Appendix_A_EDA_and_Feature_Selection

August 10, 2024

1 Exploratory Data Analysis

This notebook contains the code and details on the EDA process and feature selection.

Exploratory data was split amongst the three-member team each analyzing one-third of the dataset.

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

from pathlib import Path
from scipy.stats import chi2_contingency
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_selection import SelectKBest, chi2
```

[ ]:
```
%run ../custom/jc-functions.ipynb
```

[ ]:
```
# Import training dataset
dataset = Path('../dataset')
df = pd.read_csv(dataset/"accidents_train.csv")
df.head()
```

[ ]:
```
   Num      Time Day_of_week Age_band_of_driver Sex_of_driver  \
0    1  17:02:00      Monday              18-30          Male
1    2  17:02:00      Monday              31-50          Male
2    3  17:02:00      Monday              18-30          Male
3    4   1:06:00      Sunday              18-30          Male
4    5   1:06:00      Sunday              18-30          Male

    Educational_level Vehicle_driver_relation Driving_experience  \
0   Above high school                Employee              1-2yr
1  Junior high school                Employee          Above 10yr
2  Junior high school                Employee              1-2yr
3  Junior high school                Employee             5-10yr
4  Junior high school                Employee              2-5yr

       Type_of_vehicle Owner_of_vehicle  … Vehicle_movement  \
```

```
0           Automobile            Owner  …   Going straight
1  Public (> 45 seats)            Owner  …   Going straight
2      Lorry (41?100Q)            Owner  …   Going straight
3  Public (> 45 seats)     Governmental  …   Going straight
4                  NaN            Owner  …   Going straight

    Casualty_class Sex_of_casualty Age_band_of_casualty Casualty_severity  \
0               na              na                   na                na
1               na              na                   na                na
2  Driver or rider            Male                31-50                 3
3       Pedestrian          Female                18-30                 3
4               na              na                   na                na

  Work_of_casuality Fitness_of_casuality Pedestrian_movement  \
0               NaN                  NaN    Not a Pedestrian
1               NaN                  NaN    Not a Pedestrian
2            Driver                  NaN    Not a Pedestrian
3            Driver               Normal    Not a Pedestrian
4               NaN                  NaN    Not a Pedestrian

            Cause_of_accident Accident_severity
0             Moving Backward     Slight Injury
1                  Overtaking     Slight Injury
2   Changing lane to the left    Serious Injury
3  Changing lane to the right     Slight Injury
4                  Overtaking     Slight Injury

[5 rows x 33 columns]
```

[ ]:
```
df.columns
```

[ ]:
```
Index(['Num', 'Time', 'Day_of_week', 'Age_band_of_driver', 'Sex_of_driver',
       'Educational_level', 'Vehicle_driver_relation', 'Driving_experience',
       'Type_of_vehicle', 'Owner_of_vehicle', 'Service_year_of_vehicle',
       'Defect_of_vehicle', 'Area_accident_occured', 'Lanes_or_Medians',
       'Road_allignment', 'Types_of_Junction', 'Road_surface_type',
       'Road_surface_conditions', 'Light_conditions', 'Weather_conditions',
       'Type_of_collision', 'Number_of_vehicles_involved',
       'Number_of_casualties', 'Vehicle_movement', 'Casualty_class',
       'Sex_of_casualty', 'Age_band_of_casualty', 'Casualty_severity',
       'Work_of_casuality', 'Fitness_of_casuality', 'Pedestrian_movement',
       'Cause_of_accident', 'Accident_severity'],
      dtype='object')
```

[ ]:
```
df.shape
```

[ ]:
```
(8210, 33)
```

Original dataset has 8210 rows and 33 features. The target feature is “Accident_severity”.

```
[ ]: target = 'Accident_severity'
my_list = df.columns.tolist()
set3 = my_list[21:33]
print(len(set3), set3)
```

```
12 ['Number_of_vehicles_involved', 'Number_of_casualties', 'Vehicle_movement',
'Casualty_class', 'Sex_of_casualty', 'Age_band_of_casualty',
'Casualty_severity', 'Work_of_casuality', 'Fitness_of_casuality',
'Pedestrian_movement', 'Cause_of_accident', 'Accident_severity']
```

1.1 Dataset split into 3 sets of features for EDA by team

1.1.1 Dataset 1

```
[ ]: selected_columns = ['Num', 'Time', 'Day_of_week', 'Age_band_of_driver',␣
  ↪'Sex_of_driver', 'Educational_level', 'Vehicle_driver_relation',␣
  ↪'Driving_experience', 'Type_of_vehicle', 'Owner_of_vehicle',␣
  ↪'Service_year_of_vehicle', 'Accident_severity']
df_matt = df[selected_columns]
print(df_matt.dtypes)
```

```
Num                          int64
Time                        object
Day_of_week                 object
Age_band_of_driver          object
Sex_of_driver               object
Educational_level           object
Vehicle_driver_relation     object
Driving_experience          object
Type_of_vehicle             object
Owner_of_vehicle            object
Service_year_of_vehicle     object
Accident_severity           object
dtype: object
```

```
[ ]: print(df_matt.describe)
```

```
<bound method NDFrame.describe of        Num      Time Day_of_week
Age_band_of_driver Sex_of_driver  \
0          1  17:02:00      Monday              18-30          Male
1          2  17:02:00      Monday              31-50          Male
2          3  17:02:00      Monday              18-30          Male
3          4   1:06:00      Sunday              18-30          Male
4          5   1:06:00      Sunday              18-30          Male
…        …         …          …                 …            …
8205    8206  17:40:00      Monday              18-30          Male
8206    8207  17:45:00     Tuesday            Over 51          Male
8207    8208  17:45:00     Tuesday              18-30          Male
```

```
8208  8209   8:25:00    Thursday              18-30          Male
8209  8210   8:25:00    Thursday              31-50          Male

       Educational_level Vehicle_driver_relation Driving_experience  \
0       Above high school               Employee             1-2yr
1      Junior high school               Employee          Above 10yr
2      Junior high school               Employee             1-2yr
3      Junior high school               Employee            5-10yr
4      Junior high school               Employee             2-5yr
…                     …                      …                …
8205   Junior high school               Employee            5-10yr
8206          High school               Employee            5-10yr
8207   Junior high school               Employee            5-10yr
8208   Junior high school               Employee          Above 10yr
8209   Junior high school                    NaN             1-2yr

          Type_of_vehicle Owner_of_vehicle Service_year_of_vehicle  \
0              Automobile            Owner               Above 10yr
1     Public (> 45 seats)            Owner                  5-10yrs
2         Lorry (41?100Q)            Owner                      NaN
3     Public (> 45 seats)     Governmental                      NaN
4                     NaN            Owner                  5-10yrs
…                     …              …                        …
8205      Lorry (41?100Q)            Owner                      NaN
8206     Pick up upto 10Q            Owner                  5-10yrs
8207                Other            Owner                  5-10yrs
8208      Lorry (41?100Q)            Owner                   2-5yrs
8209         Ridden horse            Owner                   2-5yrs

     Accident_severity
0        Slight Injury
1        Slight Injury
2       Serious Injury
3        Slight Injury
4        Slight Injury
…                  …
8205     Slight Injury
8206     Slight Injury
8207     Slight Injury
8208     Slight Injury
8209     Slight Injury

[8210 rows x 12 columns]>
```

[ ]:
```
print(df_matt.shape)
```

```
(8210, 12)
```

[ ]:
```
print(df_matt.isnull().sum())
```

```
Num                          0
Time                         0
Day_of_week                  0
Age_band_of_driver           0
Sex_of_driver                0
Educational_level          476
Vehicle_driver_relation    340
Driving_experience         545
Type_of_vehicle            652
Owner_of_vehicle           321
Service_year_of_vehicle   2705
Accident_severity            0
dtype: int64
```

[ ]:
```
df_matt_cleaned = df_matt.copy()
```

[ ]:
```
# Replace the null values with the mode of each column
for column in df_matt_cleaned.columns:
    if df_matt_cleaned[column].dtype == 'object' or df_matt_cleaned[column].
 ↪dtype == 'category':
        mode_value = df_matt_cleaned[column].mode()[0]
        df_matt_cleaned[column].fillna(mode_value, inplace=True)
```

```
C:\Users\xxkjx\AppData\Local\Temp\ipykernel_46252\1655853051.py:5:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df_matt_cleaned[column].fillna(mode_value, inplace=True)
```

[ ]:
```
print(df_matt_cleaned.isnull().sum())
```

```
Num                        0
Time                       0
Day_of_week                0
Age_band_of_driver         0
Sex_of_driver              0
Educational_level          0
Vehicle_driver_relation    0
Driving_experience         0
```

```
Type_of_vehicle            0
Owner_of_vehicle           0
Service_year_of_vehicle    0
Accident_severity          0
dtype: int64
```

[ ]:
```
print(df_matt_cleaned.head())
```

```
   Num      Time Day_of_week Age_band_of_driver Sex_of_driver  \
0    1  17:02:00      Monday              18-30          Male
1    2  17:02:00      Monday              31-50          Male
2    3  17:02:00      Monday              18-30          Male
3    4   1:06:00      Sunday              18-30          Male
4    5   1:06:00      Sunday              18-30          Male

    Educational_level Vehicle_driver_relation Driving_experience  \
0   Above high school                Employee             1-2yr
1  Junior high school                Employee         Above 10yr
2  Junior high school                Employee             1-2yr
3  Junior high school                Employee            5-10yr
4  Junior high school                Employee             2-5yr

       Type_of_vehicle Owner_of_vehicle Service_year_of_vehicle  \
0           Automobile            Owner              Above 10yr
1  Public (> 45 seats)            Owner                 5-10yrs
2     Lorry (41?100Q)            Owner                 Unknown
3  Public (> 45 seats)     Governmental                 Unknown
4           Automobile            Owner                 5-10yrs

  Accident_severity
0     Slight Injury
1     Slight Injury
2    Serious Injury
3     Slight Injury
4     Slight Injury
```

[ ]:
```
for column in df_matt_cleaned.columns:
    if df_matt_cleaned[column].dtype == 'object':
        print(f"\nFrequency distribution for {column}:\n",␣
 ↪df_matt_cleaned[column].value_counts())
```

```
Frequency distribution for Time:
 Time
16:00:00    76
18:00:00    76
17:00:00    73
18:30:00    69
17:30:00    66
```

```
            ..
0:43:00      1
15:22:00     1
19:24:00     1
22:54:00     1
14:54:00     1
Name: count, Length: 1027, dtype: int64

Frequency distribution for Day_of_week:
 Day_of_week
Friday       1326
Thursday     1288
Wednesday    1250
Tuesday      1169
Monday       1139
Saturday     1127
Sunday        911
Name: count, dtype: int64

Frequency distribution for Age_band_of_driver:
 Age_band_of_driver
18-30       2728
31-50       2688
Unknown     1323
Over 51      933
Under 18     538
Name: count, dtype: int64

Frequency distribution for Sex_of_driver:
 Sex_of_driver
Male       7582
Female      462
Unknown     166
Name: count, dtype: int64

Frequency distribution for Educational_level:
 Educational_level
Junior high school    5540
Elementary school     1457
High school            754
Above high school      224
Writing & reading      138
Unknown                 65
Illiterate              32
Name: count, dtype: int64

Frequency distribution for Vehicle_driver_relation:
 Vehicle_driver_relation
```

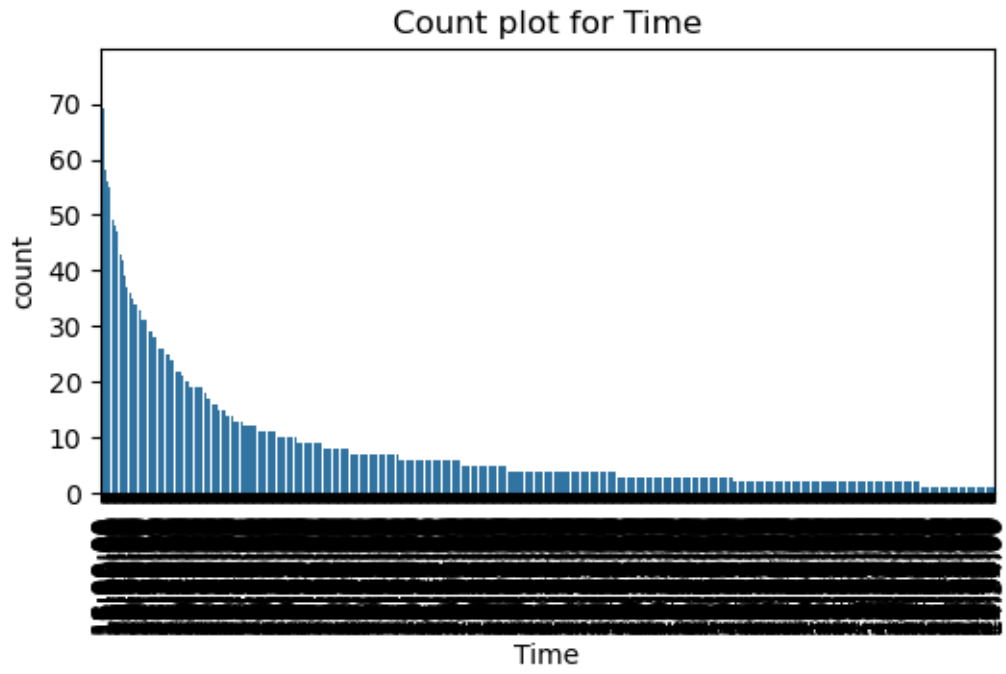
```
Employee    6713
Owner       1403
Other         80
Unknown       14
Name: count, dtype: int64

Frequency distribution for Driving_experience:
 Driving_experience
5-10yr        2813
2-5yr         1712
Above 10yr    1523
1-2yr         1147
Below 1yr      910
No Licence      81
unknown         24
Name: count, dtype: int64

Frequency distribution for Type_of_vehicle:
 Type_of_vehicle
Automobile              2782
Lorry (41?100Q)         1447
Other                    795
Pick up upto 10Q         533
Public (12 seats)        478
Stationwagen             457
Lorry (11?40Q)           365
Public (13?45 seats)     355
Public (> 45 seats)      275
Long lorry               256
Taxi                     180
Motorcycle               116
Special vehicle           56
Ridden horse              52
Turbo                     30
Bajaj                     19
Bicycle                   14
Name: count, dtype: int64

