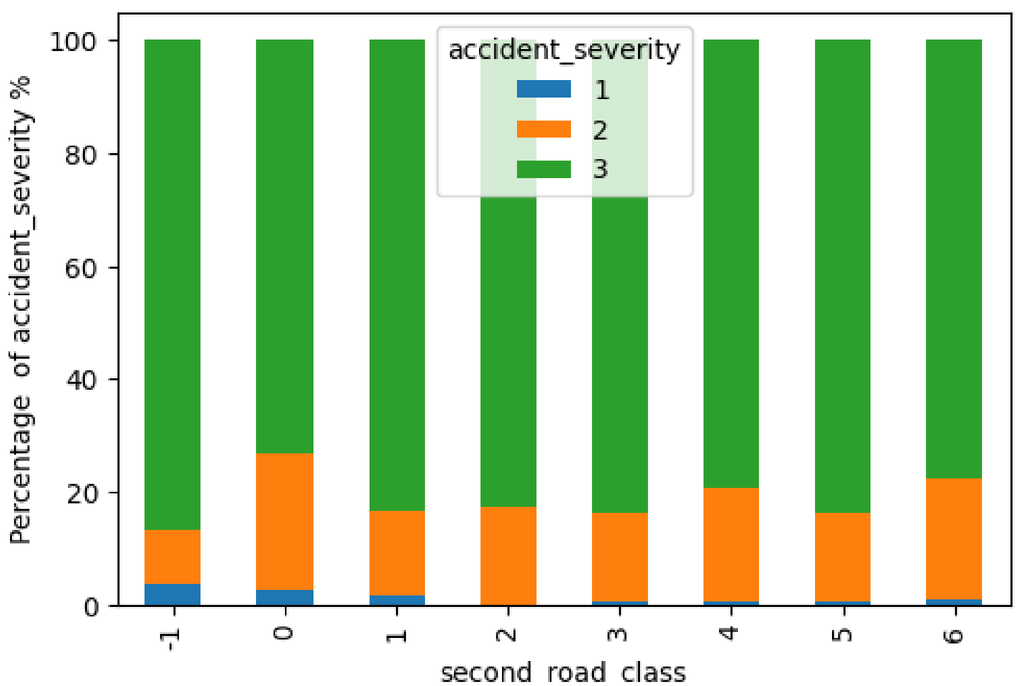
```
2      0.234890
4      0.012646
9      0.010531
3      0.004976
-1     0.004503
5      0.001410
Name: road_surface_conditions, dtype: float64
****************************************
2      0.859899
-1     0.073159
1      0.066942
Name: trunk_road_flag, dtype: float64
****************************************
1    0.669448
2    0.330374
3    0.000179
Name: urban_or_rural_area, dtype: float64
****************************************
```

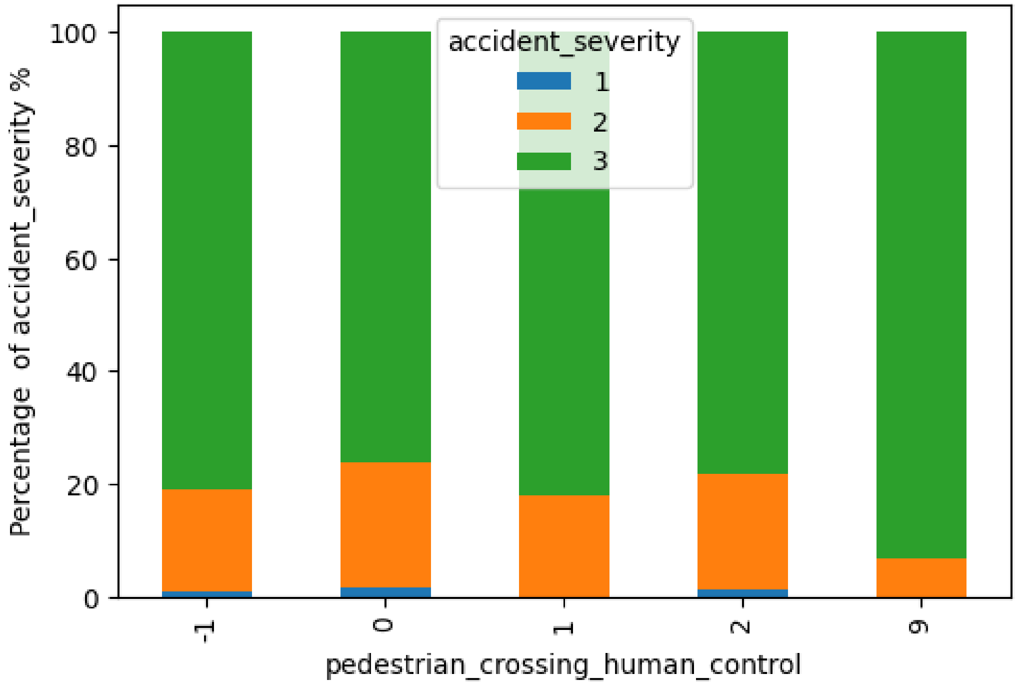
[39]:
```
for i in cat_cols:
    if i != 'accident_severity':
        (pd.crosstab(data[i], data['accident_severity'], normalize =␣
 ↪'index')*100).plot(kind = 'bar', figsize = (6, 4), stacked = True)
        plt.ylabel('Percentage  of accident_severity %')
```

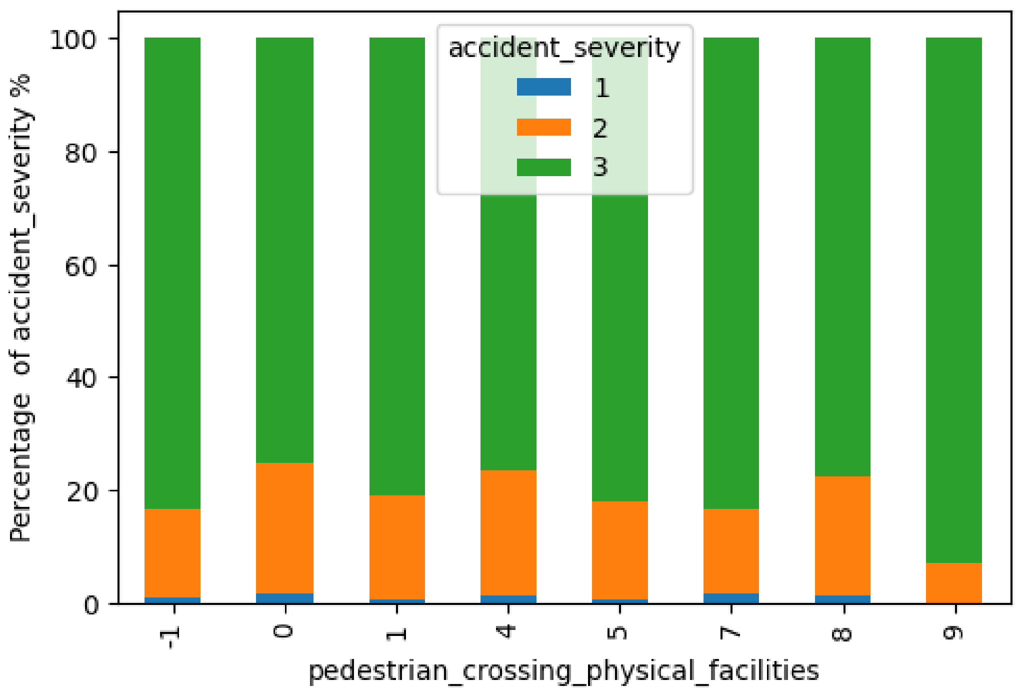

accident_severity
1
2
3
Percentage of accident_severity %
100
80
60
40
20
0
1
2
3
4
5
6
7
day_of_week

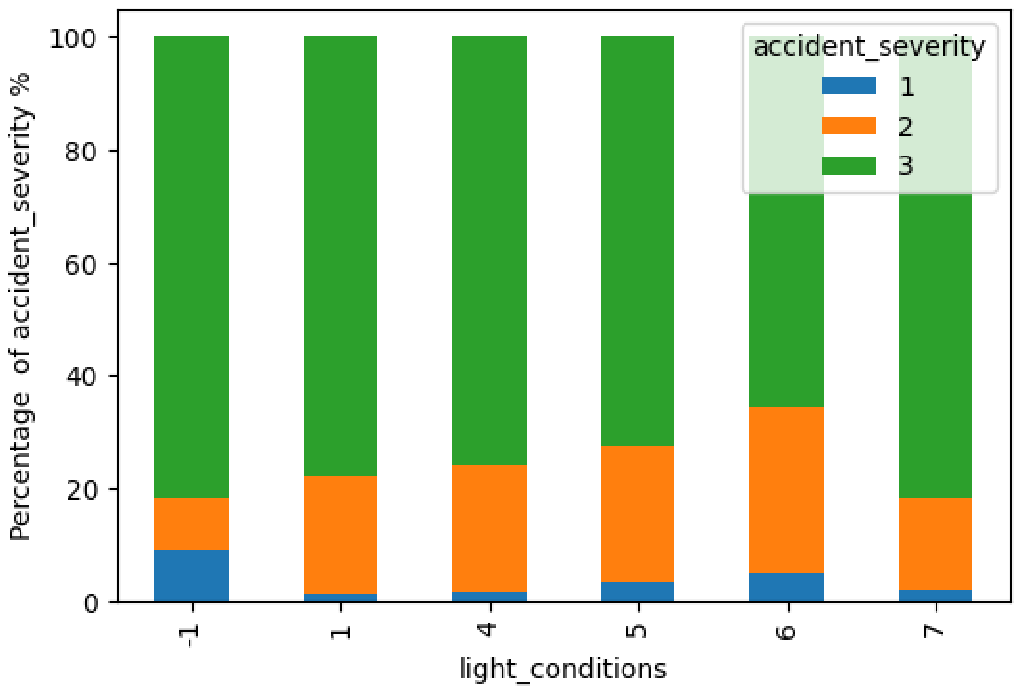

accident_severity
1
2
3
Percentage of accident_severity %
100
80
60
40
20
0
1
2
3
4
5
6
first_road_class


accident_severity
1
2
3
Percentage of accident_severity %
0
20
40
60
80
100
-1
240
241
243
245
470
471
472
473
474
475
476
477
478
480
local_authority_district

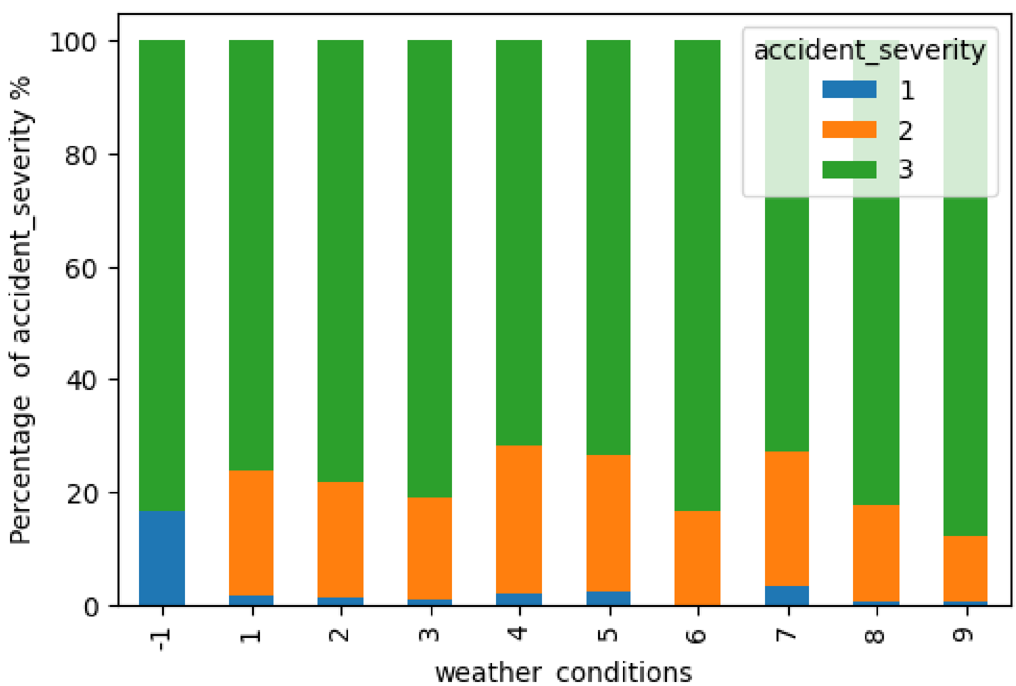

accident_severity
1
2
3
Percentage of accident_severity %
0
20
40
60
80
100
-1
0
1
2
3
4
5
6
second_road_class


accident_severity
1
2
3
Percentage of accident_severity %
0
20
40
60
80
100
-1
0
1
2
9
pedestrian_crossing_human_control