Frequency distribution for Owner_of_vehicle:
 Owner_of_vehicle
Owner           7293
Governmental     697
Organization     206
Other             14
Name: count, dtype: int64

Frequency distribution for Service_year_of_vehicle:
 Service_year_of_vehicle
```

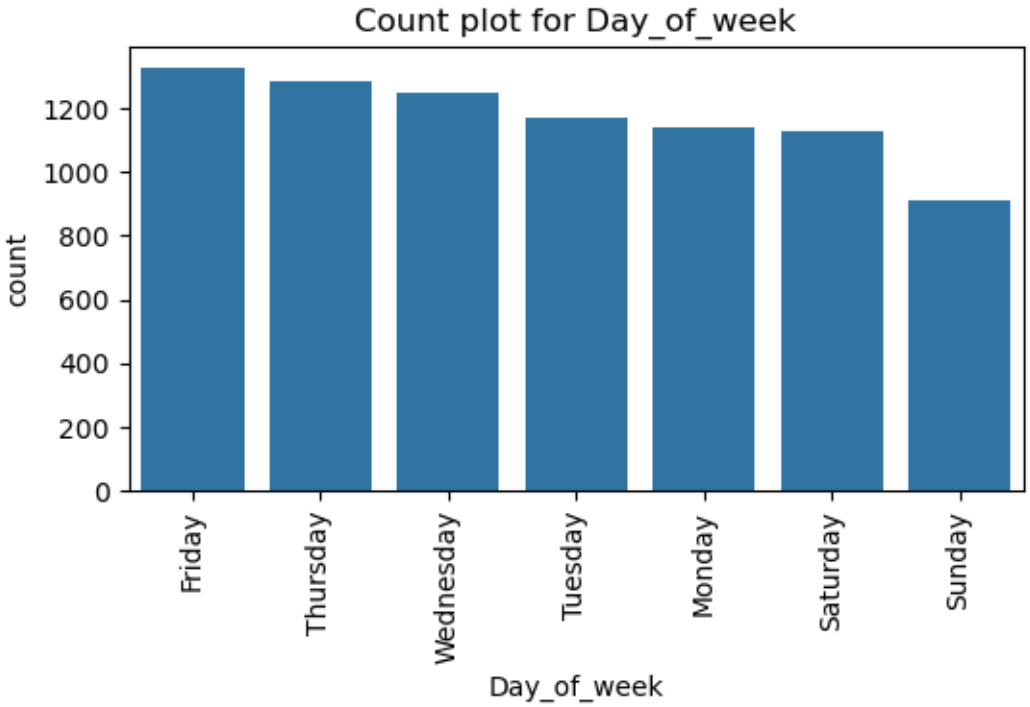
```
Unknown       4543
2-5yrs        1203
5-10yrs        867
Above 10yr     857
1-2yr          546
Below 1yr      194
Name: count, dtype: int64

Frequency distribution for Accident_severity:
 Accident_severity
Slight Injury     7082
Serious Injury    1046
Fatal injury        82
Name: count, dtype: int64
```

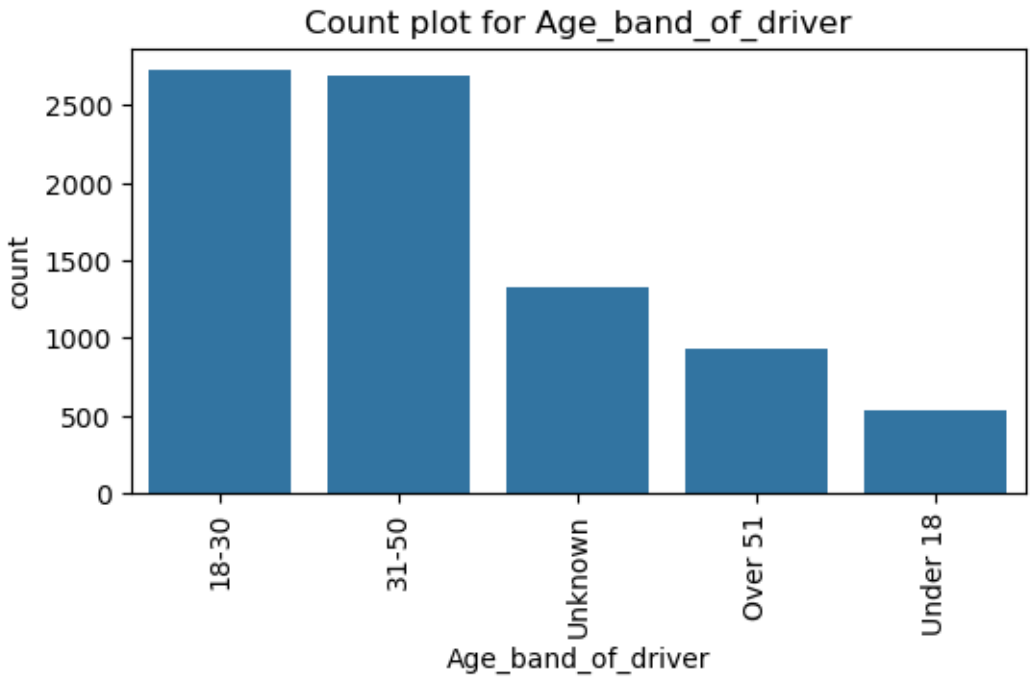
[ ]:
```
print("\nMode for num is ", df_matt_cleaned['Num'].mode()[0])
print("Median for num is ", df_matt_cleaned['Num'].median())

pq3,pq1 = np.percentile(df_matt_cleaned['Num'], [75,25])

iqr = pq3-pq1
print("IQR for Num is ", iqr)
```

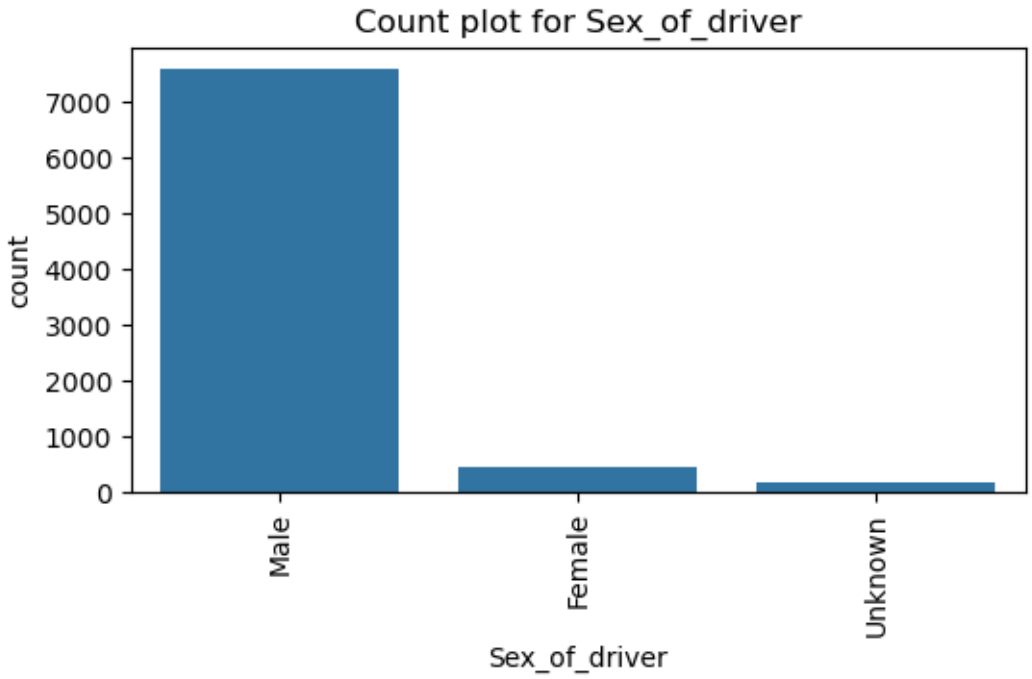
```
Mode for num is  1
Median for num is  4105.5
IQR for Num is  4104.5
```

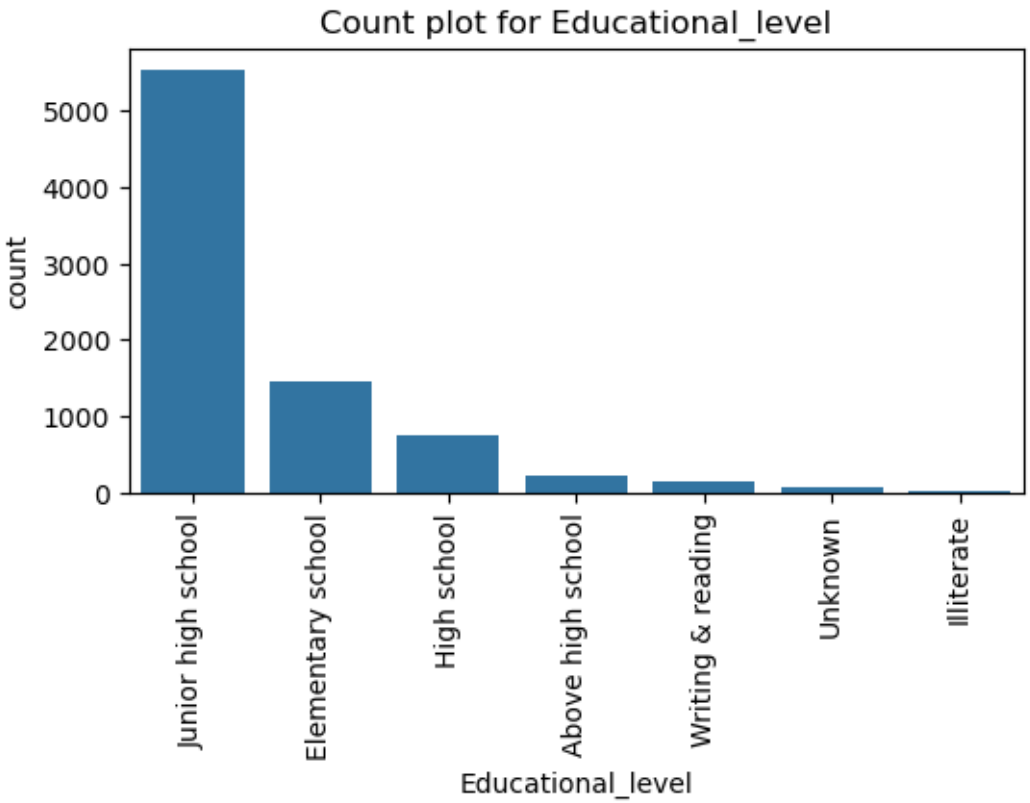
[ ]:
```
for column in df_matt_cleaned.columns:
    if df_matt_cleaned[column].dtype == 'object':
        plt.figure(figsize=(6, 3))
        sns.countplot(data=df_matt_cleaned, x=column,␣
 ↪order=df_matt_cleaned[column].value_counts().index)
        plt.xticks(rotation=90)
        plt.title(f"Count plot for {column}")
        plt.show()
```

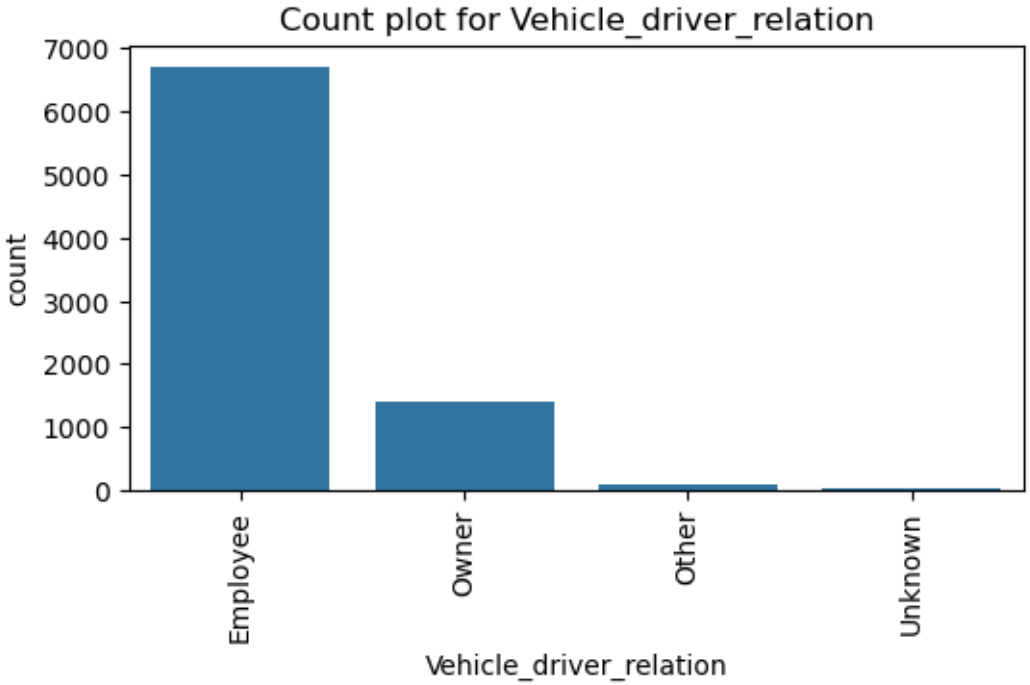

Count plot for Time
count
70
60
50
40
30
20
10
0
Time

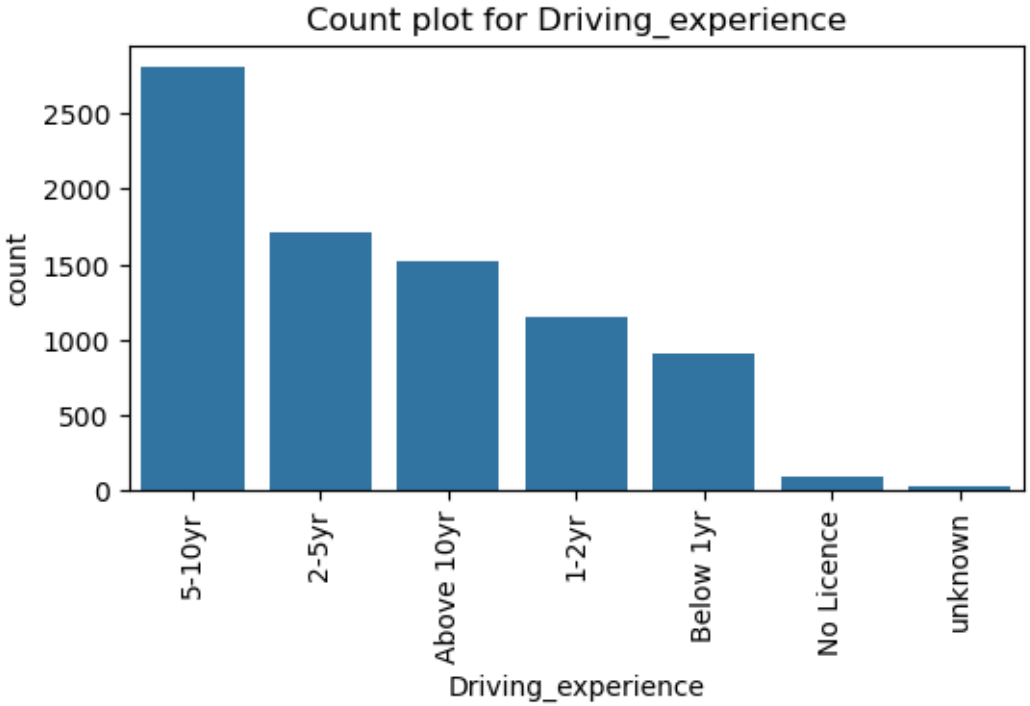

Count plot for Day_of_week
count
1200
1000
800
600
400
200
0
Friday
Thursday
Wednesday
Tuesday
Monday
Saturday
Sunday
Day_of_week

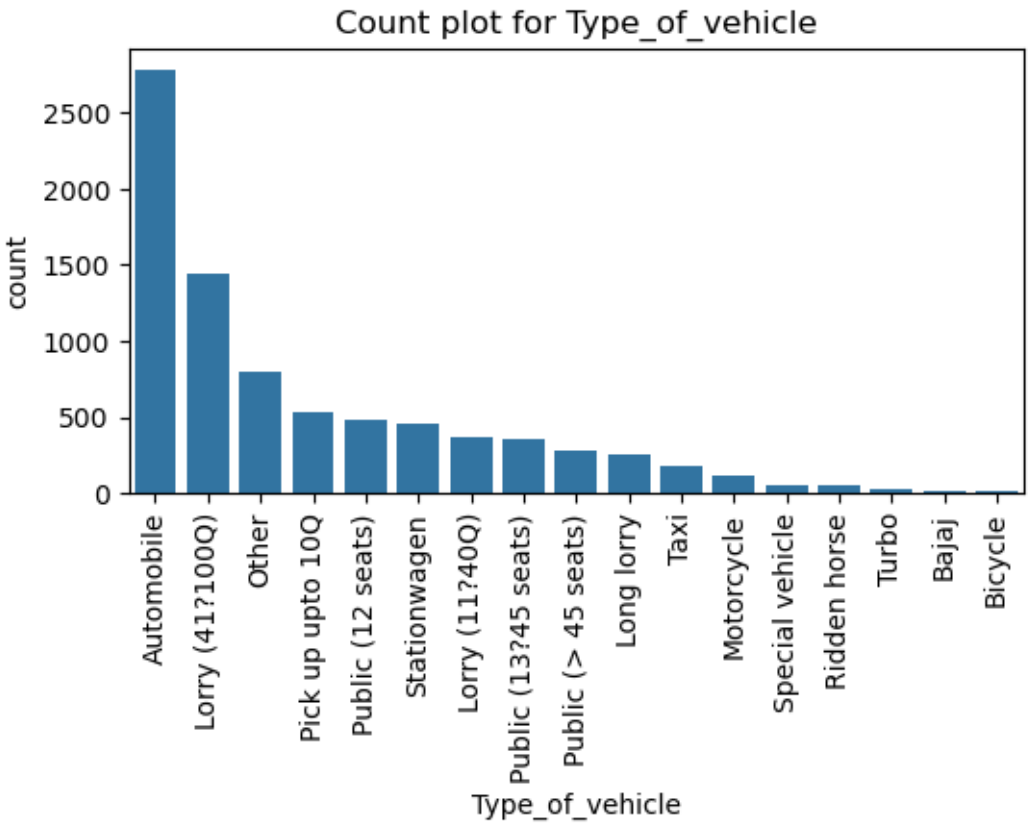

Count plot for Age_band_of_driver
count
2500
2000
1500
1000
500
0
18-30
31-50
Unknown
Over 51
Under 18
Age_band_of_driver

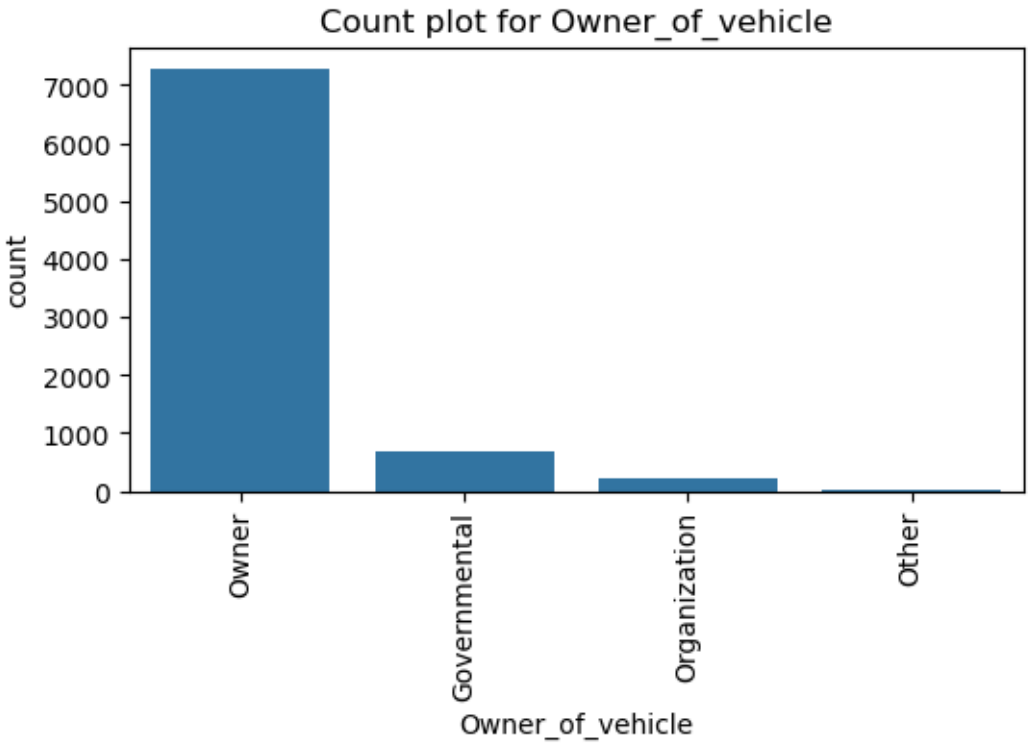

Count plot for Sex_of_driver
count
7000
6000
5000
4000
3000
2000
1000
0
Male
Female
Unknown
Sex_of_driver

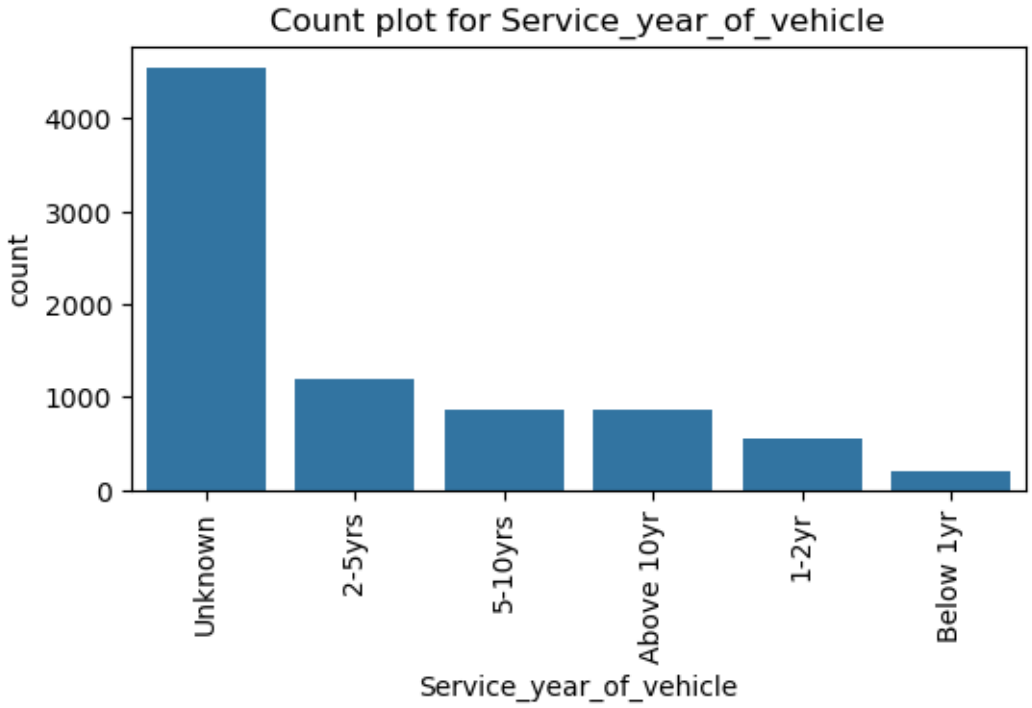

Count plot for Educational_level
count
5000
4000
3000
2000
1000
0
Junior high school
Elementary school
High school
Above high school
Writing & reading
Unknown
Illiterate
Educational_level

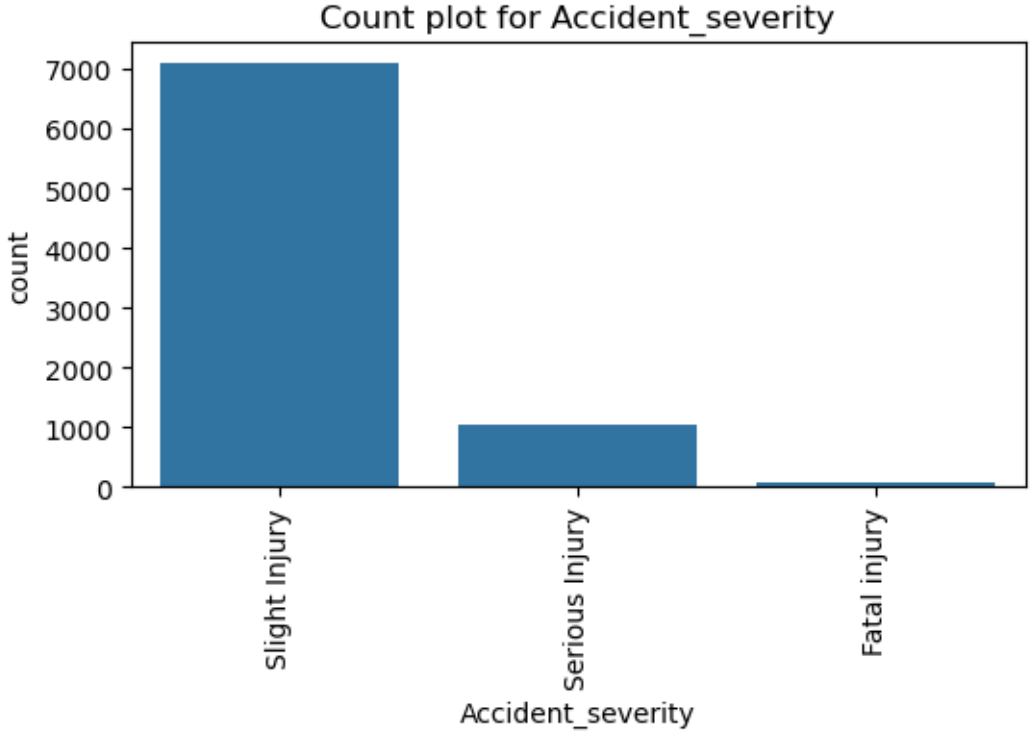

Count plot for Vehicle_driver_relation
count
7000
6000
5000
4000
3000
2000
1000
0
Employee
Owner
Other
Unknown
Vehicle_driver_relation


Count plot for Driving_experience
count
2500
2000
1500
1000
500
0
5-10yr
2-5yr
Above 10yr
1-2yr
Below 1yr
No Licence
unknown
Driving_experience

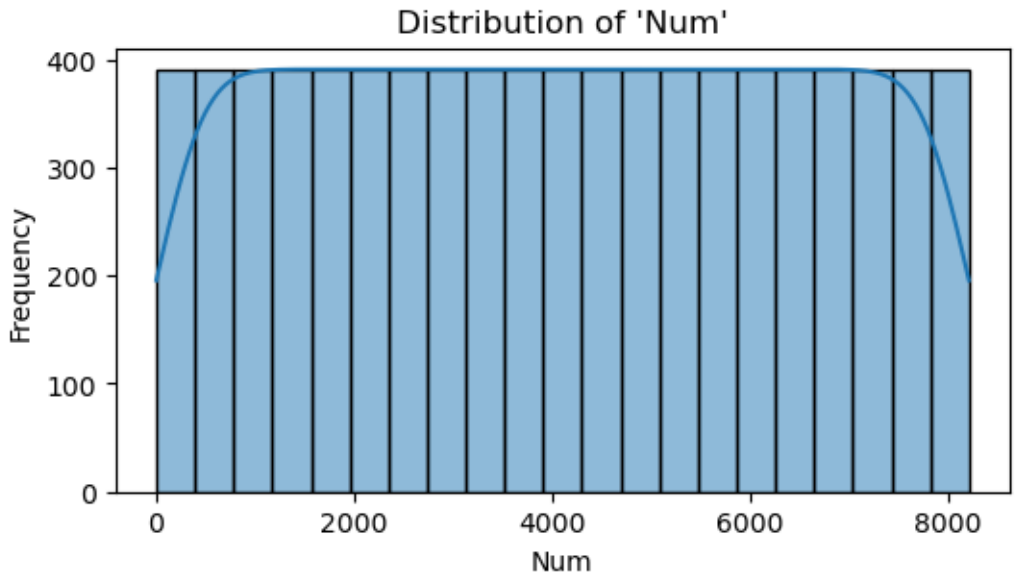

Count plot for Type_of_vehicle
count
2500
2000
1500
1000
500
0
Automobile
Lorry (41?100Q)
Other
Pick up upto 10Q
Public (12 seats)
Stationwagen
Lorry (11?40Q)
Public (13?45 seats)
Public (> 45 seats)
Long lorry
Taxi
Motorcycle
Special vehicle
Ridden horse
Turbo
Bajaj
Bicycle
Type_of_vehicle


Count plot for Owner_of_vehicle
count
7000
6000
5000
4000
3000
2000
1000
0
Owner
Governmental
Organization
Other
Owner_of_vehicle


Count plot for Service_year_of_vehicle
count
4000
3000
2000
1000
0
Unknown
2-5yrs
5-10yrs
Above 10yr
1-2yr
Below 1yr
Service_year_of_vehicle

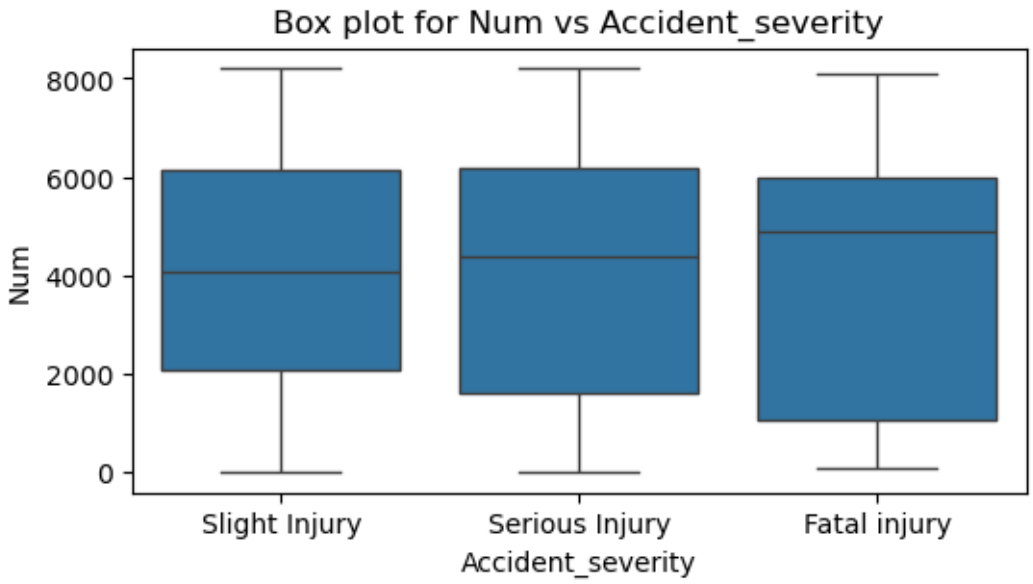

Count plot for Accident_severity
count
7000
6000
5000
4000
3000
2000
1000
0
Slight Injury
Serious Injury
Fatal injury
Accident_severity

[ ]:
```
if 'Num' in df_matt_cleaned.columns:
    plt.figure(figsize=(6, 3))
    sns.histplot(df_matt_cleaned['Num'], kde=True)
    plt.title("Distribution of 'Num'")
    plt.xlabel("Num")
    plt.ylabel("Frequency")
    plt.show()
```

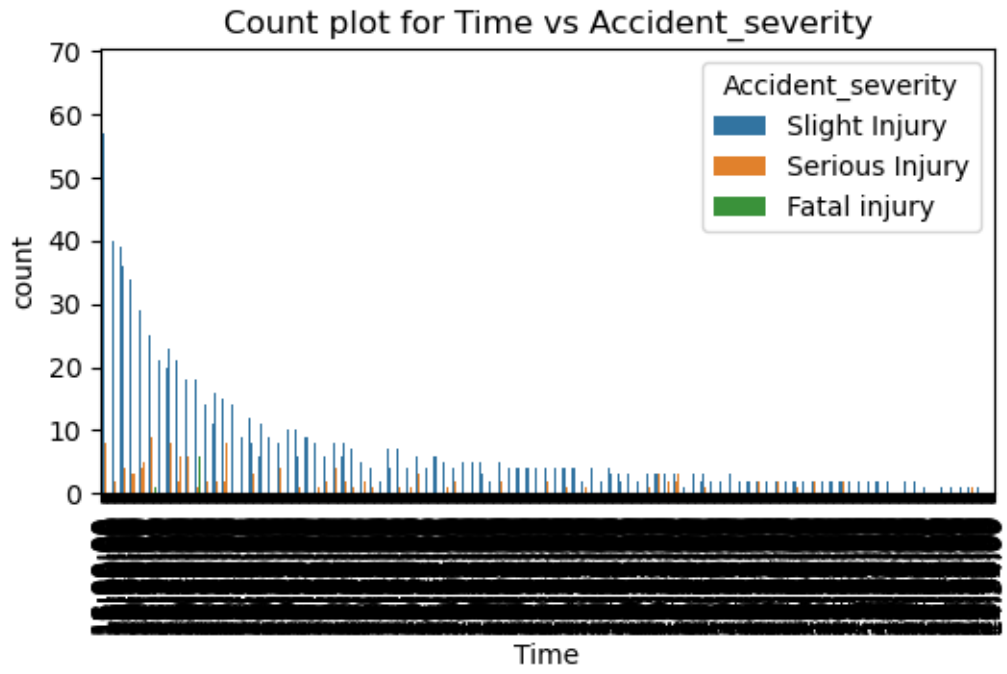


[ ]:
```
for column in df_matt_cleaned.columns:
    if column != 'Accident_severity':
        if df_matt_cleaned[column].dtype == 'object':
            plt.figure(figsize=(6, 3))
            sns.countplot(data=df_matt_cleaned, x=column,␣
 ↪hue='Accident_severity', order=df_matt_cleaned[column].value_counts().index)
            plt.xticks(rotation=90)
            plt.title(f"Count plot for {column} vs Accident_severity")
            plt.show()