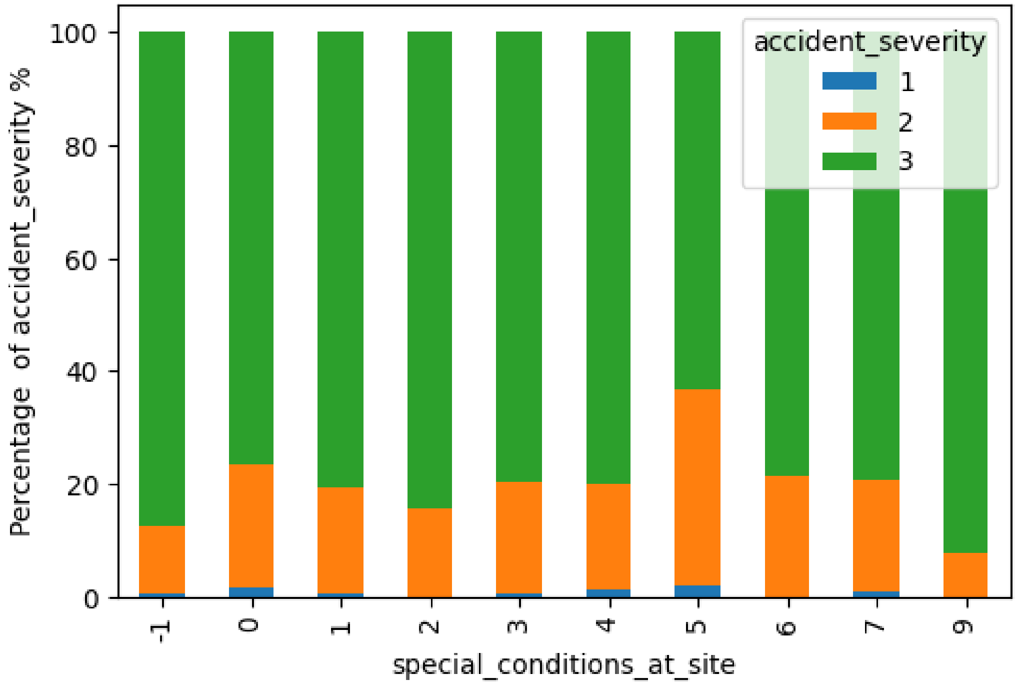

accident_severity
1
2
3
Percentage of accident_severity %
0
20
40
60
80
100
-1
0
1
4
5
7
8
9
pedestrian_crossing_physical_facilities



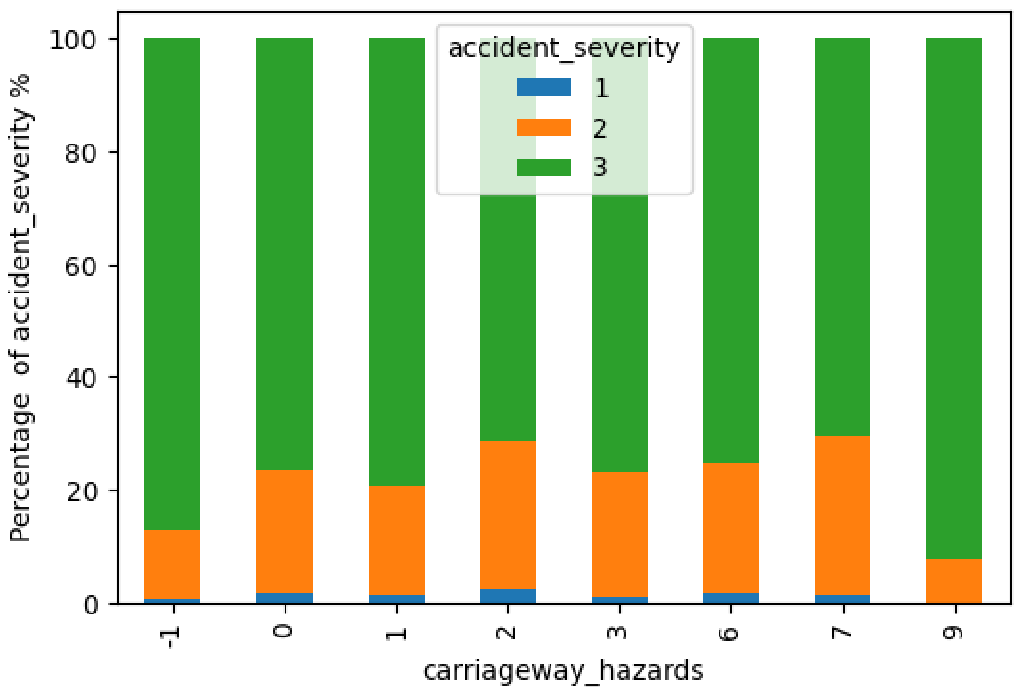
Percentage of accident_severity %
100
80
60
40
20
0
-1
1
4
5
6
7
light_conditions
accident_severity
1
2
3



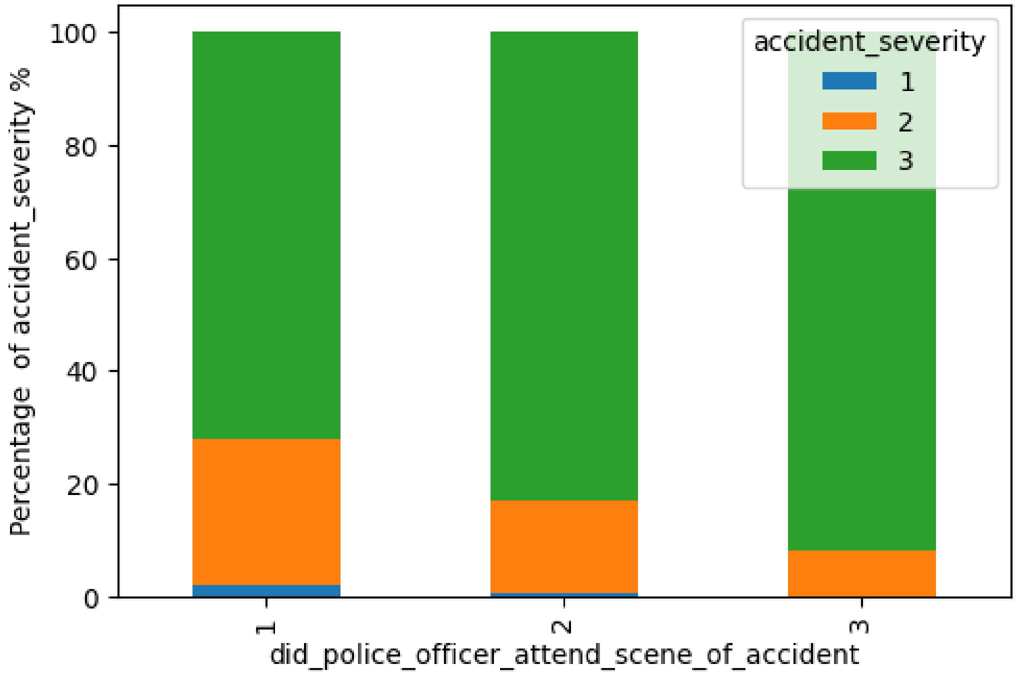
Percentage of accident_severity %
100
80
60
40
20
0
-1
1
2
3
4
5
6
7
8
9
weather_conditions
accident_severity
1
2
3

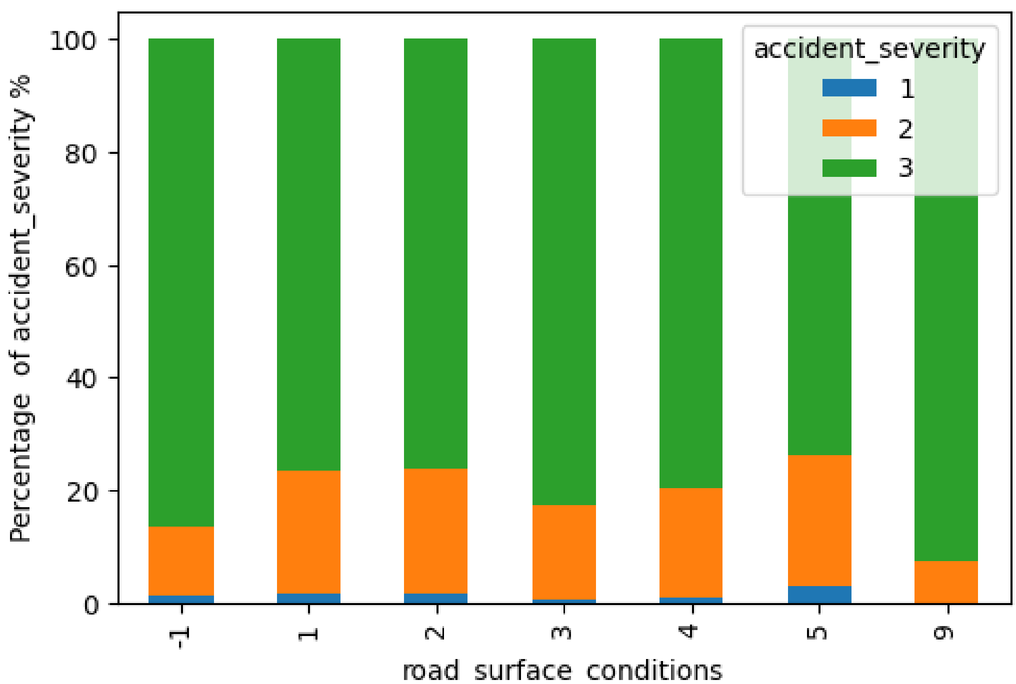


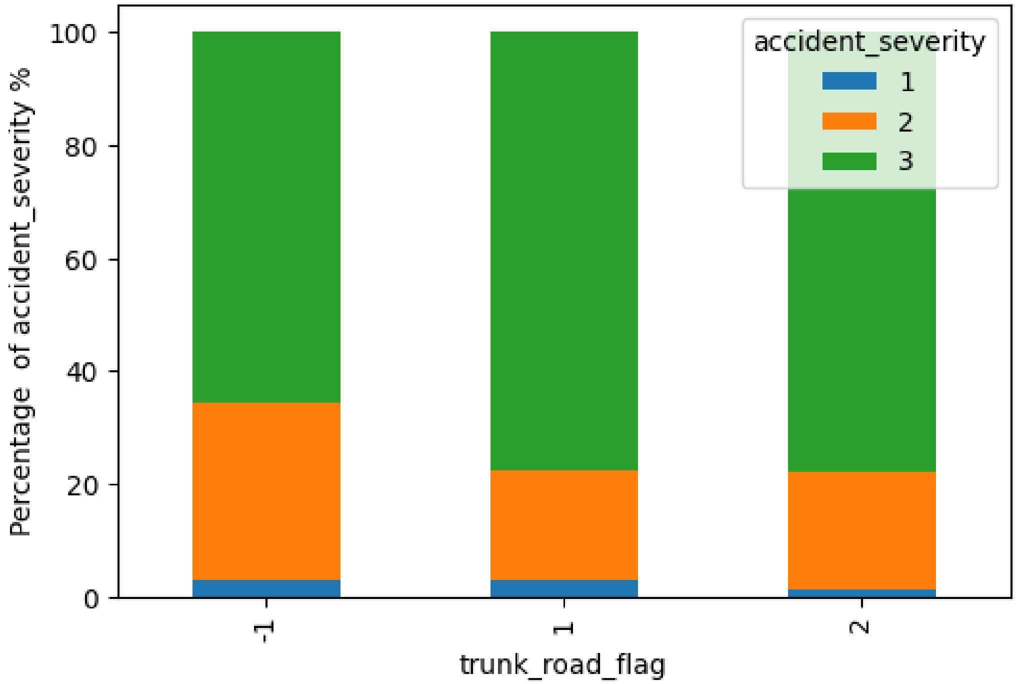
accident_severity
1
2
3
Percentage of accident_severity %
special_conditions_at_site

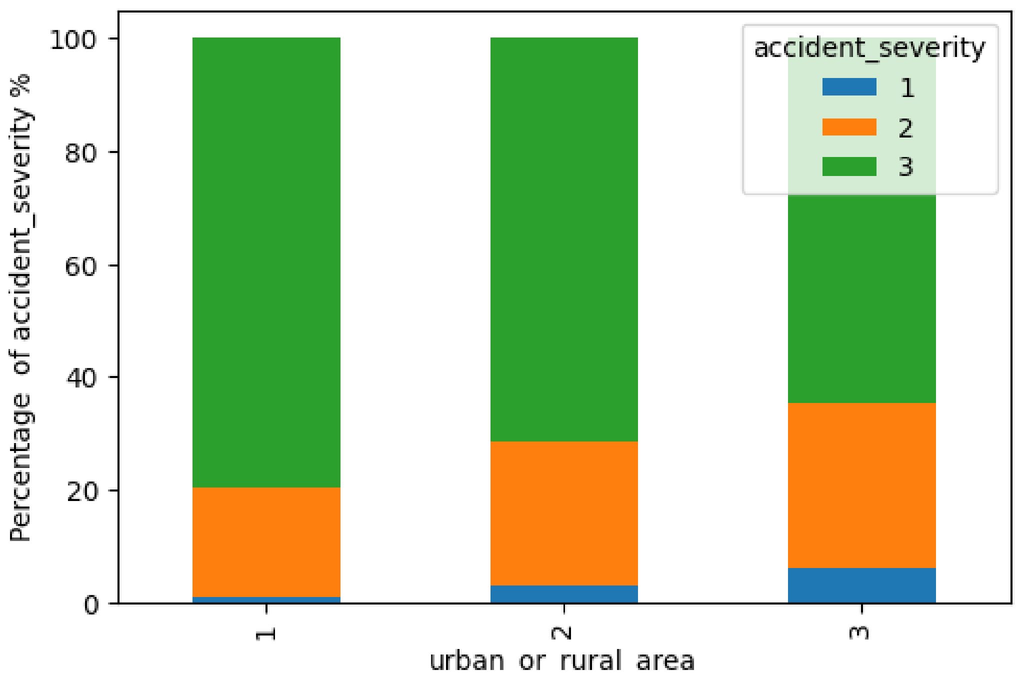


accident_severity
1
2
3
Percentage of accident_severity %
carriageway_hazards


accident_severity
1
2
3
Percentage of accident_severity %
0
20
40
60
80
100
1
2
3
did_police_officer_attend_scene_of_accident


accident_severity
1
2
3
Percentage of accident_severity %
0
20
40
60
80
100
-1
1
2
3
4
5
9
road_surface_conditions


accident_severity
1
2
3
Percentage of accident_severity %
100
80
60
40
20
0
-1
1
2
trunk_road_flag


accident_severity
1
2
3
Percentage of accident_severity %
100
80
60
40
20
0
1
2
3
urban_or_rural_area

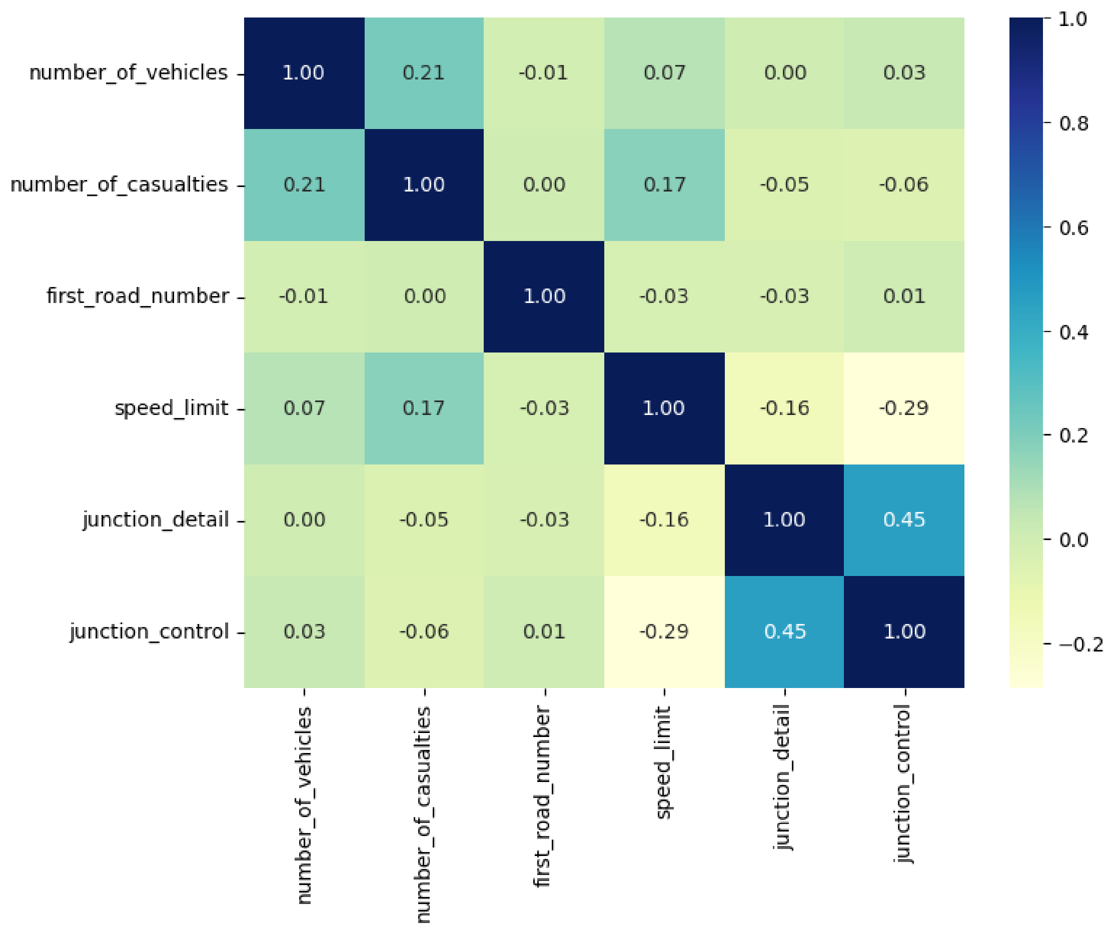
[ ]:

[41]:
```
data[num_cols].corr()
```

[41]:
```
                      number_of_vehicles  number_of_casualties  \
number_of_vehicles              1.000000              0.212728
number_of_casualties            0.212728              1.000000
first_road_number              -0.010713              0.001285
speed_limit                     0.073207              0.173840
junction_detail                 0.000665             -0.046740
junction_control                0.034882             -0.057680

                      first_road_number  speed_limit  junction_detail  \
number_of_vehicles            -0.010713     0.073207         0.000665
number_of_casualties           0.001285     0.173840        -0.046740
first_road_number              1.000000    -0.029029        -0.033807
speed_limit                   -0.029029     1.000000        -0.155483
junction_detail               -0.033807    -0.155483         1.000000
junction_control               0.006942    -0.285929         0.454381

                      junction_control
number_of_vehicles            0.034882
number_of_casualties         -0.057680
first_road_number             0.006942
speed_limit                  -0.285929
junction_detail               0.454381
junction_control              1.000000
```

[40]:
```
# Plotting the correlation between numerical variables
plt.figure(figsize = (8, 6))

sns.heatmap(data[num_cols].corr(), annot = True, fmt = '0.2f', cmap = 'YlGnBu')
```

[40]: <Axes: >

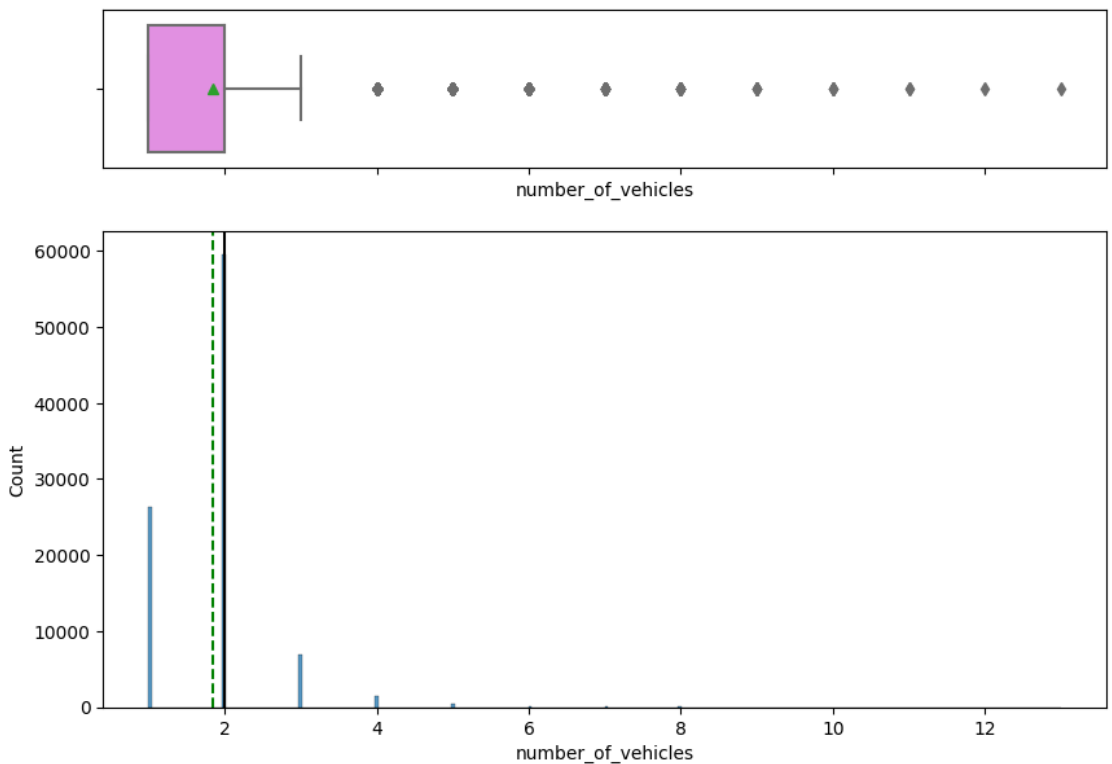


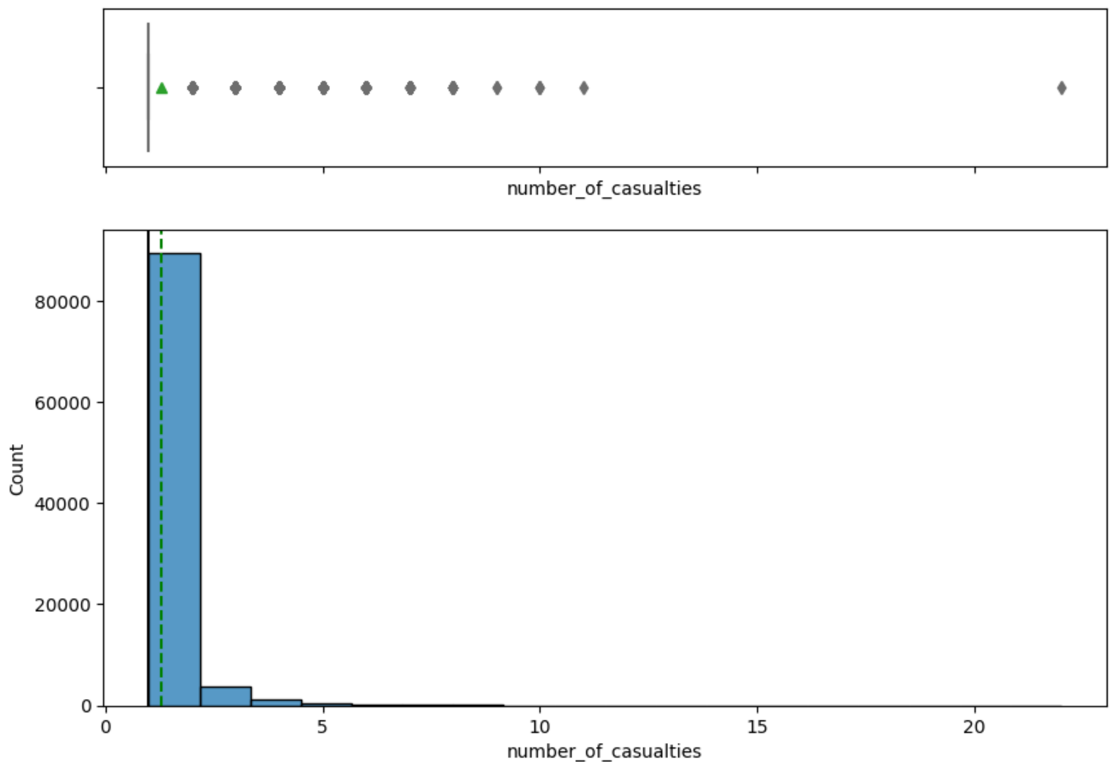
[43]:

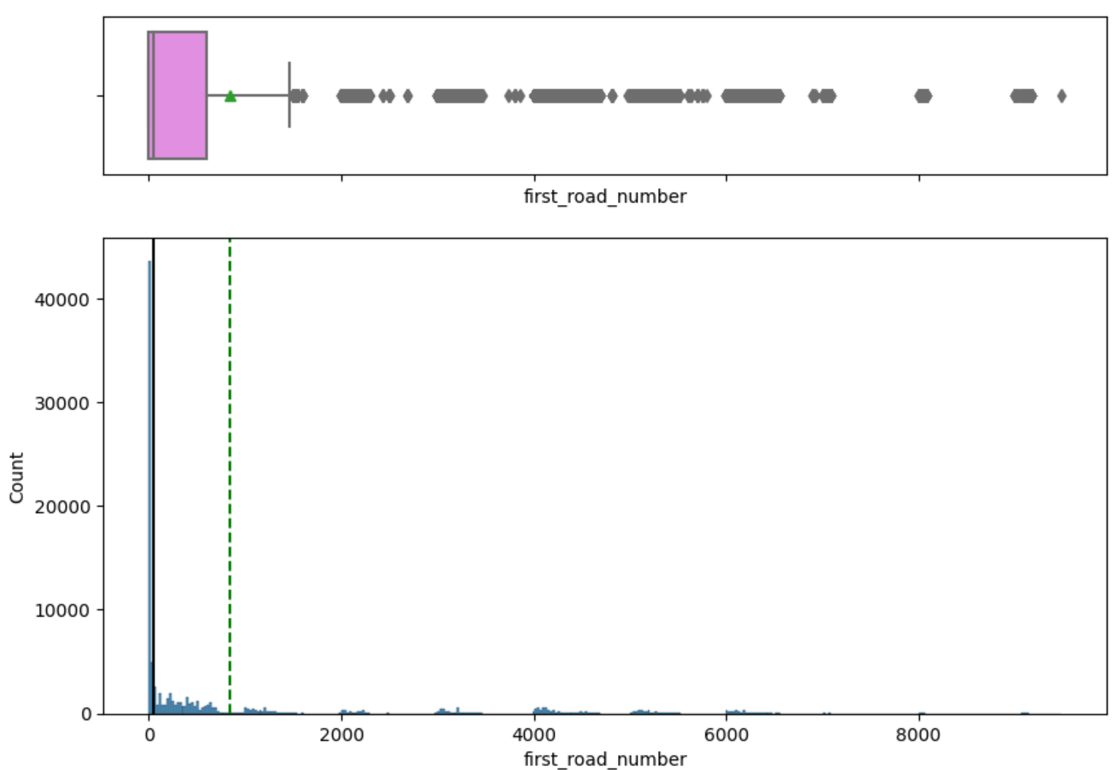
```
# Function to plot a boxplot and a histogram along the same scale

def histogram_boxplot(data, feature, figsize = (10, 7), kde = False, bins =␣
↪None):
    f2, (ax_box2, ax_hist2) = plt.subplots(
        nrows = 2,
        sharex = True,
        gridspec_kw = {"height_ratios": (0.25, 0.75)},
        figsize = figsize,
    )
    sns.boxplot(
        data = data, x = feature, ax = ax_box2, showmeans = True, color =␣
↪"violet"
    )
    sns.histplot(
        data = data, x = feature, kde = kde, ax = ax_hist2, bins = bins,␣
↪palette = "winter"
    ) if bins else sns.histplot(
```