            cross_tab = pd.crosstab(df_matt_cleaned[column],␣
 ↪df_matt_cleaned['Accident_severity'])
            chi2_stat, p_val, dof, ex = chi2_contingency(cross_tab)
            print(f"\nChi-Square Test results for {column} vs Accident_severity:
 ↪")
```

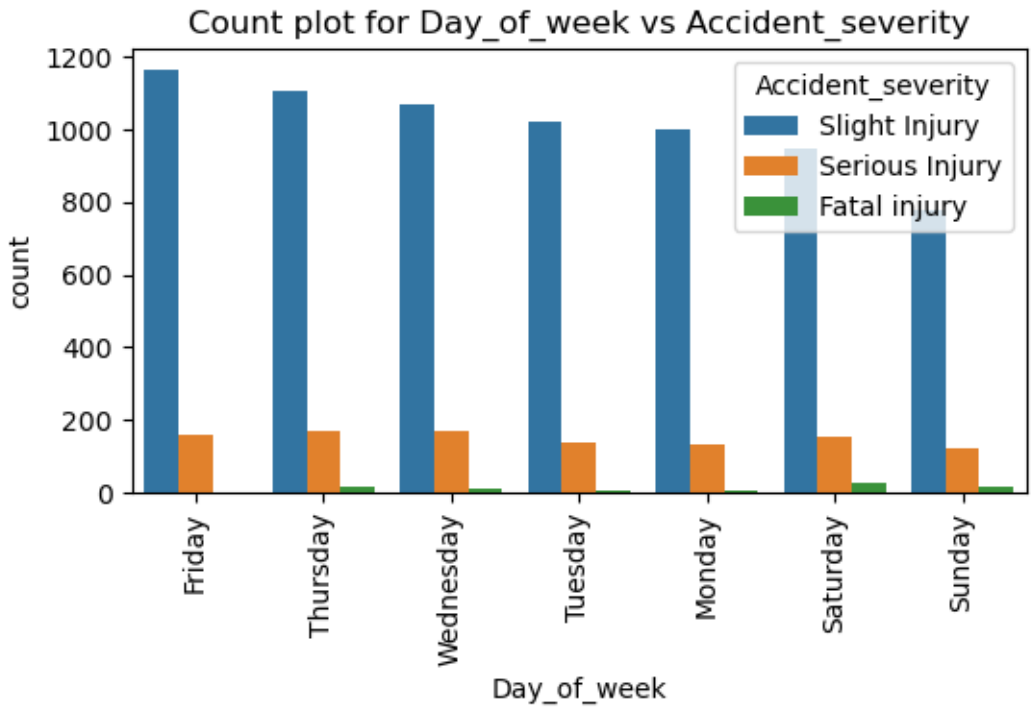
```
            print(f"Chi2 Stat: {chi2_stat}, p-value: {p_val}\n")

        elif pd.api.types.is_numeric_dtype(df_matt_cleaned[column]):
            plt.figure(figsize=(6, 3))
            sns.boxplot(data=df_matt_cleaned, x='Accident_severity', y=column)
            plt.title(f"Box plot for {column} vs Accident_severity")
            plt.show()


            print(f"\nSummary statistics for {column} by Accident_severity:")
            print(df_matt_cleaned.groupby('Accident_severity')[column].
↪describe(), "\n")
```



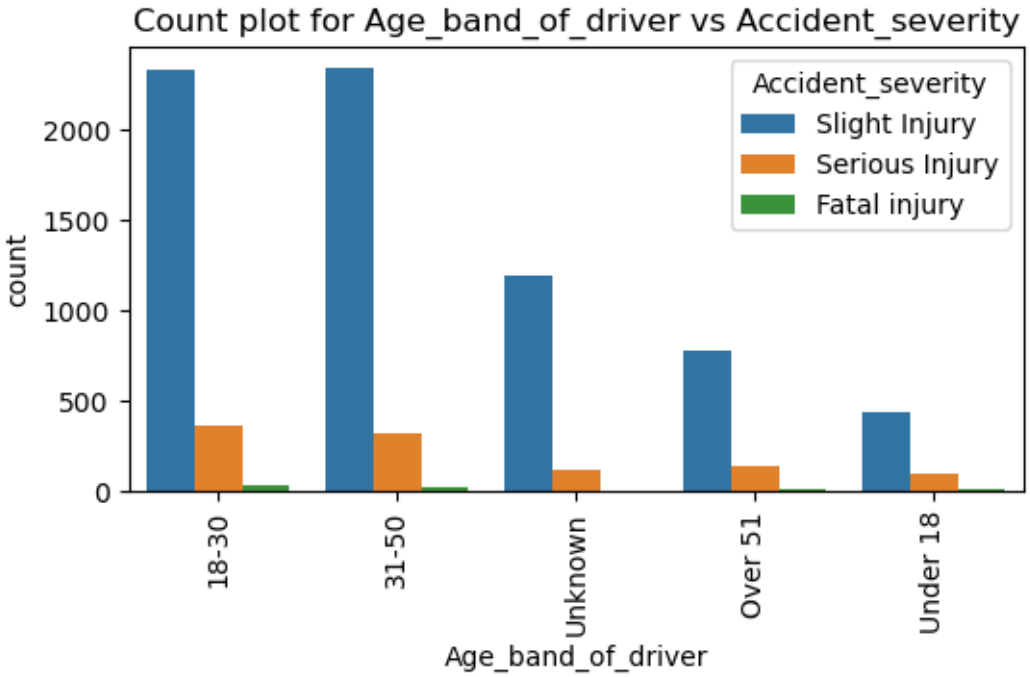

```
Summary statistics for Num by Accident_severity:
                     count         mean          std   min      25%     50%  \
Accident_severity
Fatal injury          82.0  4207.951220  2495.144756  90.0  1063.25  4885.5
Serious Injury      1046.0  4085.738050  2471.899158   3.0  1610.25  4376.0
Slight Injury       7082.0  4107.232561  2353.602660   1.0  2094.25  4061.5

                        75%     max
Accident_severity
Fatal injury        5968.75  8114.0
Serious Injury      6176.75  8197.0
Slight Injury       6154.75  8210.0
```



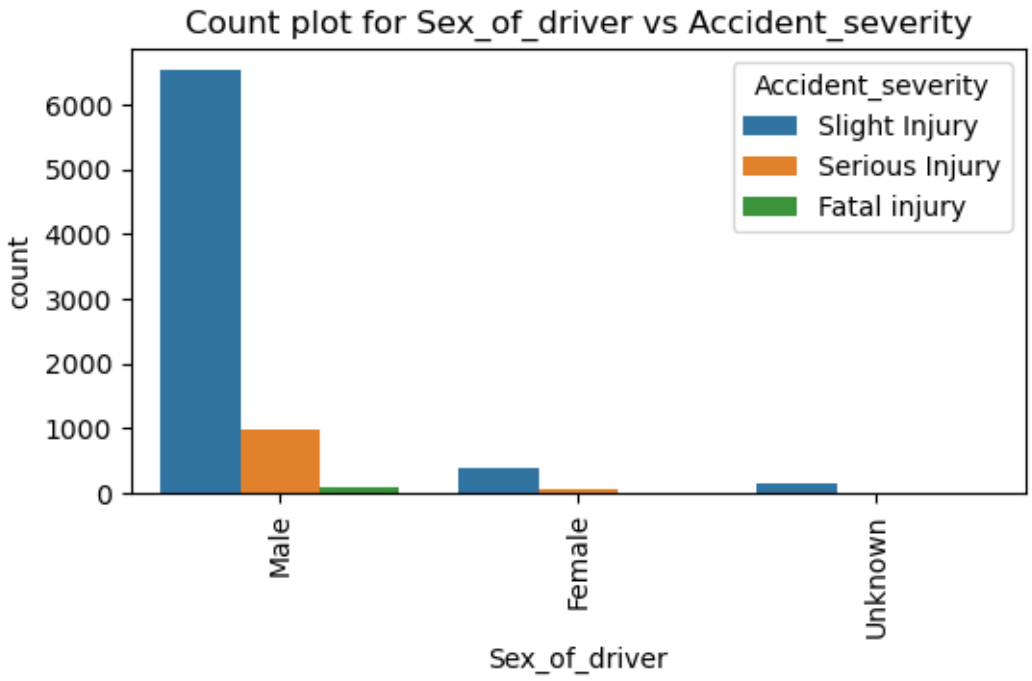

```
Chi-Square Test results for Time vs Accident_severity:
Chi2 Stat: 5890.165646544385, p-value: 0.0
```



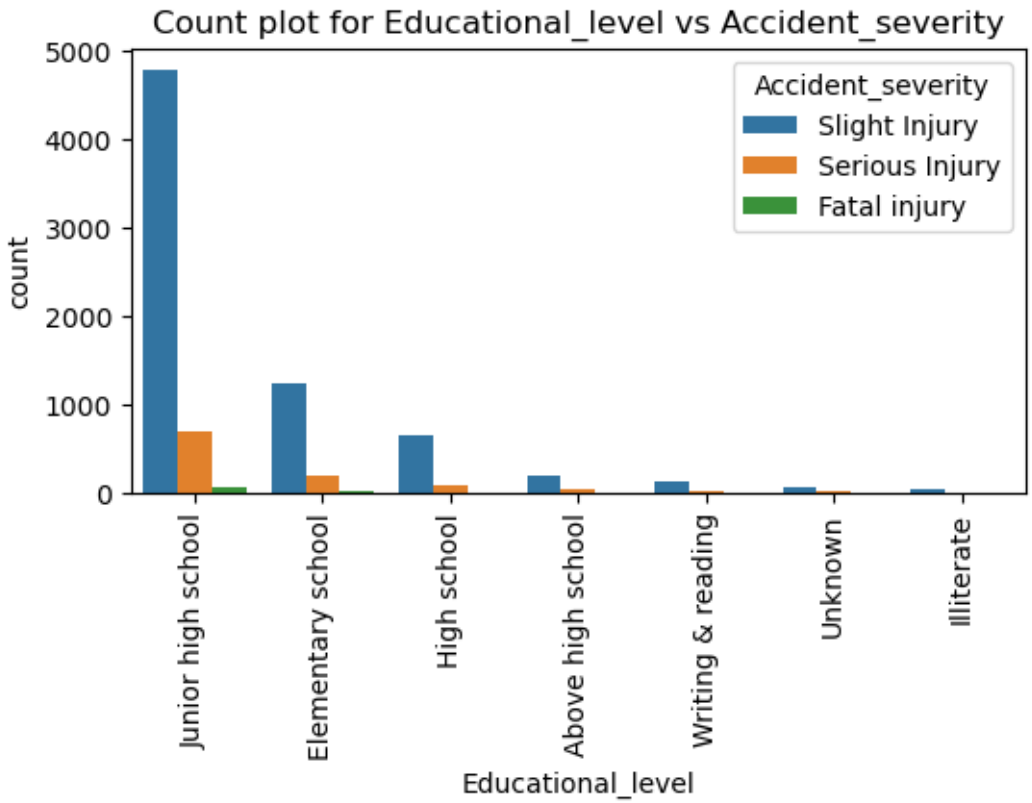
```
Chi-Square Test results for Day_of_week vs Accident_severity:
Chi2 Stat: 40.01753132688368, p-value: 7.142862409371859e-05
```



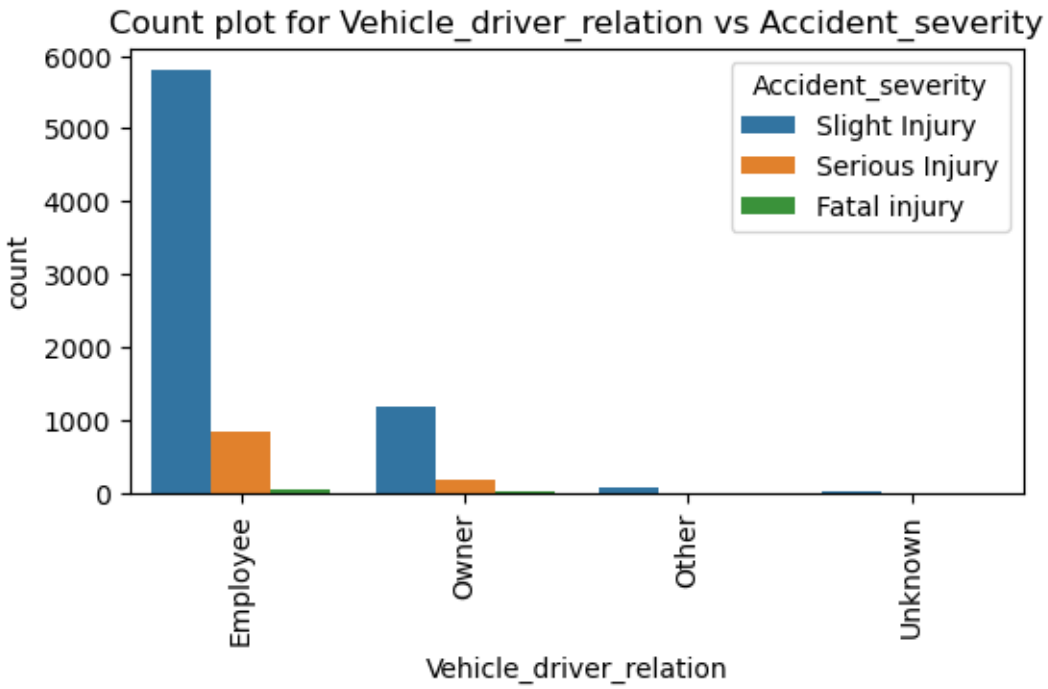

```
Chi-Square Test results for Age_band_of_driver vs Accident_severity:
Chi2 Stat: 44.406175437055005, p-value: 4.767492656555021e-07
```



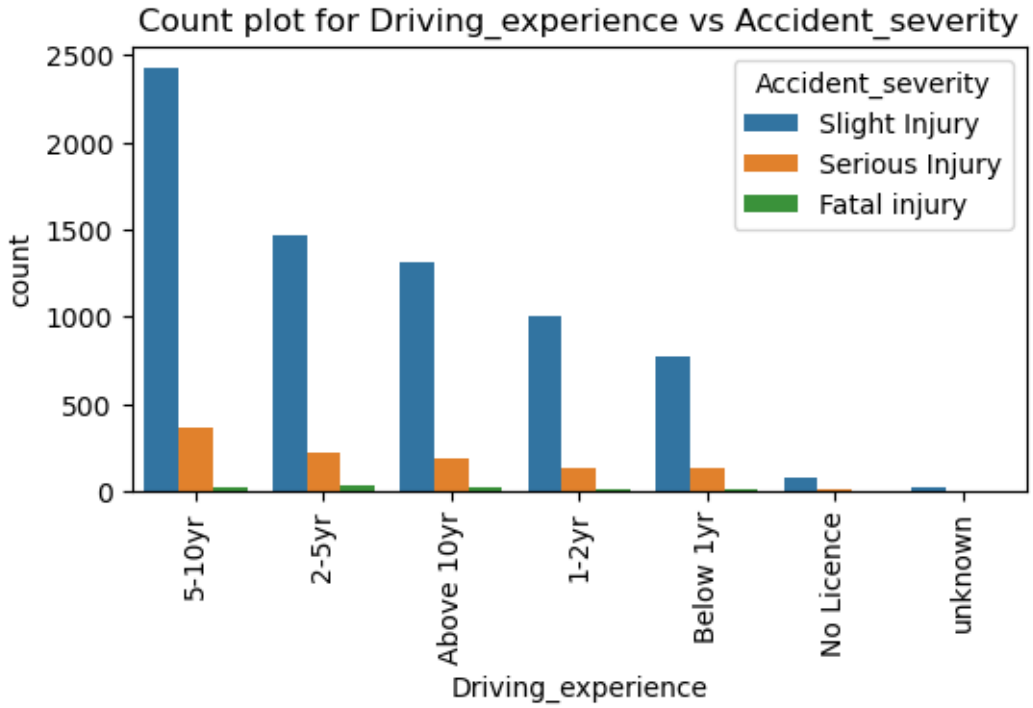

```
Chi-Square Test results for Sex_of_driver vs Accident_severity:
Chi2 Stat: 2.0066350826267967, p-value: 0.7345384293334768
```



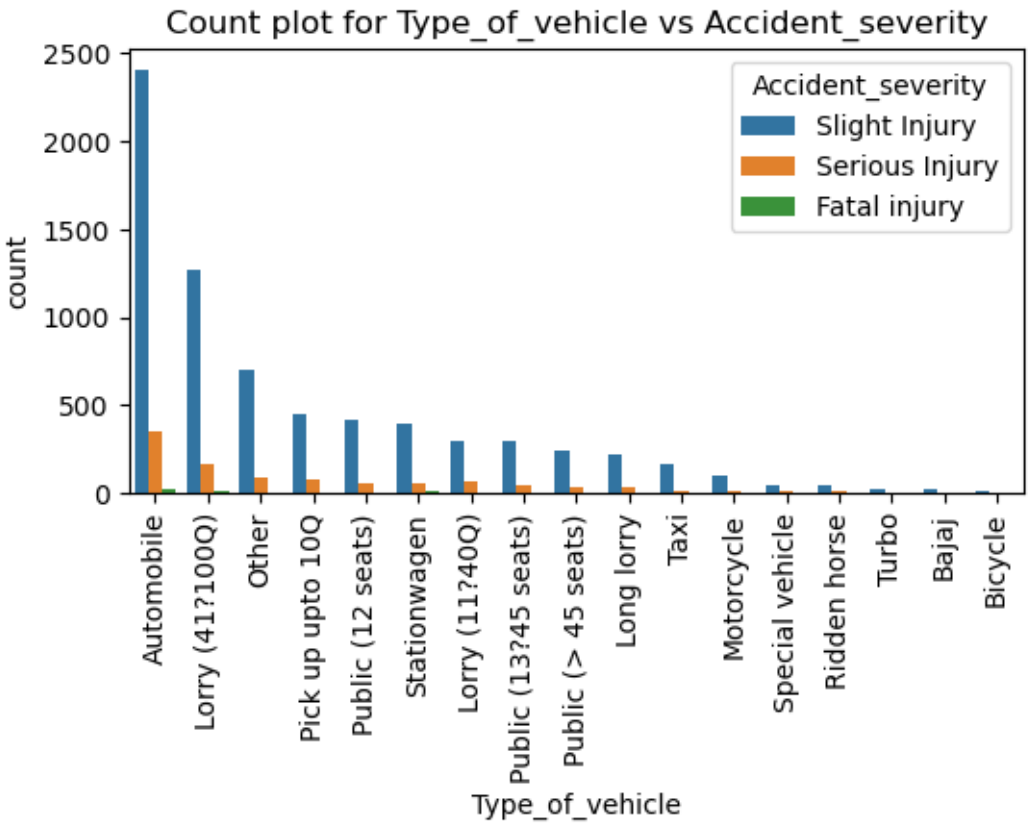

```
Chi-Square Test results for Educational_level vs Accident_severity:
Chi2 Stat: 11.167676540591525, p-value: 0.5146061174442822
```



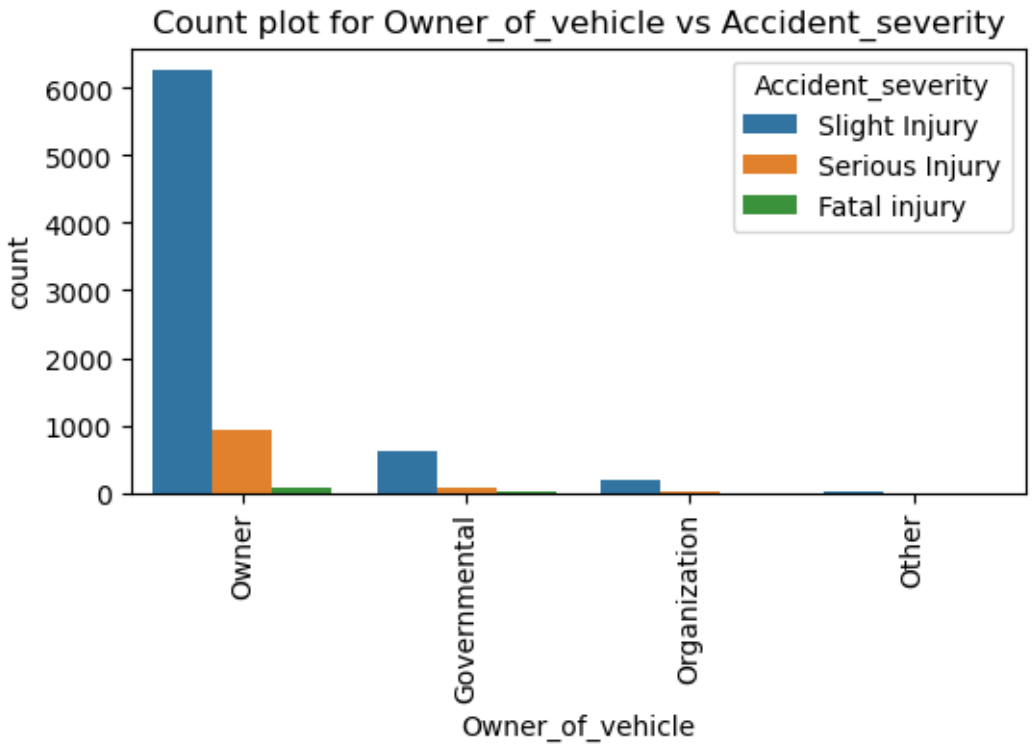

```
Chi-Square Test results for Vehicle_driver_relation vs Accident_severity:
Chi2 Stat: 11.078032481406902, p-value: 0.08599447681287752
```



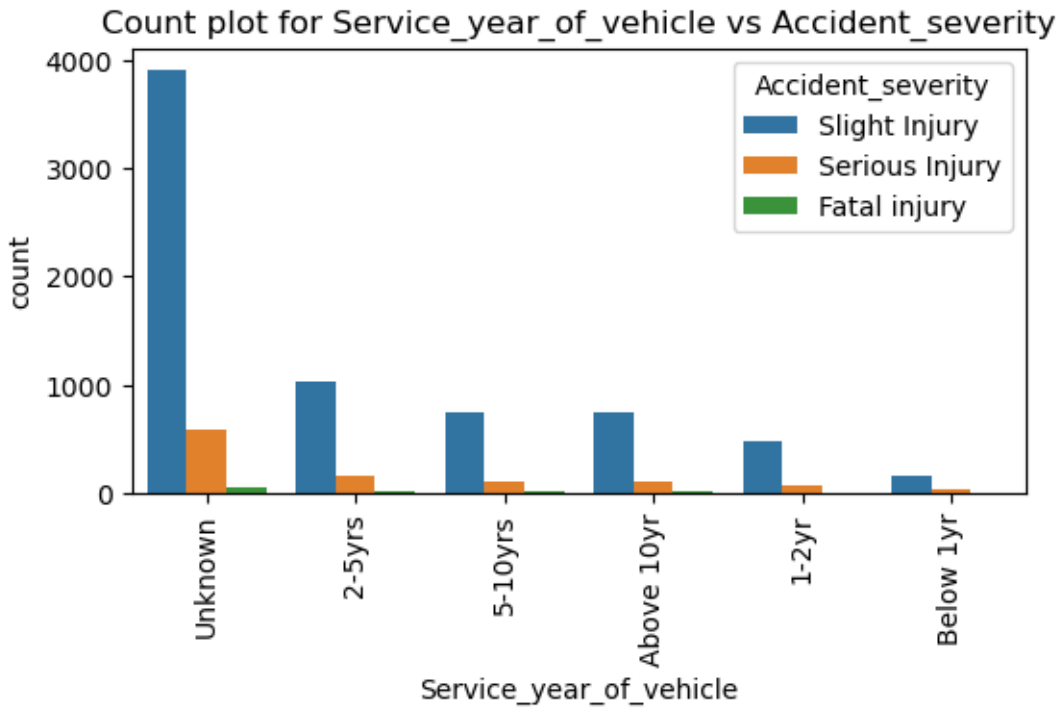
```
Chi-Square Test results for Driving_experience vs Accident_severity:
Chi2 Stat: 14.517552444887768, p-value: 0.2688854071998473
```