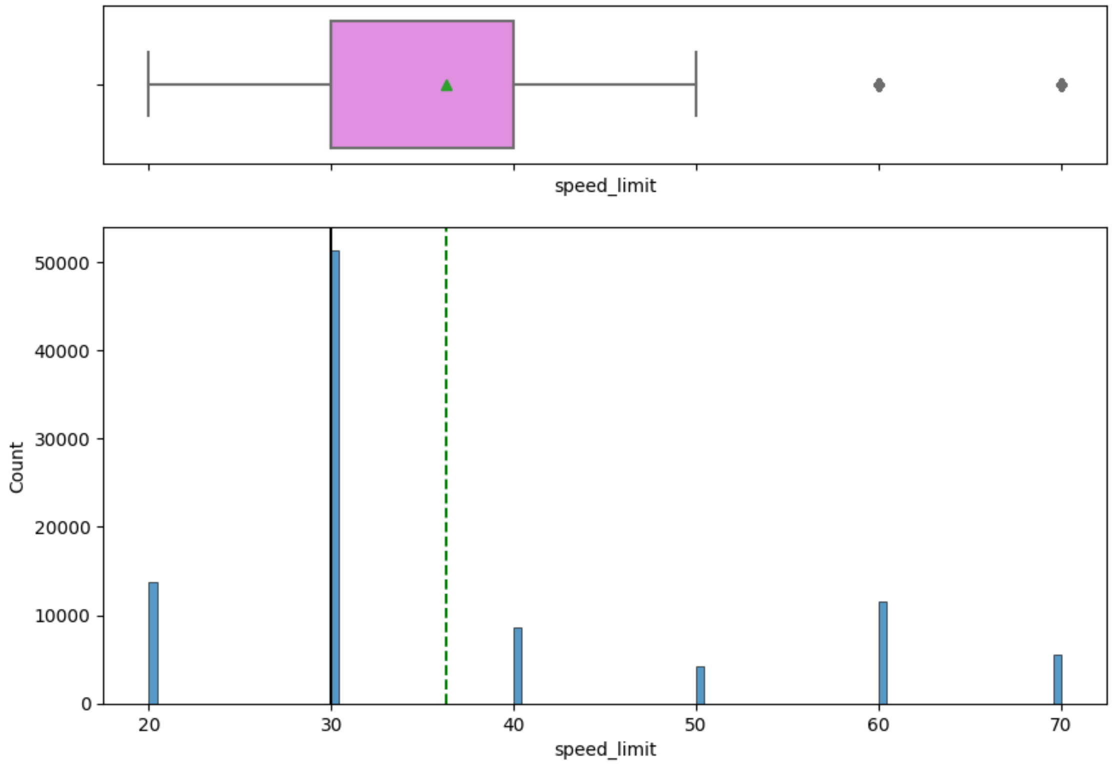
```
        data = data, x = feature, kde = kde, ax = ax_hist2
    )
    ax_hist2.axvline(
        data[feature].mean(), color = "green", linestyle = "--"
    )
    ax_hist2.axvline(
        data[feature].median(), color = "black", linestyle = "-"
    )
```

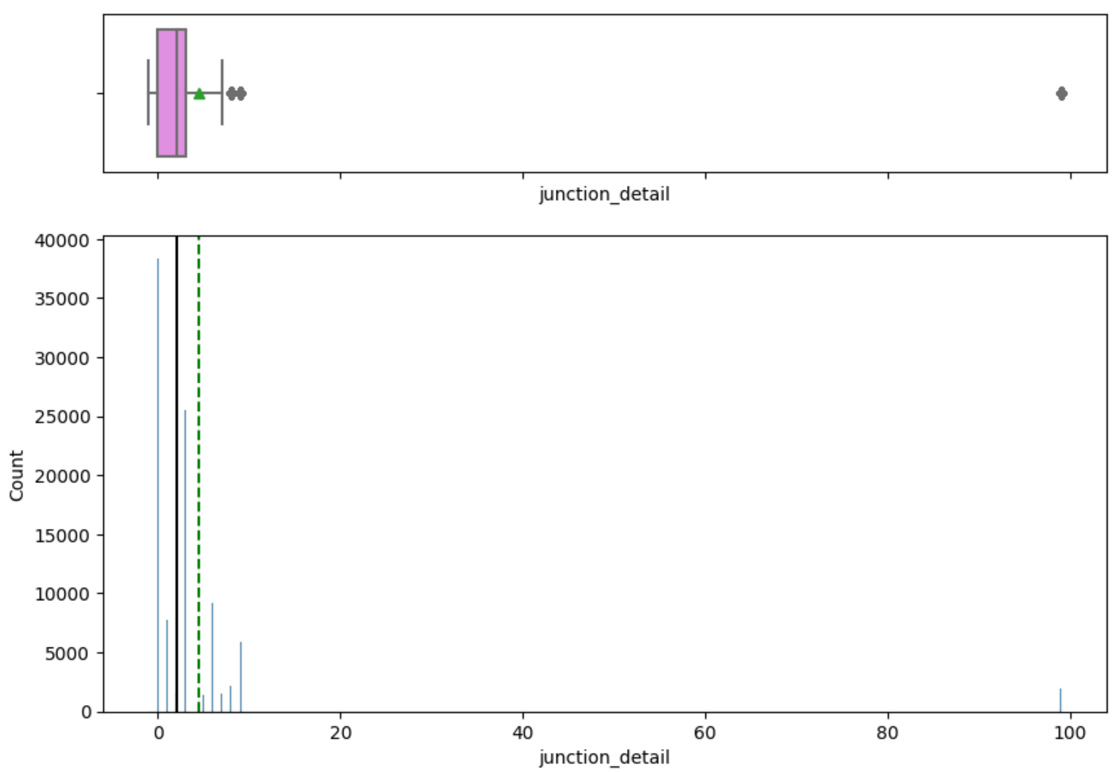
[44]:
```
for i in num_cols:
  histogram_boxplot(data, i)
```

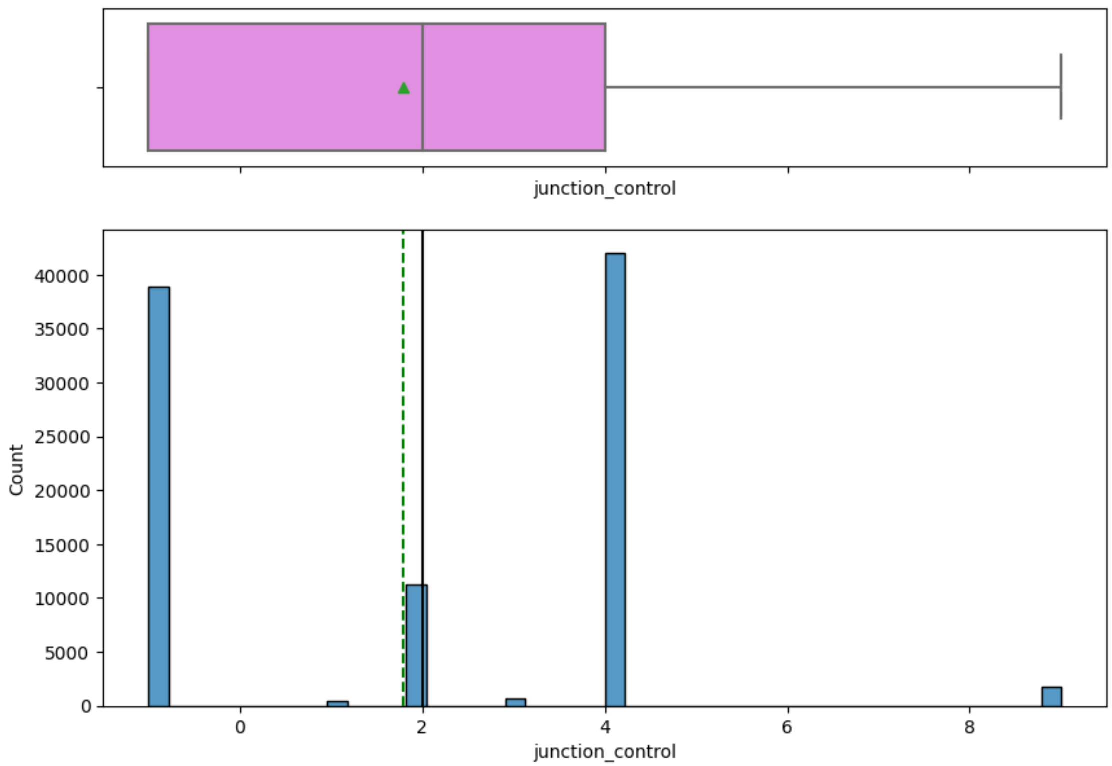



number_of_casualties
80000
60000
40000
20000
0
Count
0
5
10
15
20
number_of_casualties


first_road_number
40000
30000
20000
10000
0
Count
0
2000
4000
6000
8000
first_road_number


speed_limit
50000
40000
30000
20000
10000
0
Count
20
30
40
50
60
70
speed_limit


junction_detail
40000
35000
30000
25000
20000
15000
10000
5000
0
Count
0
20
40
60
80
100
junction_detail

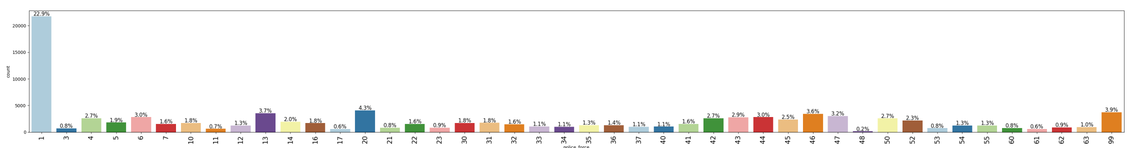


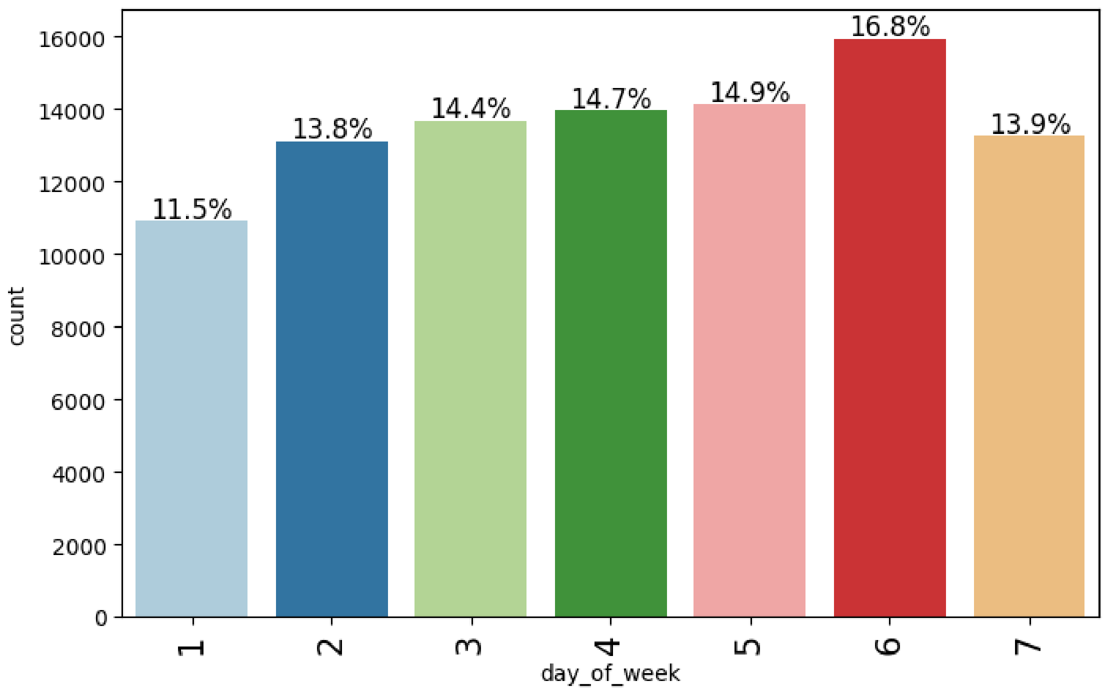
[45]:

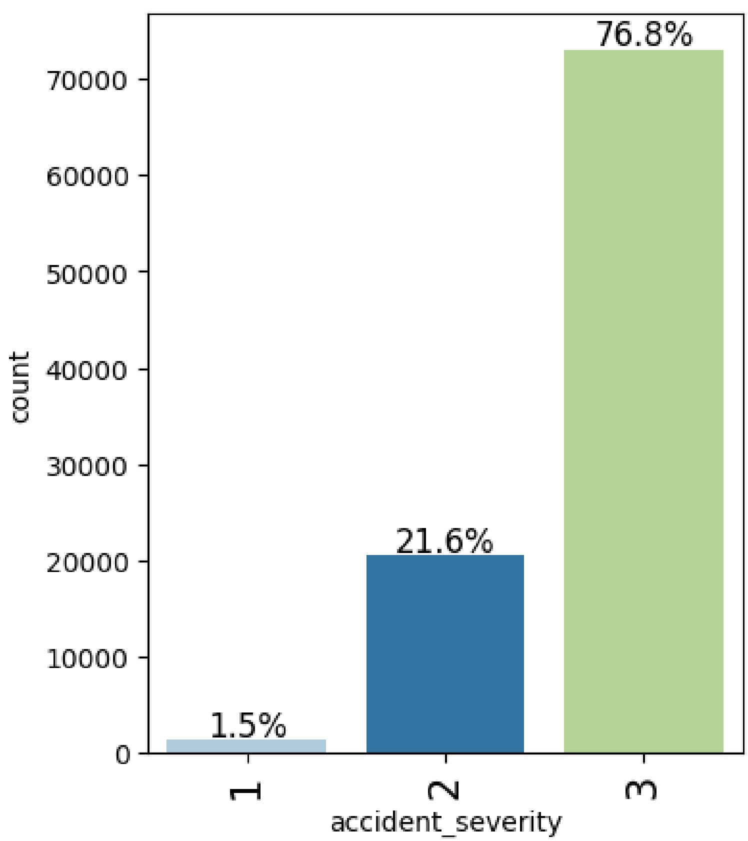
```
# Function to create labeled barplots

def labeled_barplot(data, feature, perc = False, n = None):


    total = len(data[feature])
    count = data[feature].nunique()
    if n is None:
        plt.figure(figsize = (count + 1, 5))
    else:
        plt.figure(figsize = (n + 1, 5))

    plt.xticks(rotation = 90, fontsize = 15)
    ax = sns.countplot(
        data = data,
        x = feature,
        palette = "Paired",
        order = data[feature].value_counts().index[:n].sort_values(),
    )
```

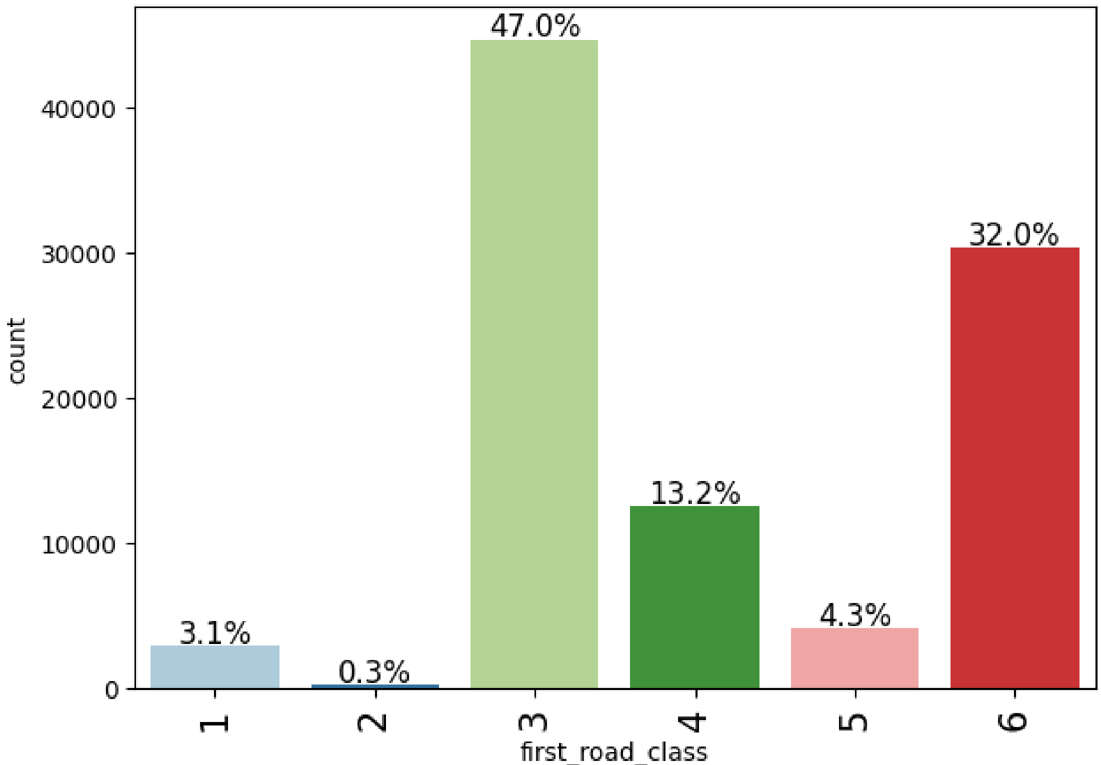
```
    for p in ax.patches:
        if perc == True:
            label = "{:.1f}%".format(
                100 * p.get_height() / total
            )
        else:
            label = p.get_height()

        x = p.get_x() + p.get_width() / 2
        y = p.get_height()

        ax.annotate(
            label,
            (x, y),
            ha = "center",
            va = "center",
            size = 12,
            xytext = (0, 5),
            textcoords = "offset points",
        )

    plt.show()
```

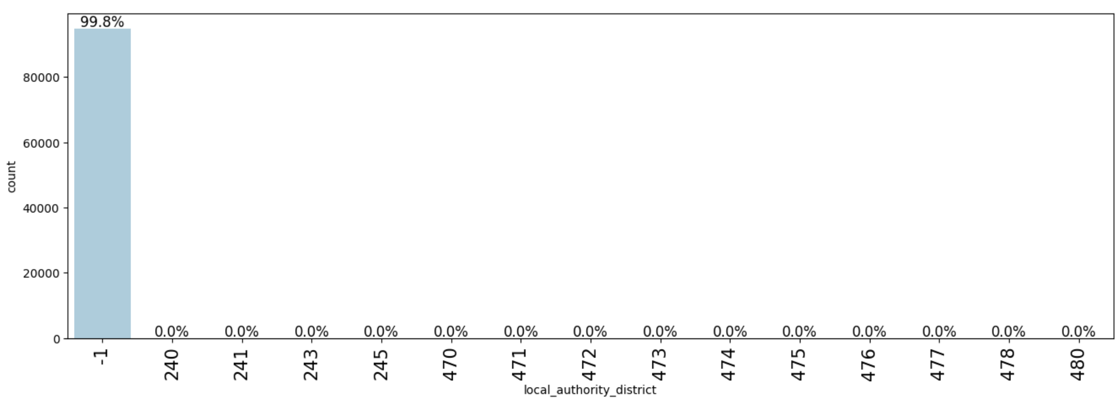
[46]:
```
for i in cat_cols:
  labeled_barplot(data, i, perc = True)
```

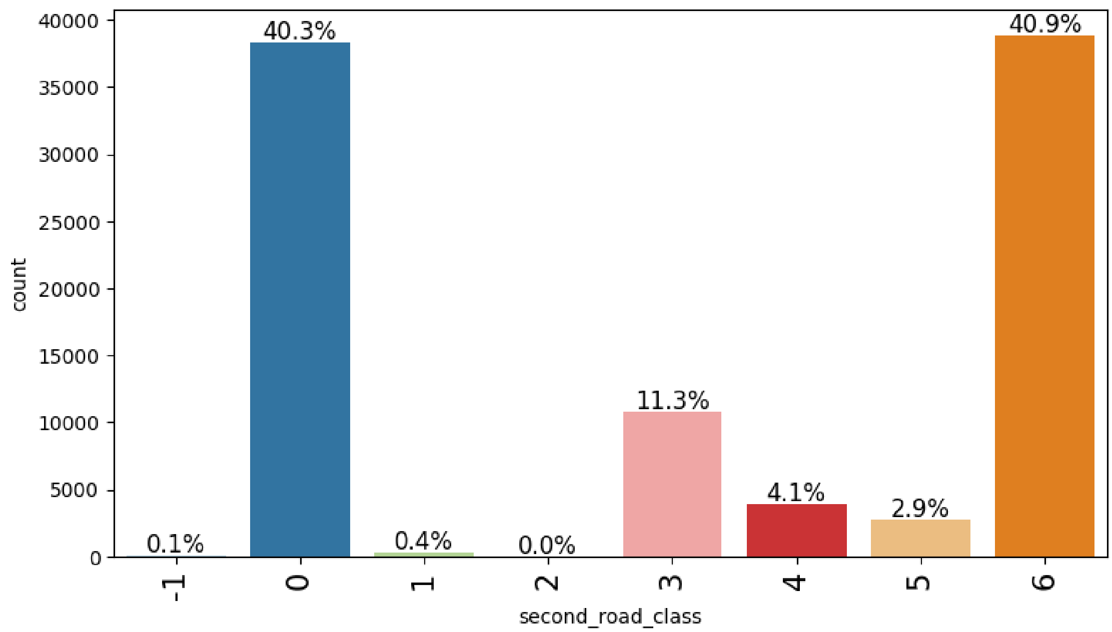



16.8%
13.8%
14.4%
14.7%
14.9%
13.9%
11.5%
count
day_of_week

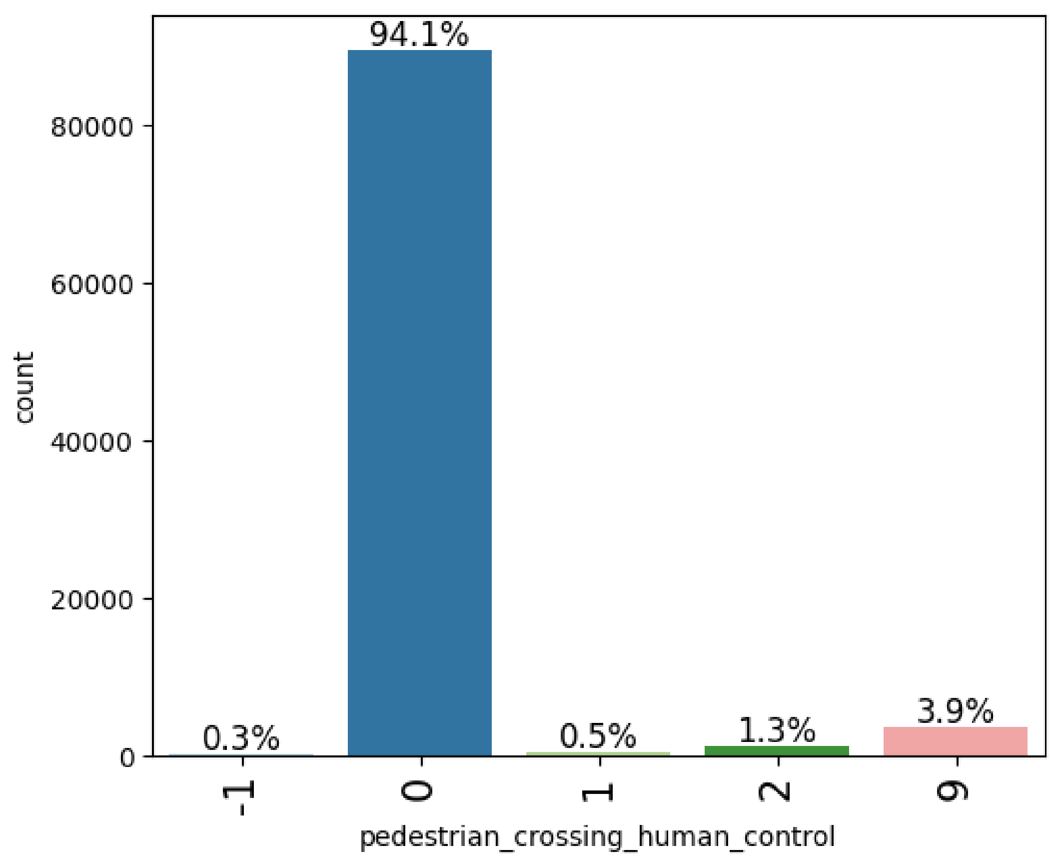

76.8%
21.6%
1.5%
count
accident_severity


47.0%
32.0%
13.2%
4.3%
3.1%
0.3%
40000
30000
20000
10000
0
count
1
2
3
4
5
6
first_road_class

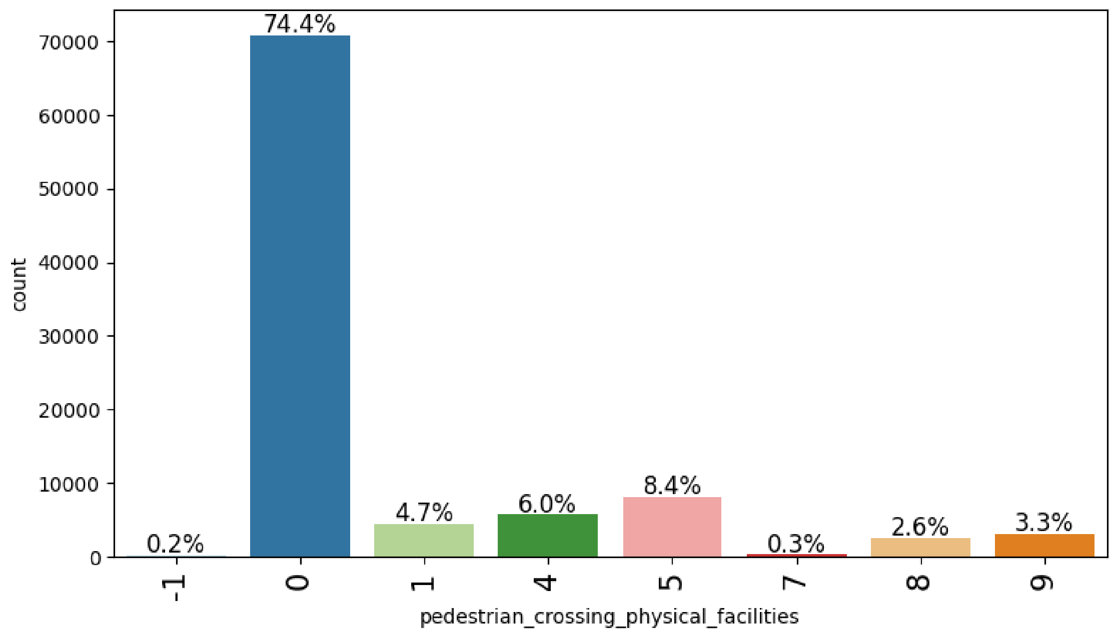

99.8%
80000
60000
40000
20000
0
count
0.0%
-1
240
241
243
245
470
471
472
473
474
475
476
477
478
480
local_authority_district

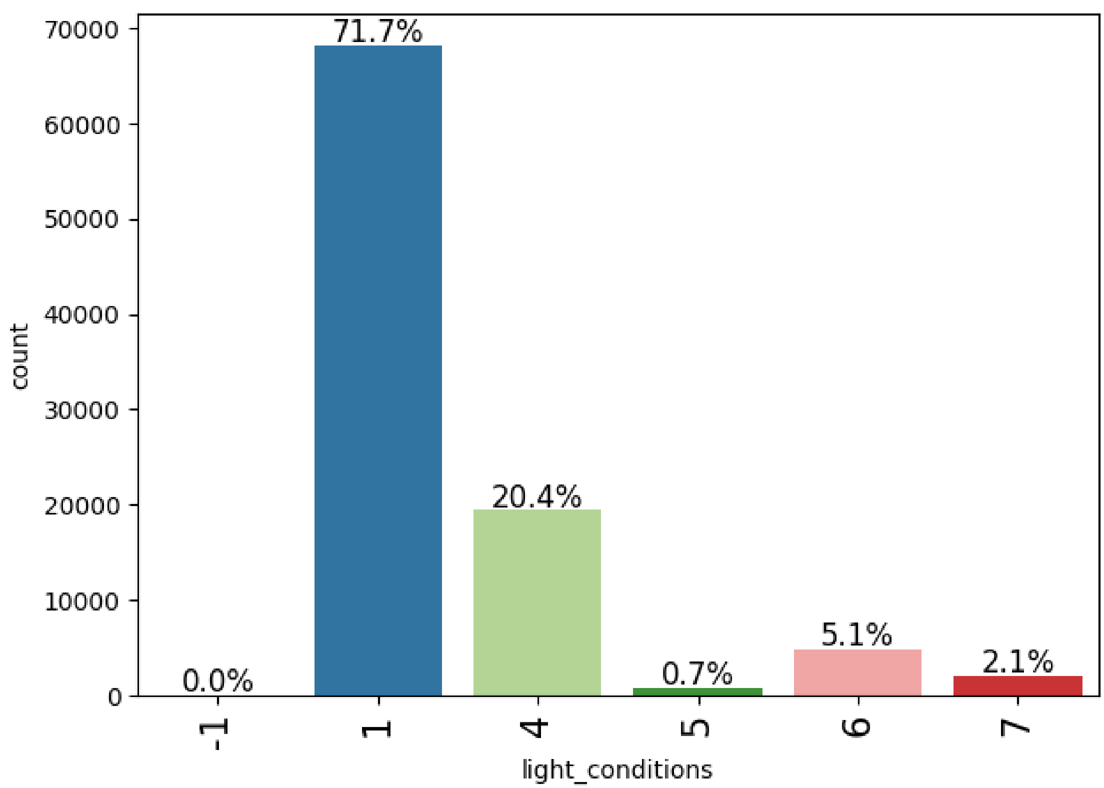


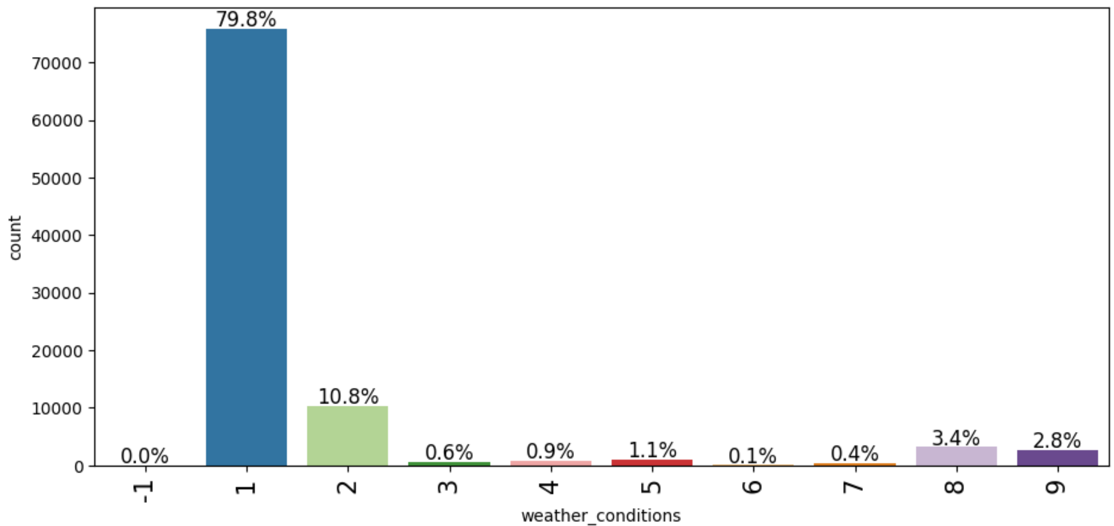
40.3%
40.9%
11.3%
4.1%
2.9%
0.1%
0.4%
0.0%
40000
35000
30000
25000
20000
15000
10000
5000
0
count
-1
0
1
2
3
4
5
6
second_road_class



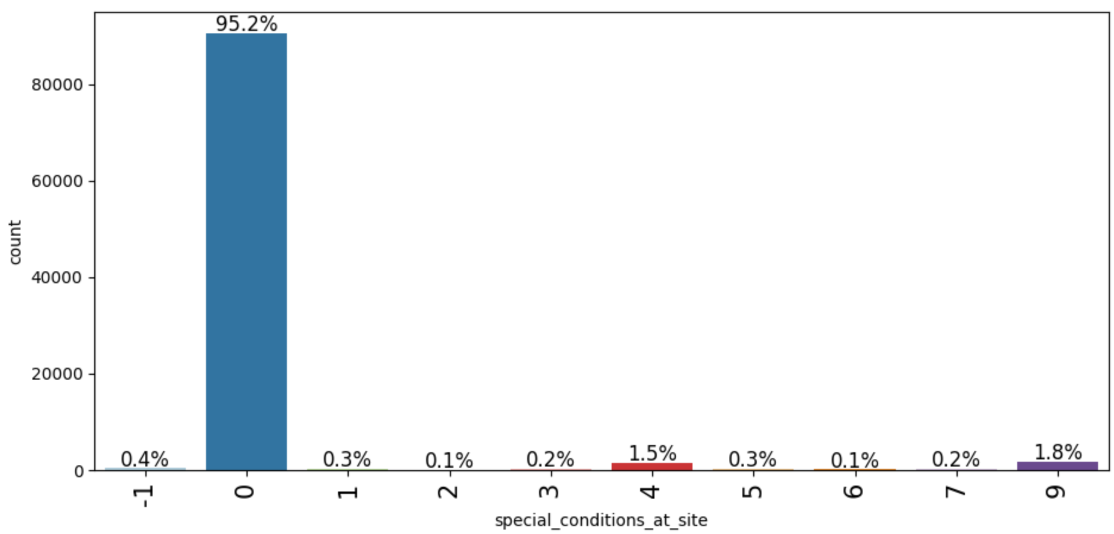
94.1%
0.3%
0.5%
1.3%
3.9%
80000
60000
40000
20000
0
count
-1
0
1
2
9
pedestrian_crossing_human_control


74.4%
4.7%
6.0%
8.4%
0.2%
0.3%
2.6%
3.3%
count
70000
60000
50000
40000
30000
20000
10000
0
-1
0
1
4
5
7
8
9
pedestrian_crossing_physical_facilities

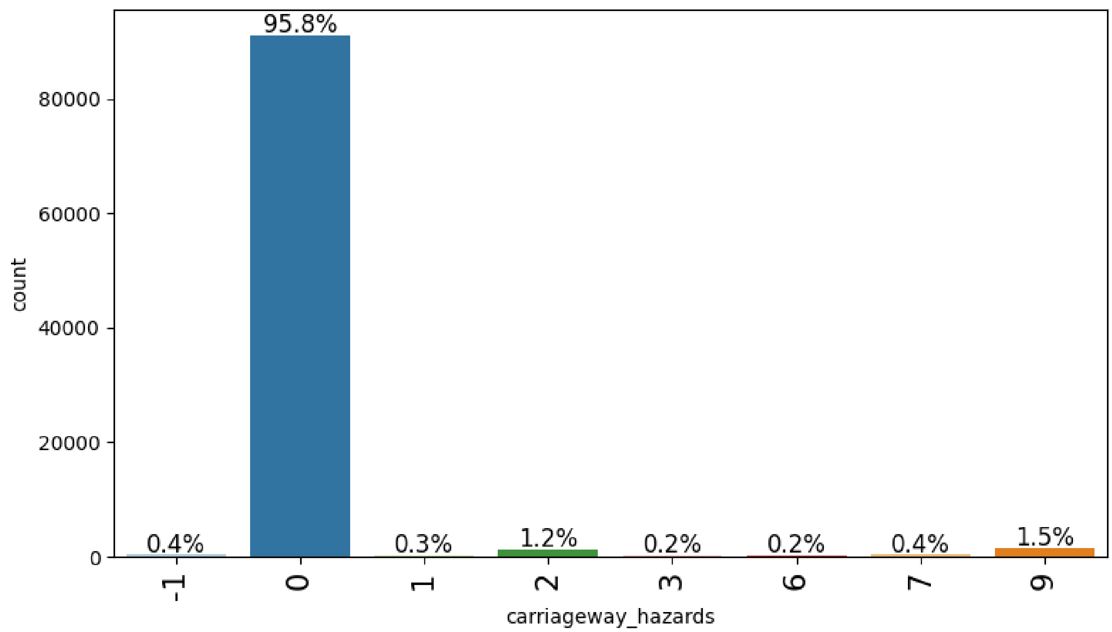

71.7%
20.4%
0.0%
0.7%
5.1%
2.1%
count
70000
60000
50000
40000
30000
20000
10000
0
-1
1
4
5
6
7
light_conditions



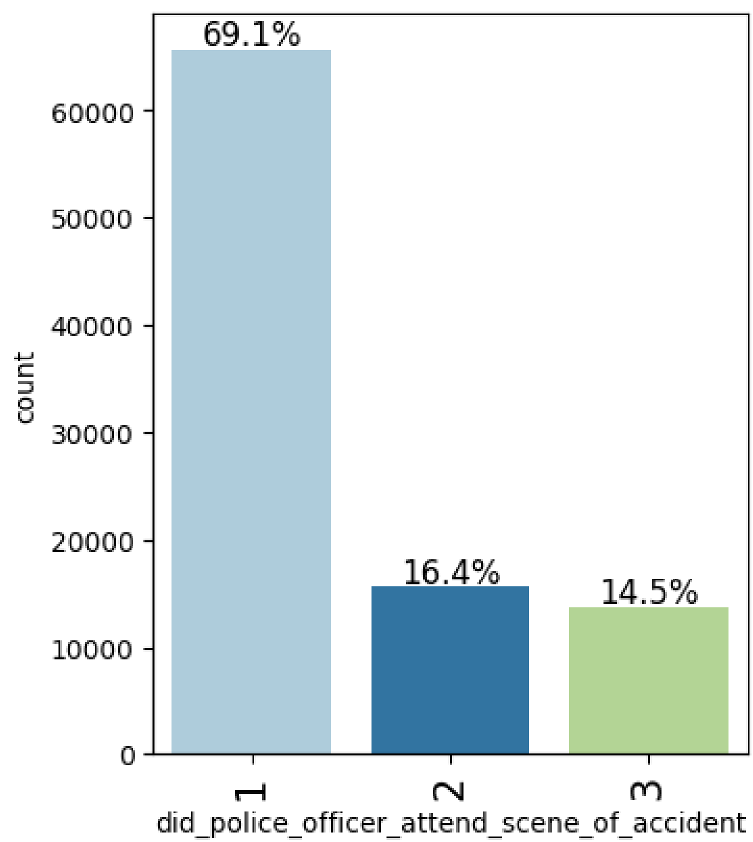
79.8%
10.8%
0.0%
0.6%
0.9%
1.1%
0.1%
0.4%
3.4%
2.8%
70000
60000
50000
40000
30000
20000
10000
0
count
-1
1
2
3
4
5
6
7
8
9
weather_conditions