```
Chi-Square Test results for Type_of_vehicle vs Accident_severity:
Chi2 Stat: 43.06394541090455, p-value: 0.09160985848944223
```




```
Chi-Square Test results for Owner_of_vehicle vs Accident_severity:
Chi2 Stat: 4.8690355699772985, p-value: 0.560716870417195
```

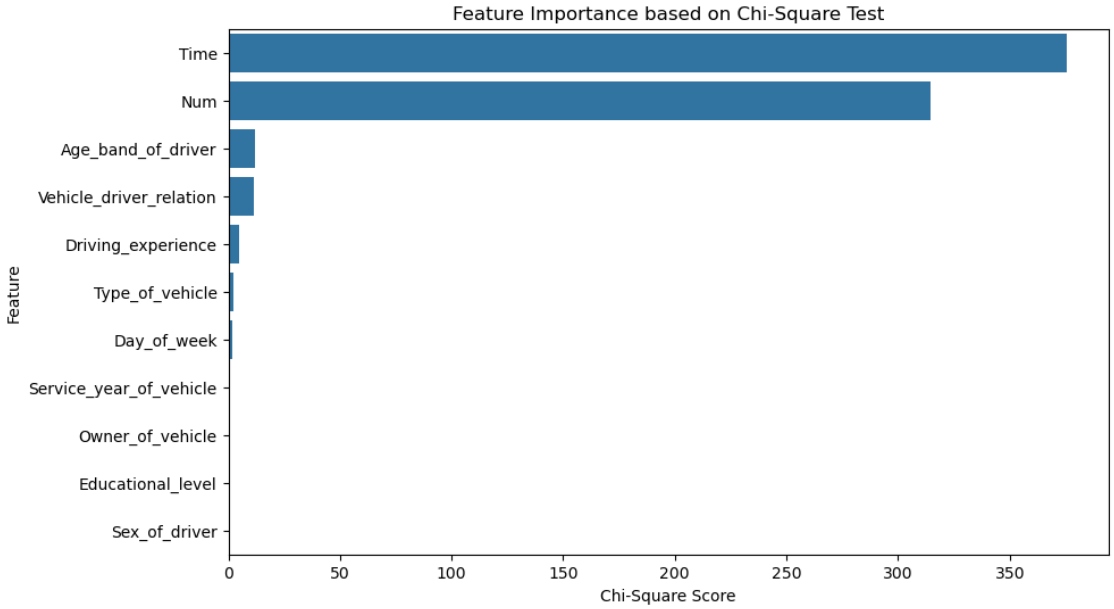



```
Chi-Square Test results for Service_year_of_vehicle vs Accident_severity:
Chi2 Stat: 3.3552187111115326, p-value: 0.971787163822214
```

[ ]:
```
label_encoder = LabelEncoder()


categorical_columns = ['Time', 'Day_of_week', 'Age_band_of_driver',␣
 ↪'Sex_of_driver',
                       'Educational_level', 'Vehicle_driver_relation',␣
 ↪'Driving_experience',
                       'Type_of_vehicle', 'Owner_of_vehicle',␣
 ↪'Service_year_of_vehicle']

for col in categorical_columns:
    df_matt_cleaned[col] = label_encoder.fit_transform(df_matt_cleaned[col])


df_matt_cleaned['Accident_severity'] = label_encoder.
 ↪fit_transform(df_matt_cleaned['Accident_severity'])


X = df_matt_cleaned.drop(columns=['Accident_severity'])
```

```
y = df_matt_cleaned['Accident_severity']


best_features = SelectKBest(score_func=chi2, k='all')
fit = best_features.fit(X, y)


feature_scores = pd.DataFrame({'Feature': X.columns, 'Chi-Square Score': fit.
 ↪scores_, 'P-Value': fit.pvalues_})


print(feature_scores.sort_values(by='Chi-Square Score', ascending=False))


plt.figure(figsize=(10, 6))
sns.barplot(x='Chi-Square Score', y='Feature', data=feature_scores.
 ↪sort_values(by='Chi-Square Score', ascending=False))
plt.title('Feature Importance based on Chi-Square Test')
plt.show()
```

	Feature	Chi-Square Score	P-Value
1	Time	375.352440	3.113532e-82
0	Num	314.322372	5.568833e-69
3	Age_band_of_driver	11.921414	2.578089e-03
6	Vehicle_driver_relation	11.637317	2.971588e-03
7	Driving_experience	4.727723	9.405632e-02
8	Type_of_vehicle	2.038530	3.608601e-01
2	Day_of_week	1.872532	3.920893e-01
10	Service_year_of_vehicle	0.839878	6.570869e-01
9	Owner_of_vehicle	0.572299	7.511504e-01
5	Educational_level	0.276298	8.709687e-01
4	Sex_of_driver	0.017023	9.915244e-01



1.1.2 Dataset 2

[ ]:
```
target = 'Accident_severity'
my_list = df.columns.tolist()
set2 = my_list[11:21]
set2.append(target)
print(len(set2),set2)
df1 = df[set2]
df1.head()
```

```
11 ['Defect_of_vehicle', 'Area_accident_occured', 'Lanes_or_Medians',
'Road_allignment', 'Types_of_Junction', 'Road_surface_type',
'Road_surface_conditions', 'Light_conditions', 'Weather_conditions',
'Type_of_collision', 'Accident_severity']
```

[ ]:

	Defect_of_vehicle	Area_accident_occured	Lanes_or_Medians	Road_allignment	Types_of_Junction
0	No defect	Residential areas	NaN	Tangent road with flat terrain	No junction
1	No defect	Office areas	Undivided Two way	Tangent road with flat terrain	No junction
2	No defect	Recreational areas	other	NaN	No junction
3	No defect	Office areas	other	Tangent road with mild grade and flat terrain	Y Shape
4	No defect	Industrial areas	other		

```
4                 Tangent road with flat terrain           Y Shape

  Road_surface_type Road_surface_conditions       Light_conditions  \
0     Asphalt roads                      Dry               Daylight
1     Asphalt roads                      Dry               Daylight
2     Asphalt roads                      Dry               Daylight
3       Earth roads                      Dry  Darkness - lights lit
4     Asphalt roads                      Dry  Darkness - lights lit

  Weather_conditions                          Type_of_collision  \
0             Normal  Collision with roadside-parked vehicles
1             Normal            Vehicle with vehicle collision
2             Normal           Collision with roadside objects
3             Normal            Vehicle with vehicle collision
4             Normal            Vehicle with vehicle collision

  Accident_severity
0     Slight Injury
1     Slight Injury
2    Serious Injury
3     Slight Injury
4     Slight Injury
```

[ ]:
```
#Initial Inspection
print(df1.isnull().sum())

print(df1.info())

print(df1.describe(include='object'))
```

```
Defect_of_vehicle          2985
Area_accident_occured       160
Lanes_or_Medians            267
Road_allignment             102
Types_of_Junction             0
Road_surface_type           115
Road_surface_conditions       0
Light_conditions              0
Weather_conditions            0
Type_of_collision           100
Accident_severity             0
dtype: int64
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8210 entries, 0 to 8209
Data columns (total 11 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Defect_of_vehicle        5225 non-null   object
```

```
 1   Area_accident_occured    8050 non-null   object
 2   Lanes_or_Medians         7943 non-null   object
 3   Road_allignment          8108 non-null   object
 4   Types_of_Junction        8210 non-null   object
 5   Road_surface_type        8095 non-null   object
 6   Road_surface_conditions  8210 non-null   object
 7   Light_conditions         8210 non-null   object
 8   Weather_conditions       8210 non-null   object
 9   Type_of_collision        8110 non-null   object
 10  Accident_severity        8210 non-null   object
dtypes: object(11)
memory usage: 705.7+ KB
None
       Defect_of_vehicle Area_accident_occured  \
count               5225                  8050
unique                 3                    14
top           No defect                 Other
freq                5143                  2511

                                      Lanes_or_Medians  \
count                                             7943
unique                                               7
top     Two-way (divided with broken lines road marking)
freq                                              2898

                      Road_allignment Types_of_Junction Road_surface_type  \
count                            8108              8210               8095
unique                              9                 8                  5
top     Tangent road with flat terrain           Y Shape     Asphalt roads
freq                             6942              3118               7539

       Road_surface_conditions Light_conditions Weather_conditions  \
count                     8210             8210               8210
unique                       4                4                  9
top                        Dry         Daylight             Normal
freq                      6515             5924               6782

                    Type_of_collision Accident_severity
count                            8110              8210
unique                             10                 3
top     Vehicle with vehicle collision     Slight Injury
freq                             5821              7082
```

[ ]:
```
#Define categorical features
```

```
categorical_features = ['Defect_of_vehicle', 'Area_accident_occured',␣
 ↪'Lanes_or_Medians', 'Road_allignment', 'Types_of_Junction',␣
 ↪'Road_surface_type', 'Road_surface_conditions', 'Light_conditions',␣
 ↪'Weather_conditions', 'Type_of_collision']
```

[ ]:
```
#Data quality report
data_quality_report_cat(df1, categorical_features)
```

```
Data Quality Report for Categorical Features
============================================
Stats
-----
                   Feature  Count  Missing  % Missing  Cardinality
0        Defect_of_vehicle   5225     2985      57.13            4
1    Area_accident_occured   8050      160       1.99           15
2         Lanes_or_Medians   7943      267       3.36            8
3          Road_allignment   8108      102       1.26           10
4        Types_of_Junction   8210        0       0.00            8
5        Road_surface_type   8095      115       1.42            6
6  Road_surface_conditions   8210        0       0.00            4
7         Light_conditions   8210        0       0.00            4
8       Weather_conditions   8210        0       0.00            9
9        Type_of_collision   8110      100       1.23           11


Mode 1
------
                   Feature                                        Mode 1  \
0        Defect_of_vehicle                                     No defect
1    Area_accident_occured                                         Other
2         Lanes_or_Medians  Two-way (divided with broken lines road marking)
3          Road_allignment                  Tangent road with flat terrain
4        Types_of_Junction                                       Y Shape
5        Road_surface_type                                  Asphalt roads
6  Road_surface_conditions                                            Dry
7         Light_conditions                                       Daylight
8       Weather_conditions                                         Normal
9        Type_of_collision                  Vehicle with vehicle collision

   Mode 1 Freq.  Mode 1 %
0          5143     98.43
1          2511     31.19
2          2898     36.48
3          6942     85.62
4          3118     37.98
5          7539     93.13
6          6515     79.35
7          5924     72.16
```

```
8          6782     82.61
9          5821     71.78


Mode 2
------
                   Feature                                        Mode 2  \
0        Defect_of_vehicle                                             7
1    Area_accident_occured                                   Office areas
2         Lanes_or_Medians                                Undivided Two way
3          Road_allignment  Tangent road with mild grade and flat terrain
4        Types_of_Junction                                      No junction
5        Road_surface_type                                      Earth roads
6  Road_surface_conditions                                      Wet or damp
7         Light_conditions                          Darkness - lights lit
8       Weather_conditions                                         Raining
9        Type_of_collision                Collision with roadside objects

   Mode 2 Freq.  Mode 2 %
0            55      1.05
1          2323     28.86
2          2530     31.85
3           354      4.37
4          2924     35.62
5           236      2.92
6          1660     20.22
7          2171     26.44
8           778      9.48
9          1187     14.64


Descriptive Stats
-----------------
                        count unique  \
Defect_of_vehicle        5225      3
Area_accident_occured    8050     14
Lanes_or_Medians         7943      7
Road_allignment          8108      9
Types_of_Junction        8210      8
Road_surface_type        8095      5
Road_surface_conditions  8210      4
Light_conditions         8210      4
Weather_conditions       8210      9
Type_of_collision        8110     10

                                                          top  \
Defect_of_vehicle                                  No defect
Area_accident_occured                                  Other
```

```
Lanes_or_Medians         Two-way (divided with broken lines road marking)
Road_allignment                          Tangent road with flat terrain
Types_of_Junction                                                Y Shape
Road_surface_type                                          Asphalt roads
Road_surface_conditions                                              Dry
Light_conditions                                                 Daylight
Weather_conditions                                                 Normal
Type_of_collision                        Vehicle with vehicle collision

                         freq
Defect_of_vehicle        5143
Area_accident_occured    2511
Lanes_or_Medians         2898
Road_allignment          6942
Types_of_Junction        3118
Road_surface_type        7539
Road_surface_conditions  6515
Light_conditions         5924
Weather_conditions       6782
Type_of_collision        5821
```

Cleaning the data

[ ]:
```
#Data cleaning
for col in df1.columns:
    df1[col] = df1[col].fillna(df1[col].mode()[0])
```

```
C:\Users\xxkjx\AppData\Local\Temp\ipykernel_46252\4285372874.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1[col] = df1[col].fillna(df1[col].mode()[0])
```

[ ]:
```
#Double-check clean data
data_quality_report_cat(df1, categorical_features)
```

```
Data Quality Report for Categorical Features
============================================
Stats
-----
                   Feature  Count  Missing  % Missing  Cardinality
0        Defect_of_vehicle   8210        0        0.0            3
1    Area_accident_occured   8210        0        0.0           14
2         Lanes_or_Medians   8210        0        0.0            7
3          Road_allignment   8210        0        0.0            9
4        Types_of_Junction   8210        0        0.0            8
```

```
5        Road_surface_type   8210        0        0.0            5
6  Road_surface_conditions   8210        0        0.0            4
7         Light_conditions   8210        0        0.0            4
8       Weather_conditions   8210        0        0.0            9
9        Type_of_collision   8210        0        0.0           10


Mode 1
------
                   Feature                                          Mode 1  \
0        Defect_of_vehicle                                       No defect
1    Area_accident_occured                                           Other
2         Lanes_or_Medians  Two-way (divided with broken lines road marking)
3          Road_allignment                    Tangent road with flat terrain
4        Types_of_Junction                                         Y Shape
5        Road_surface_type                                    Asphalt roads
6  Road_surface_conditions                                             Dry
7         Light_conditions                                        Daylight
8       Weather_conditions                                          Normal
9        Type_of_collision                  Vehicle with vehicle collision

   Mode 1 Freq.  Mode 1 %
0          8128     99.00
1          2671     32.53
2          3165     38.55
3          7044     85.80
4          3118     37.98
5          7654     93.23
6          6515     79.35
7          5924     72.16
8          6782     82.61
9          5921     72.12


Mode 2
------
                   Feature                                          Mode 2  \
0        Defect_of_vehicle                                               7
1    Area_accident_occured                                     Office areas
2         Lanes_or_Medians                                Undivided Two way
3          Road_allignment  Tangent road with mild grade and flat terrain
4        Types_of_Junction                                      No junction
5        Road_surface_type                                      Earth roads
6  Road_surface_conditions                                      Wet or damp
7         Light_conditions                          Darkness - lights lit
8       Weather_conditions                                         Raining
9        Type_of_collision                Collision with roadside objects
```

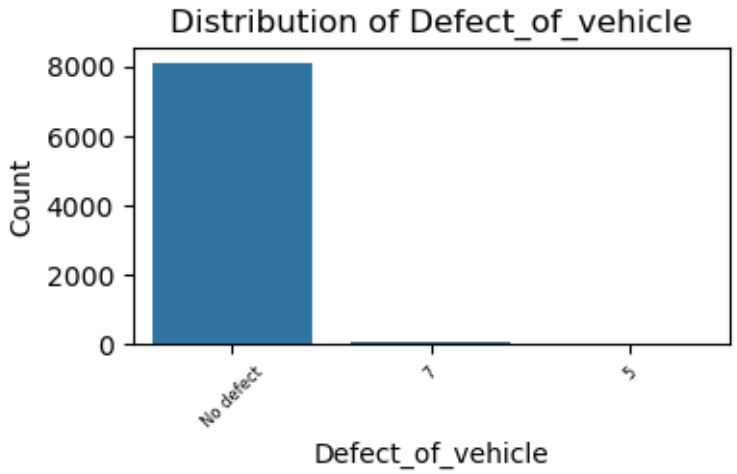
	Mode 2 Freq.	Mode 2 %
0	55	0.67
1	2323	28.29
2	2530	30.82
3	354	4.31
4	2924	35.62
5	236	2.87
6	1660	20.22
7	2171	26.44
8	778	9.48
9	1187	14.46

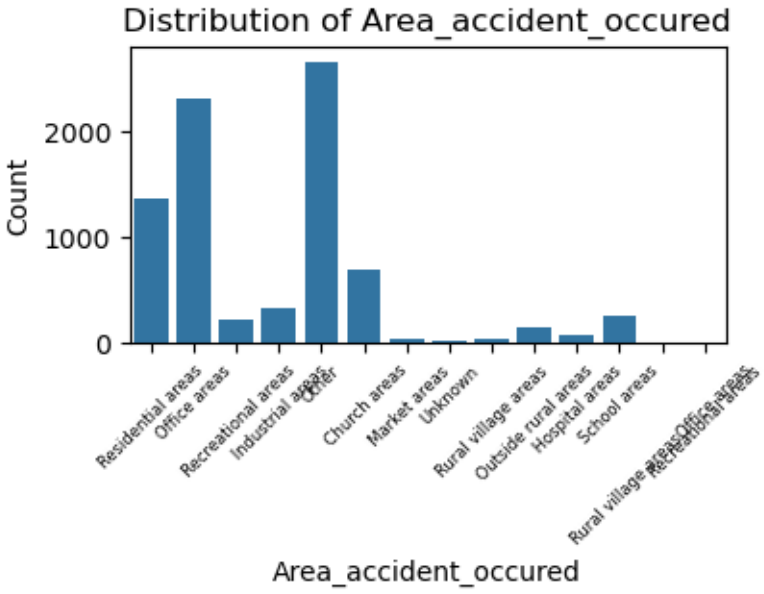
Descriptive Stats

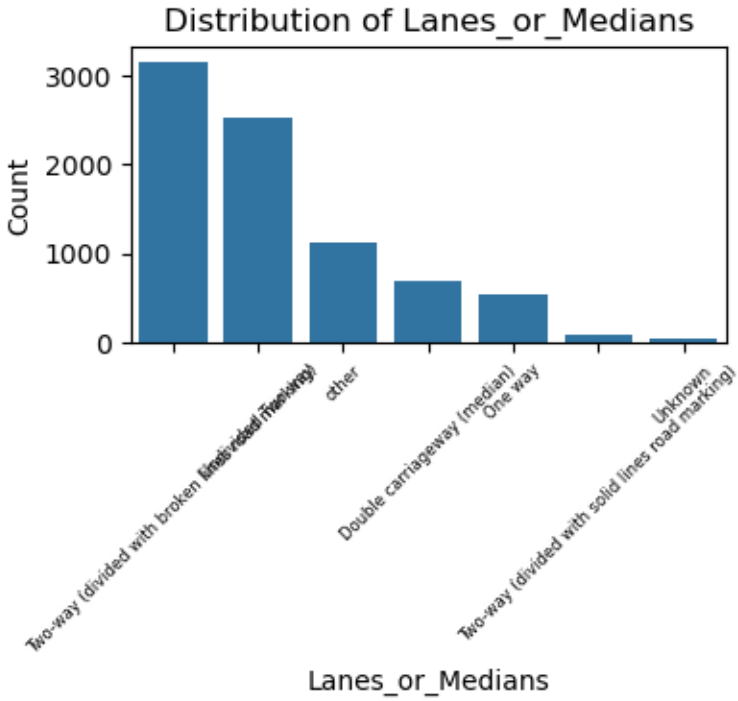
	count	unique	top	freq
Defect_of_vehicle	8210	3	No defect	8128
Area_accident_occured	8210	14	Other	2671
Lanes_or_Medians	8210	7	Two-way (divided with broken lines road marking)	3165
Road_allignment	8210	9	Tangent road with flat terrain	7044
Types_of_Junction	8210	8	Y Shape	3118
Road_surface_type	8210	5	Asphalt roads	7654
Road_surface_conditions	8210	4	Dry	6515
Light_conditions	8210	4	Daylight	5924
Weather_conditions	8210	9	Normal	
Type_of_collision	8210	10	Vehicle with vehicle collision	

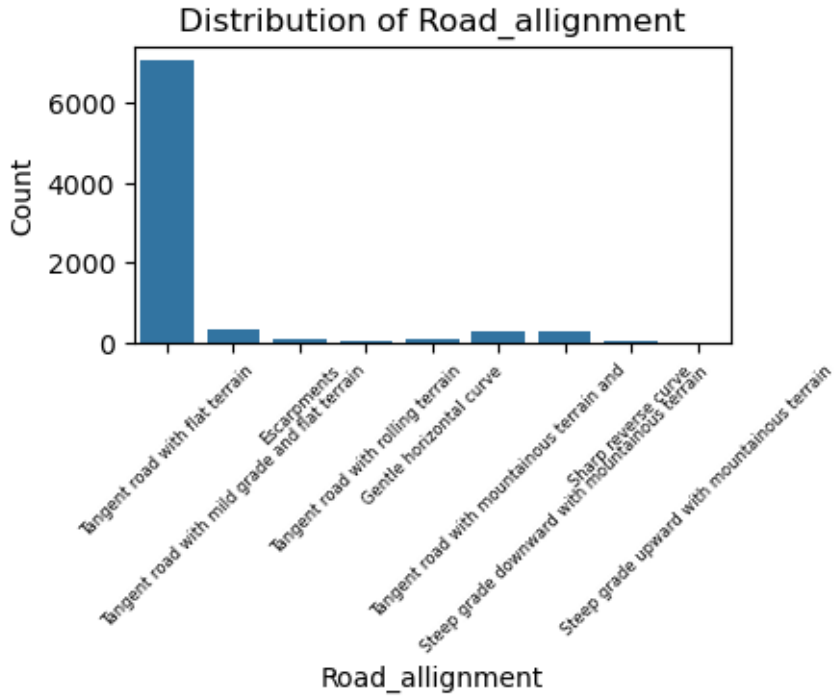
```
Weather_conditions       6782
Type_of_collision        5921
```

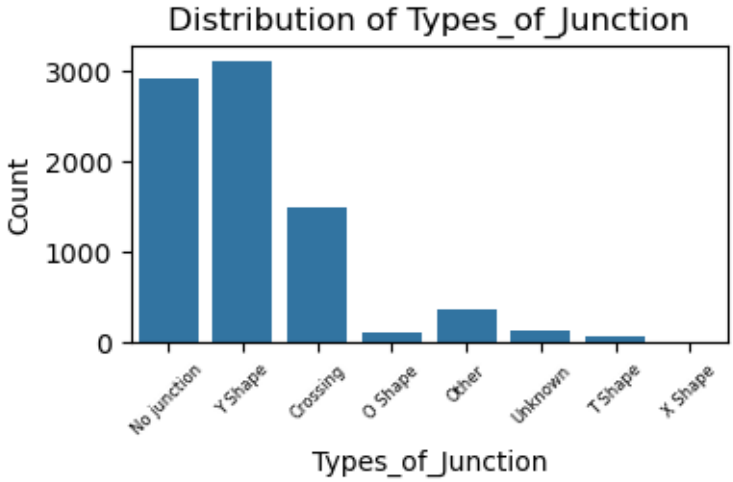
```
[ ]: #Visualize distributions
for feature in categorical_features:
    plt.figure(figsize=(4, 2))
    sns.countplot(data=df1, x=feature)
    plt.title('Distribution of ' + feature)
    plt.xlabel(feature)
    plt.xticks(rotation=45, fontsize=6)
    plt.ylabel('Count')
    plt.show()
```

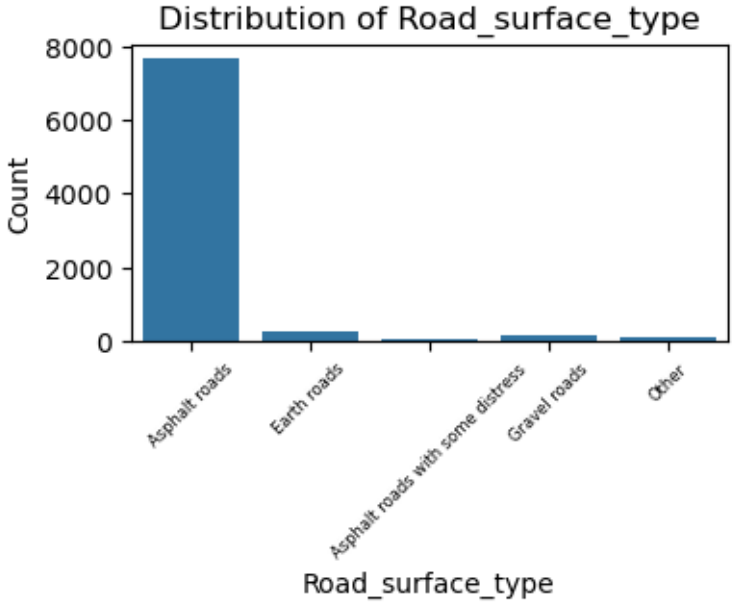


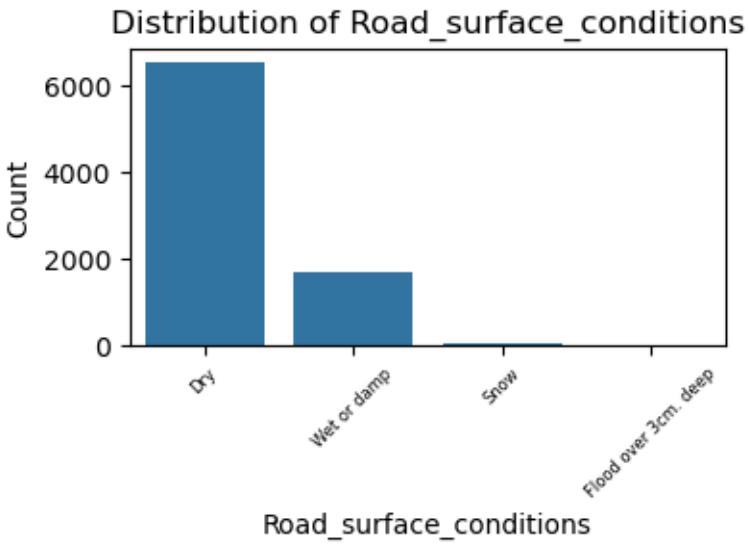


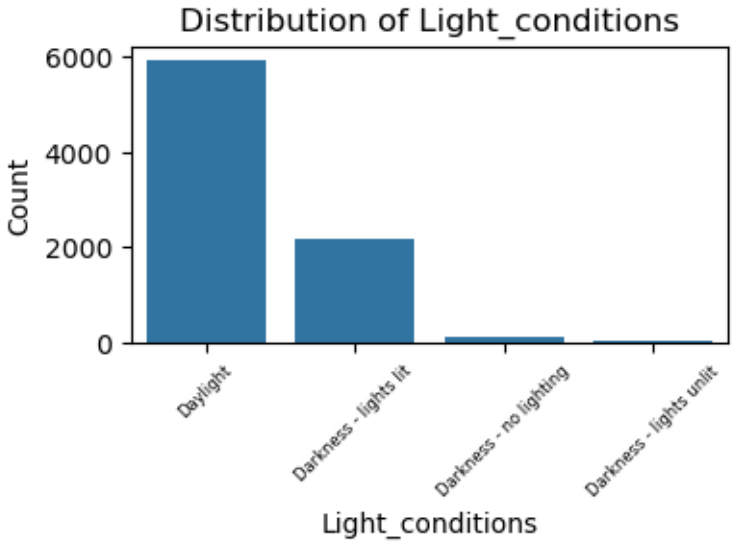

Distribution of Lanes_or_Medians
Count
3000
2000
1000
0
other
Double carriageway (median)
One way
Unknown
Two-way (divided with solid lines road marking)
Lanes_or_Medians

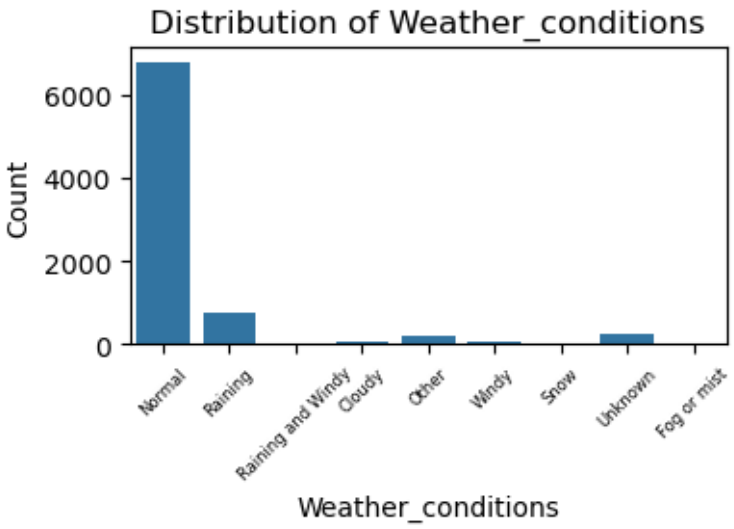

Distribution of Road_allignment
Count
6000
4000
2000
0
Tangent road with flat terrain
Escarpments
Tangent road with mild grade and flat terrain
Tangent road with rolling terrain
Gentle horizontal curve
Tangent road with mountainous terrain and
Sharp reverse curve
Steep grade downward with mountainous terrain
Steep grade upward with mountainous terrain
Road_allignment

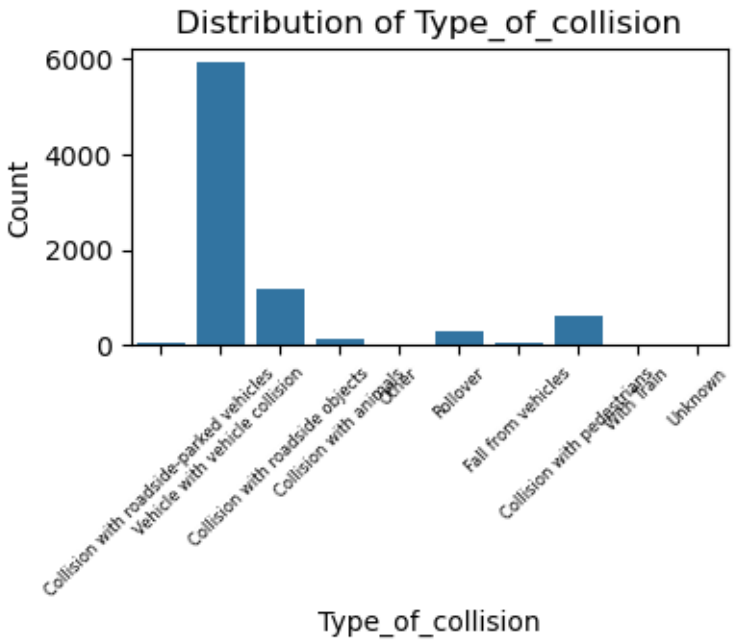

Distribution of Types_of_Junction
Count
3000
2000
1000
0
No junction
Y Shape
Crossing
O Shape
Other
Unknown
T Shape
X Shape
Types_of_Junction


Distribution of Road_surface_type
Count
8000
6000
4000
2000
0
Asphalt roads
Earth roads
Asphalt roads with some distress
Gravel roads
Other
Road_surface_type


Distribution of Road_surface_conditions
Count
6000
4000
2000
0
Dry
Wet or damp
Snow
Flood over 3cm. deep
Road_surface_conditions


Distribution of Light_conditions
Count
6000
4000
2000
0
Daylight
Darkness - lights lit
Darkness - no lighting
Darkness - lights unlit
Light_conditions





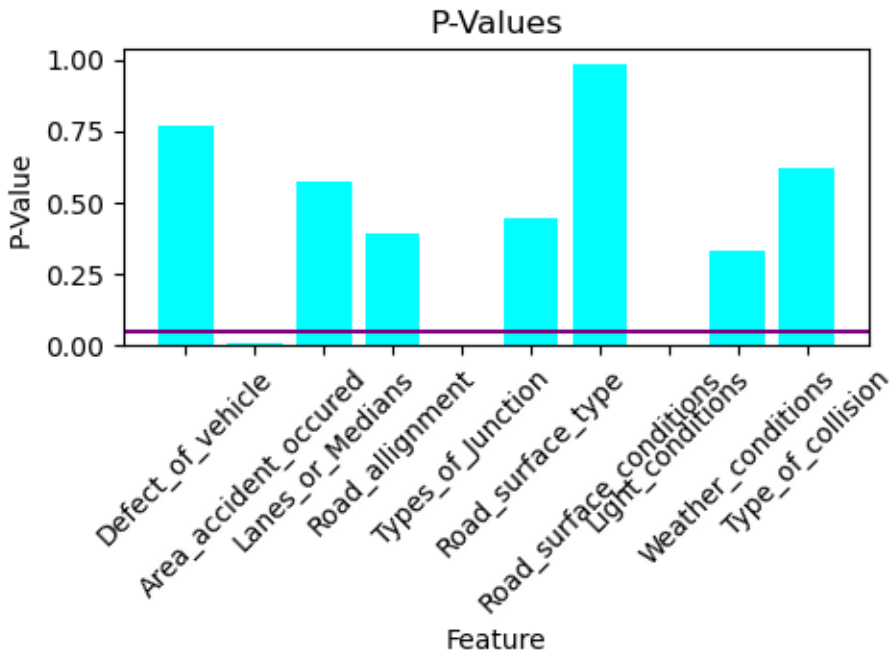
[ ]:
```
#Define function to perform chi-square test
def perform_chi2_test(df, feature1, target):
    contingency_table = pd.crosstab(df[feature1], df[target])
    chi2, p_value, dof, expected = chi2_contingency(contingency_table)
    return chi2, p_value
```

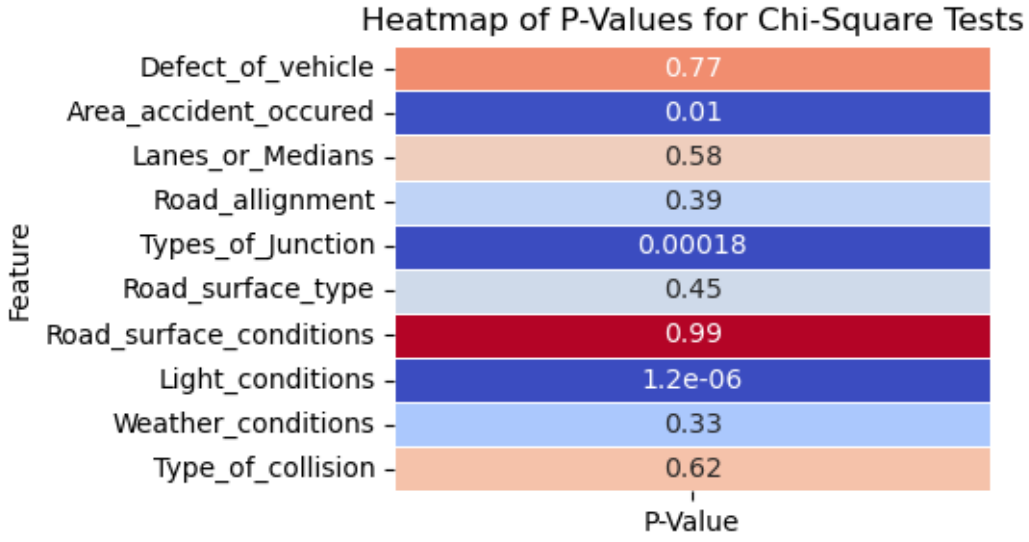
[ ]:
```
chi_square_results = []
chi_square_values = []
p_values = []