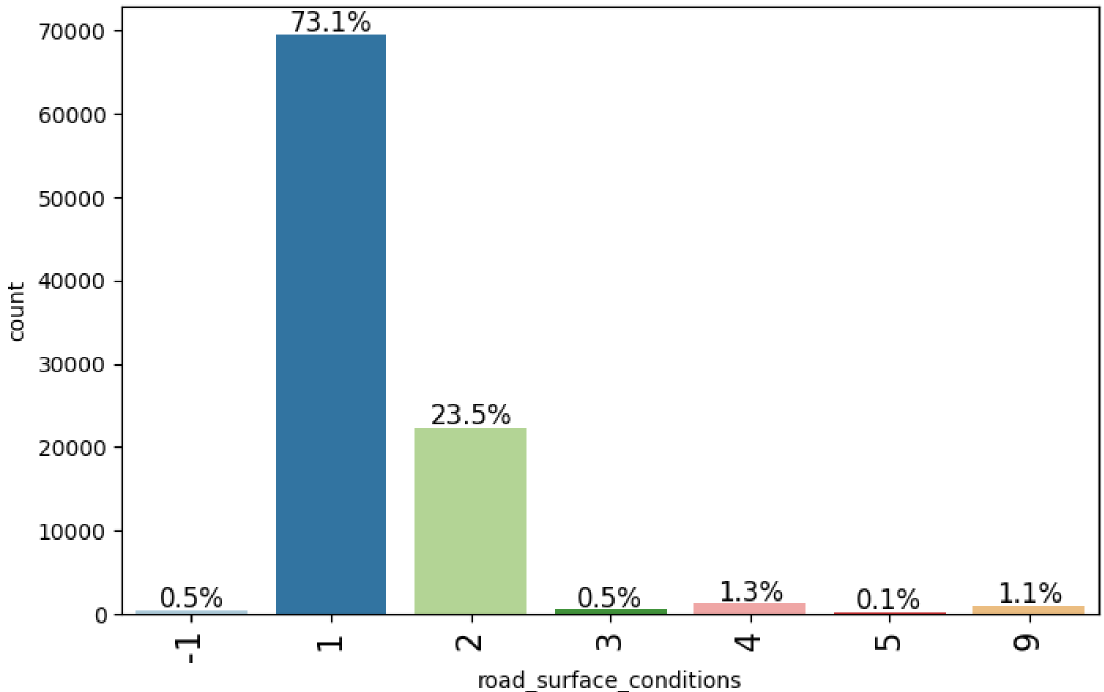
95.2%
0.4%
0.3%
0.1%
0.2%
1.5%
0.3%
0.1%
0.2%
1.8%
80000
60000
40000
20000
0
count
-1
0
1
2
3
4
5
6
7
9
special_conditions_at_site



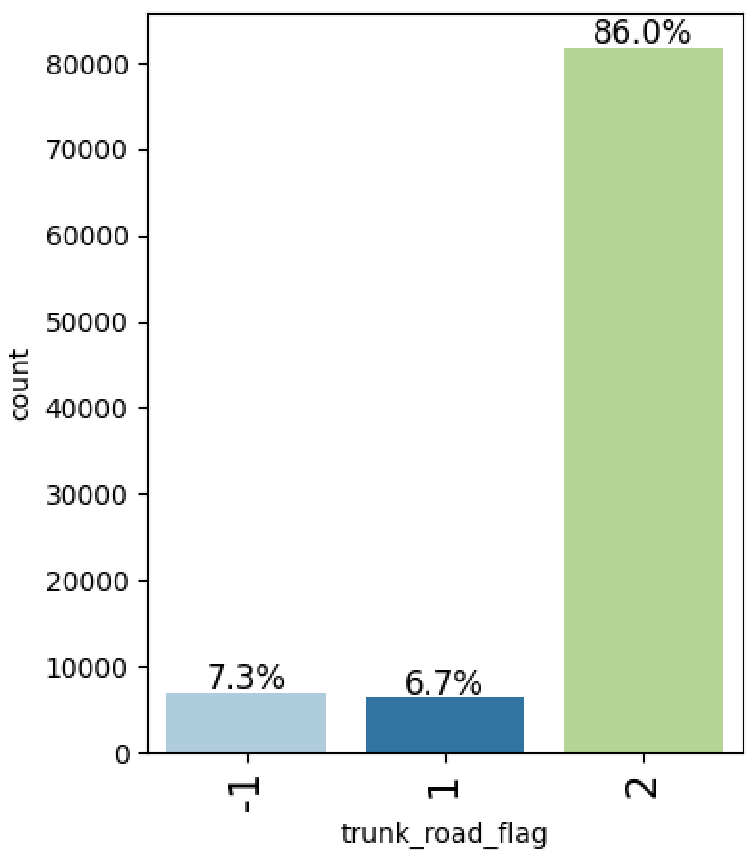
95.8%
80000
60000
40000
20000
0
count
0.4%
0.3%
1.2%
0.2%
0.2%
0.4%
1.5%
-1
0
1
2
3
6
7
9
carriageway_hazards



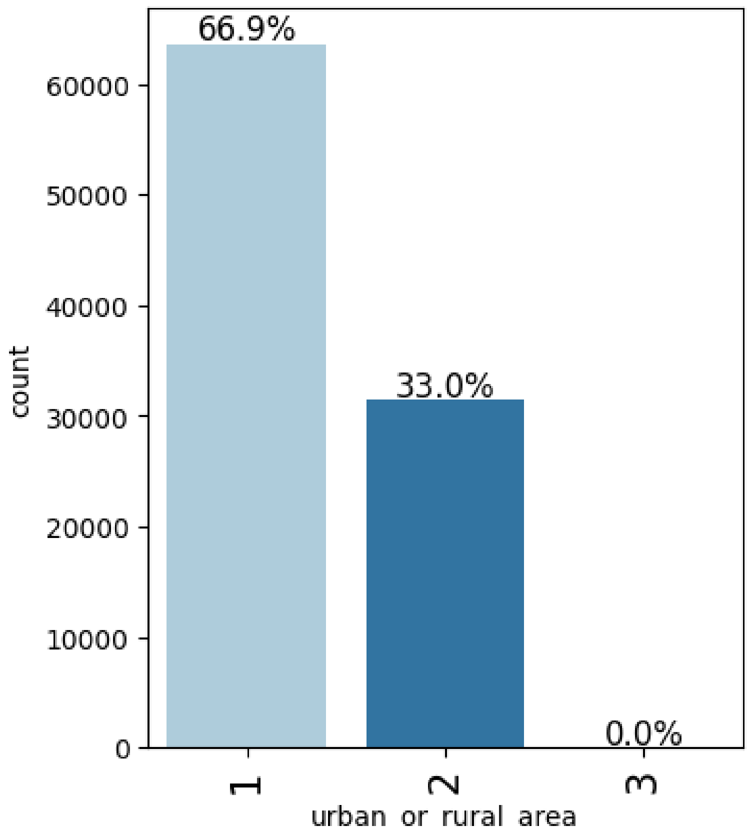
69.1%
60000
50000
40000
30000
20000
10000
0
count
16.4%
14.5%
1
2
3
did_police_officer_attend_scene_of_accident



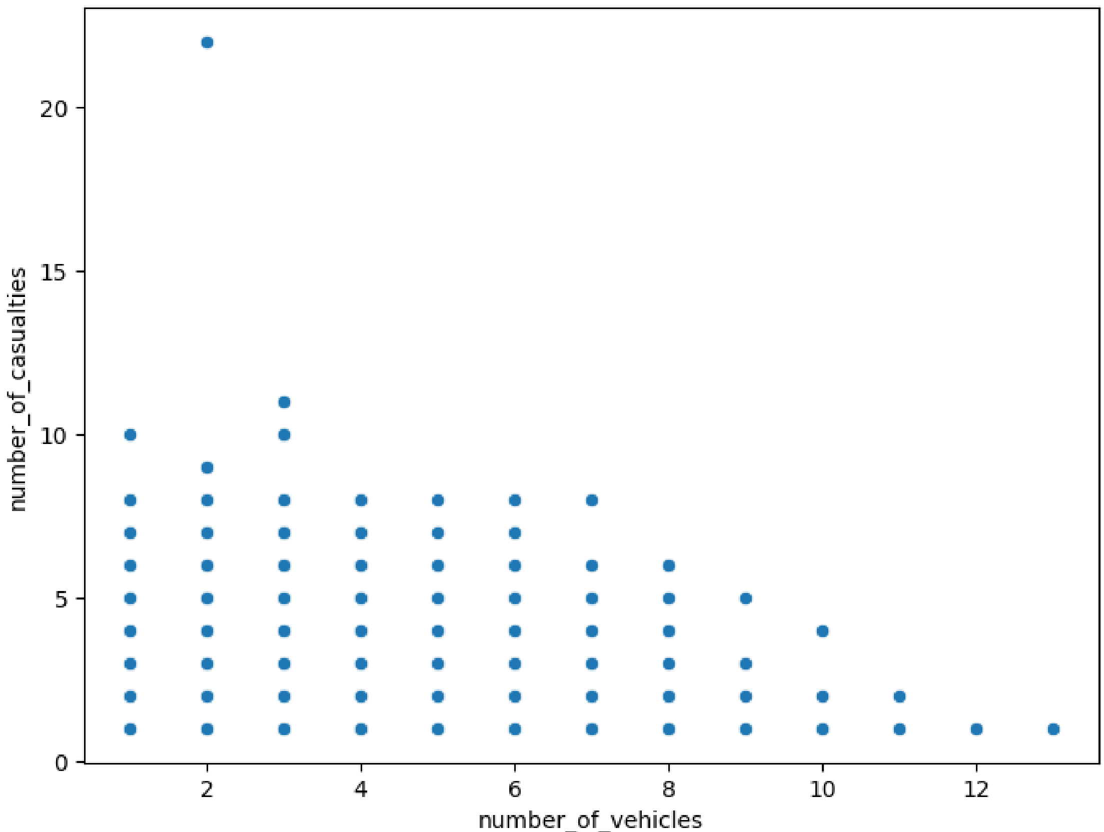
73.1%
23.5%
0.5%
0.5%
1.3%
0.1%
1.1%
count
0
10000
20000
30000
40000
50000
60000
70000
-1
1
2
3
4
5
9
road_surface_conditions



86.0%
7.3%
6.7%
count
0
10000
20000
30000
40000
50000
60000
70000
80000
-1
1
2
trunk_road_flag



[48]:
```
plt.figure(figsize = [8, 6])
sns.scatterplot(x = data.number_of_vehicles, y = data.number_of_casualties)
plt.show()
```



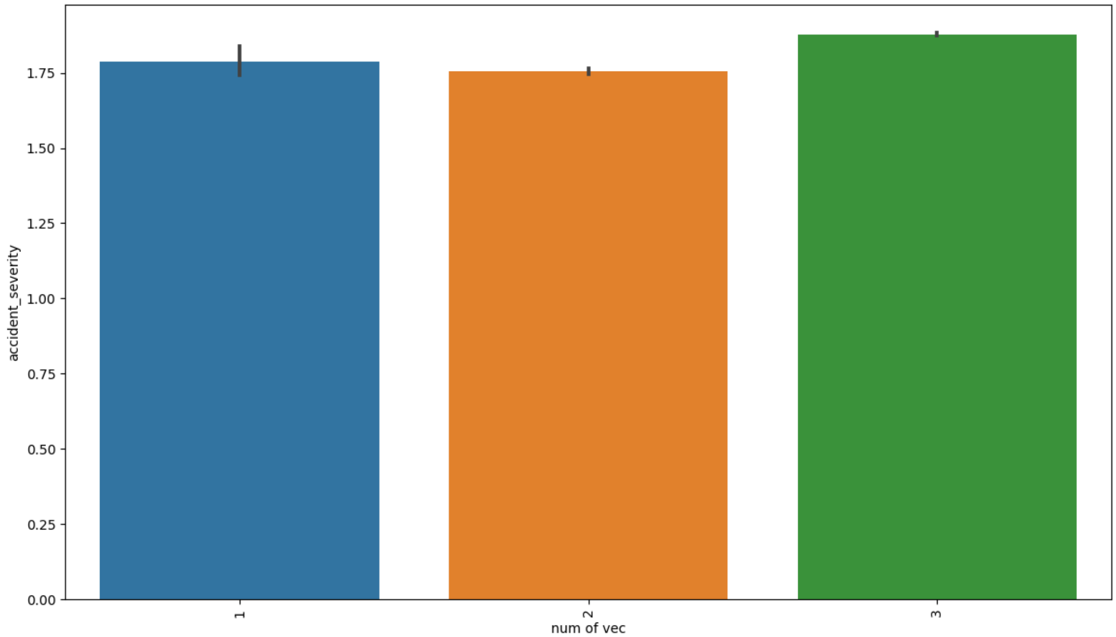
[47]:
```
data.head()
```

[47]:
```
   police_force accident_severity  number_of_vehicles  number_of_casualties  \
0             1                 3                   3                     1
1             1                 2                   2                     3
2             1                 2                   2                     4
3             1                 1                   1                     1
4             1                 3                   4                     1

   day_of_week local_authority_district first_road_class  first_road_number  \
0            6                       -1                6                  0
1            6                       -1                3               1203
2            6                       -1                4                272
3            6                       -1                3               3220
4            6                       -1                5                  0

    road_type  speed_limit  junction_detail  junction_control  \
0           6           30                9                 4
1           3           30                7                 2
2           6           30                9                 2
```

```
3           2            30                 9                 4
4           6            20                 3                 4

  second_road_class  second_road_number pedestrian_crossing_human_control  \
0                 6                   0                                 0
1                 3                1204                                 0
2                 5                   0                                 0
3                 6                   0                                 0
4                 6                   0                                 0

  pedestrian_crossing_physical_facilities light_conditions weather_conditions  \
0                                       0                4                  7
1                                       5                4                  1
2                                       5                4                  1
3                                       4                4                  1
4                                       0                4                  1

  road_surface_conditions special_conditions_at_site carriageway_hazards  \
0                       4                          1                   0
1                       1                          0                   0
2                       1                          0                   0
3                       1                          0                   0
4                       1                          0                   0

  urban_or_rural_area did_police_officer_attend_scene_of_accident  \
0                   1                                           1
1                   1                                           1
2                   1                                           1
3                   1                                           1
4                   1                                           1

  trunk_road_flag
0               2
1               2
2               2
3               2
4               2
```

[50]:

```
df_revenue1 = data.groupby(["number_of_vehicles"], as_index = False)[
    "accident_severity"
].sum()
plt.figure(figsize = [14, 8])
plt.xticks(rotation = 90)
a = sns.barplot(x = data.accident_severity, y = data.number_of_vehicles)
a.set_xlabel("num of vec")
a.set_ylabel("accident_severity")
plt.show()
```



[54]:
```
# Creating the list of columns for which we need to create the dummy variables
to_get_dummies_for = ['police_force',
 'day_of_week',
 'first_road_class',
 'local_authority_district',
 'second_road_class',
 'pedestrian_crossing_human_control',
 'pedestrian_crossing_physical_facilities',
 'light_conditions',
 'weather_conditions',
 'special_conditions_at_site',
 'carriageway_hazards',
 'did_police_officer_attend_scene_of_accident',
 'road_surface_conditions',
 'trunk_road_flag',
 'urban_or_rural_area']

# Creating dummy variables
data = pd.get_dummies(data = data, columns = to_get_dummies_for, drop_first =␣
↪True)
```

[56]:
```
# Separating the target variable and other variables
X = data.drop(columns = ['accident_severity'])
y = data.accident_severity
```

[57]:
```
# Splitting the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3,␣
↪random_state = 1, stratify = y)
```

Model Building

[58]:
```
# reference for function is used fro google.
def model_performance_classification_sklearn(model, predictors, target):

    pred = model.predict(predictors)

    acc = accuracy_score(target, pred)
    recall = recall_score(target, pred, average = "weighted")
    precision = precision_score(target, pred, average = 'weighted')
    f1 = f1_score(target, pred, average = 'weighted')

    df_perf = pd.DataFrame(
        {"Accuracy": acc, "Recall": recall, "Precision": precision, "F1": f1,},
        index=[0],
    )

    return df_perf
```

Training models on original data

[65]:
```
models = []

models.append(("Logistic regression", LogisticRegression(random_state=1)))
models.append(("dtree", DecisionTreeClassifier(random_state=1)))
models.append(("Random forest", RandomForestClassifier(random_state=1)))


print("\n" "Training Performance:" "\n")
for name, model in models:
    model.fit(X_train, y_train)
    scores_train = recall_score(y_train, model.predict(X_train), average =␣
↪"weighted")
    print("{}: {}".format(name, scores_train))

print("\n" "Validation Performance:" "\n")

for name, model in models:
    model.fit(X_train, y_train)
    scores_val = recall_score(y_test, model.predict(X_test), average =␣
↪"weighted")
    print("{}: {}".format(name, scores_val))
```

```
Training Performance:

Logistic regression: 0.7685047417226506
dtree: 0.9869546267490269
Random forest: 0.9869395975171709

Validation Performance:

Logistic regression: 0.7684808528545378
dtree: 0.6419553934633189
Random forest: 0.7416888764202553
```

- The training performance of the decision tree and random forest models is very high, with accuracy scores of 0.9869 and 0.9869 respectively. This indicates that these models are likely overfitting the training data, as they are performing almost perfectly on the training set but not as well on the validation set.
- In contrast, the training performance of the logistic regression model is lower, with an accuracy score of 0.7685. This suggests that the logistic regression model may be less prone to overfitting, and may be a more robust model in this case. The validation performance of the logistic regression model is similar to its training performance, with an accuracy score of 0.7685. This suggests that the model is likely performing consistently across both the training and validation sets, and may be a good candidate for deployment in practice.
- The validation performance of the decision tree model is much lower than its training performance, with an accuracy score of 0.6420. This indicates that the model is overfitting the training data, and is not generalizing well to new data.
- The validation performance of the random forest model is also lower than its training performance, but not as much as the decision tree model. Its accuracy score on the validation set is 0.7417, which is still a relatively good performance, but may indicate some overfitting.

[63]:
```python
print("Before Oversampling,'1': {}".format(sum(y_train == 1)))
print("Before Oversampling,'2': {} ".format(sum(y_train == 2)))
print("Before Oversampling,'3': {} ".format(sum(y_train == 3)))
```

```
Before Oversampling,'1': 1030
Before Oversampling,'2': 14373
Before Oversampling,'3': 51134
```

[64]:
```python
sm = SMOTE(
    sampling_strategy="auto", k_neighbors=3, random_state=1
)  # Synthetic Minority Over Sampling Technique
X_train_over, y_train_over = sm.fit_resample(X_train, y_train)
```

[66]:
```python
models = []

models.append(("Logistic regression", LogisticRegression(random_state=1)))
models.append(("dtree", DecisionTreeClassifier(random_state=1)))
models.append(("Random forest", RandomForestClassifier(random_state=1)))
```

```python
print("\n" "Training Performance:" "\n")
for name, model in models:
    model.fit(X_train_over, y_train_over)
    scores_train = recall_score(y_train_over, model.predict(X_train_over),␣
 ↪average = "weighted")
    print("{}: {}".format(name, scores_train))

print("\n" "Validation Performance:" "\n")

for name, model in models:
    model.fit(X_train, y_train)
    scores_val = recall_score(y_test, model.predict(X_test), average =␣
 ↪"weighted")
    print("{}: {}".format(name, scores_val))
```

```
Training Performance:

Logistic regression: 0.4662977014641269
dtree: 0.9880118903273751
Random forest: 0.9880053715075423

Validation Performance:

Logistic regression: 0.7684808528545378
dtree: 0.6419553934633189
Random forest: 0.7416888764202553
```

- The training performance of the decision tree and random forest models is very high (close to 1.0), while the training performance of the logistic regression model is relatively low (around 0.47). This suggests that the decision tree and random forest models may be overfitting the training data, while the logistic regression model may not be complex enough to capture the patterns in the data.
- The validation performance of the logistic regression model is better than the other models (0.77 vs. 0.64 and 0.74 for the decision tree and random forest, respectively). This suggests that the logistic regression model may be better at generalizing to new, unseen data than the other models.
- he validation performance of the decision tree model is the worst among the three models. This is consistent with the observation that the decision tree model may be overfitting the training data.
- The random forest model has a relatively high training performance but a lower validation performance than the logistic regression model. This suggests that the random forest model may be overfitting the training data to some extent, but not as severely as the decision tree model.
- It may be a good idea to try tuning the hyperparameters of the decision tree and random

forest models using techniques like grid search or random search to see if their performance can be improved. It may also be worth exploring other classification algorithms to see if they can achieve better performance on this dataset.

Tuning random forest on the orignal data

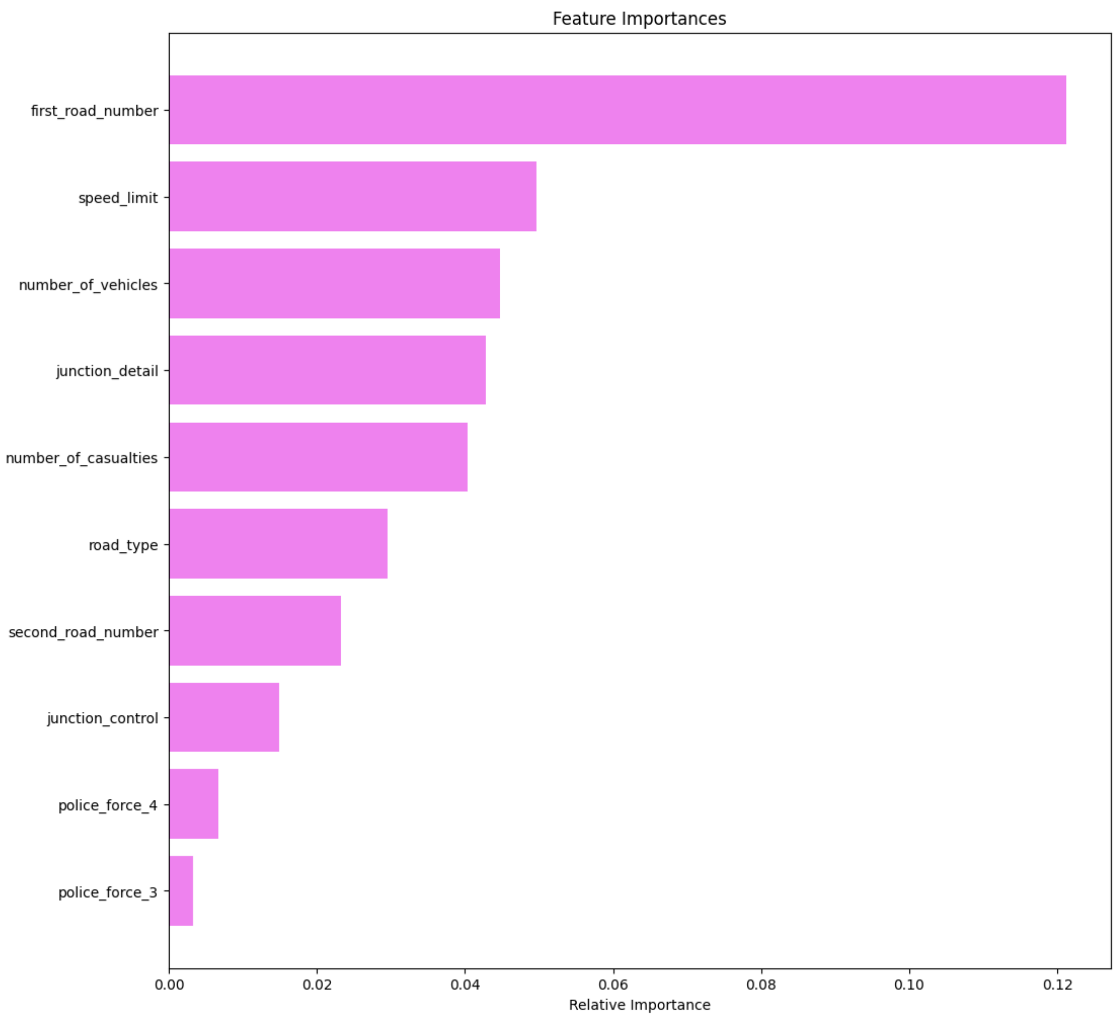
[68]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score
import pandas as pd
import numpy as np

# Define the parameter grid for Grid Search
param_grid = {
    'n_estimators': [50, 100],
    'max_depth': [10, 20],
    'max_features': [3, 6],
    'min_samples_split': [2, 4],
    'min_samples_leaf': [1, 2]
}

# Define the Random Forest model
rf = RandomForestClassifier(random_state=42)

# Perform Grid Search to find the best parameters
grid_search = GridSearchCV(rf, param_grid=param_grid, cv=5, n_jobs=-1)
grid_search.fit(X_train, y_train)
```

[ ]:
```
# # Print the best parameters and accuracy score
# print("Best parameters: ", grid_search.best_params_)
# print("Best accuracy score: ", grid_search.best_score_)

# Make predictions using the best model
best_rf = grid_search.best_estimator_
y_pred = best_rf.predict(X_train)

# # Evaluate the performance of the best model
# accuracy = accuracy_score(y_test, y_pred)
# print("Accuracy score: ", accuracy)

# scores_train = recall_score(y_train, model.predict(X_train), average =
 ↪"weighted")
# print("{Model }: {}".format(name, scores_train))
```

Feature importance

[74]:
```
best_rf = models[2][1]
feature_names = X_train.columns
```

```
importances = best_rf.feature_importances_[:10]
indices = np.argsort(importances)

plt.figure(figsize=(12, 12))
plt.title("Feature Importances")
plt.barh(range(len(indices)), importances[indices], color="violet",␣
 ↪align="center")
plt.yticks(range(len(indices)), [feature_names[i] for i in indices])
plt.xlabel("Relative Importance")
plt.show()
```



- we can also observe the top 10 features

0.1.1 Summary/Conlusion

EDA:

We performed exploratory data analysis (EDA) on the accident severity dataset and found that there are 24 features in total, including factors such as the number of vehicles involved, the weather conditions, and whether a police officer attended the scene. We also observed that some features are categorical and others are numerical, and that there are some missing values in the dataset.

Model Development:

We then built three different machine learning models - logistic regression, decision tree, and random forest - to predict accident severity based on the features in the dataset. We trained these models on a training set and evaluated their performance on a validation set.

Results:

Our analysis showed that the decision tree and random forest models had very high accuracy scores on the training set, but lower scores on the validation set. This suggests that these models are overfitting the training data and may not generalize well to new data. The logistic regression model, on the other hand, had a lower accuracy score on the training set, but a similar score on the validation set, indicating that it may be a more robust model for predicting accident severity.

Conclusion:

Based on our analysis, we recommend using a logistic regression model to predict accident severity, as it appears to be more robust and less prone to overfitting. We also recommend further investigation into the most important features for predicting accident severity, as this information could be useful for developing targeted interventions to reduce the number and severity of accidents. Overall, our analysis demonstrates the potential of machine learning techniques for improving road safety and reducing the human and economic costs of accidents.

[ ]:

```

```