#Perform chi-squared test
for column in categorical_features:
    chi2, p_value = perform_chi2_test(df, column, 'Accident_severity')
    chi_square_results.append((column, chi2, p_value))
    chi_square_values.append(chi2)
    p_values.append(p_value)
    print(f"Chi-squared test for {column}:")
    print(f"Chi-squared value: {chi2}")
    print(f"P-value: {p_value}")
    print(f"Degrees of Freedom: {dof:.02f}\n")
```

```
Chi-squared test for Defect_of_vehicle:
Chi-squared value: 1.815558107965728
P-value: 0.7696347037158582
Degrees of Freedom: 10.00

Chi-squared test for Area_accident_occured:
Chi-squared value: 45.60537733466555
P-value: 0.010093215397474282
Degrees of Freedom: 10.00

Chi-squared test for Lanes_or_Medians:
Chi-squared value: 10.46542304855572
P-value: 0.5751993036180403
Degrees of Freedom: 10.00

Chi-squared test for Road_allignment:
Chi-squared value: 16.91403666623602
P-value: 0.39118073901723627
Degrees of Freedom: 10.00

Chi-squared test for Types_of_Junction:
Chi-squared value: 41.002640045213184
P-value: 0.0001777029383432938
Degrees of Freedom: 10.00

Chi-squared test for Road_surface_type:
Chi-squared value: 7.878070949618699
P-value: 0.4454703827962947
Degrees of Freedom: 10.00

Chi-squared test for Road_surface_conditions:
Chi-squared value: 0.963243628552523
P-value: 0.9869673605112773
```

```
Degrees of Freedom: 10.00

Chi-squared test for Light_conditions:
Chi-squared value: 37.936244258774565
P-value: 1.156063053914734e-06
Degrees of Freedom: 10.00

Chi-squared test for Weather_conditions:
Chi-squared value: 17.843412656490138
P-value: 0.3331458243936798
Degrees of Freedom: 10.00

Chi-squared test for Type_of_collision:
Chi-squared value: 15.586953591596322
P-value: 0.6213492686462301
Degrees of Freedom: 10.00
```

[ ]:
```
# Visualize P-Values
plt.figure(figsize=(5, 2))
plt.bar(categorical_features, p_values, color='cyan')
#Significance level:
plt.axhline(0.05, color='purple')
plt.title('P-Values')
plt.xlabel('Feature')
plt.ylabel('P-Value')
plt.xticks(rotation=45)
plt.show()

# Heatmap for P-Values
p_values_df = pd.DataFrame({'Feature': categorical_features, 'P-Value':␣
 ↪p_values})
p_values_df.set_index('Feature', inplace=True)

plt.figure(figsize=(4, 3))
sns.heatmap(p_values_df, annot=True, cmap='coolwarm', cbar=False, linewidths=0.
 ↪5)
plt.title('Heatmap of P-Values for Chi-Square Tests')
plt.show()
```





1.1.3 Dataset 3

[ ]:
```
features = my_list[21:33]
df1 = df[features]

df1.head()
```

[ ]:
```
   Number_of_vehicles_involved  Number_of_casualties Vehicle_movement  \
0                            2                     2   Going straight
1                            2                     2   Going straight
2                            2                     2   Going straight
3                            2                     2   Going straight
4                            2                     2   Going straight

    Casualty_class Sex_of_casualty Age_band_of_casualty Casualty_severity  \
0               na              na                   na                na
1               na              na                   na                na
2  Driver or rider            Male                31-50                 3
3       Pedestrian          Female                18-30                 3
4               na              na                   na                na

  Work_of_casuality Fitness_of_casuality Pedestrian_movement  \
0               NaN                  NaN    Not a Pedestrian
1               NaN                  NaN    Not a Pedestrian
2            Driver                  NaN    Not a Pedestrian
3            Driver               Normal    Not a Pedestrian
4               NaN                  NaN    Not a Pedestrian

            Cause_of_accident Accident_severity
0             Moving Backward     Slight Injury
1                  Overtaking     Slight Injury
2   Changing lane to the left    Serious Injury
3  Changing lane to the right     Slight Injury
4                  Overtaking     Slight Injury
```

1.2 Data Understanding

[ ]:
```
# Set continuous and categorical features
disc_feat = ['Number_of_vehicles_involved', 'Number_of_casualties']
cat_feat = ['Vehicle_movement', 'Casualty_class', 'Sex_of_casualty',␣
 ↪'Age_band_of_casualty', 'Casualty_severity', 'Work_of_casuality',␣
 ↪'Fitness_of_casuality', 'Pedestrian_movement', 'Cause_of_accident',␣
 ↪'Accident_severity']
```

Unique values for discrete features:

[ ]:
```
for feat in disc_feat:
    list = df1[feat].unique()
    print(f"{feat}: There are {len(list)} unique items in this list. \n {list}")
```

```
Number_of_vehicles_involved: There are 6 unique items in this list.
 [2 1 3 6 4 7]
Number_of_casualties: There are 8 unique items in this list.
 [2 1 3 4 6 5 8 7]
```

Unique values for categorical features:

[ ]:
```
for feat in cat_feat:
    list = df1[feat].unique()
    print(f"{feat}: There are {len(list)} unique items in this list. \n {list}")
```

```
Vehicle_movement: There are 14 unique items in this list.
 ['Going straight' 'U-Turn' 'Moving Backward' 'Turnover' 'Waiting to go'
 'Getting off' 'Reversing' 'Unknown' 'Parked' 'Stopping' 'Overtaking'
 'Other' 'Entering a junction' nan]
Casualty_class: There are 4 unique items in this list.
 ['na' 'Driver or rider' 'Pedestrian' 'Passenger']
Sex_of_casualty: There are 3 unique items in this list.
 ['na' 'Male' 'Female']
Age_band_of_casualty: There are 6 unique items in this list.
 ['na' '31-50' '18-30' 'Under 18' 'Over 51' '5']
Casualty_severity: There are 4 unique items in this list.
 ['na' '3' '2' '1']
Work_of_casuality: There are 8 unique items in this list.
 [nan 'Driver' 'Other' 'Unemployed' 'Employee' 'Self-employed' 'Student'
 'Unknown']
Fitness_of_casuality: There are 6 unique items in this list.
 [nan 'Normal' 'Deaf' 'Other' 'Blind' 'NormalNormal']
Pedestrian_movement: There are 9 unique items in this list.
 ['Not a Pedestrian' "Crossing from driver's nearside"
 'Crossing from nearside - masked by parked or statioNot a Pedestrianry vehicle'
 'Unknown or other'
 'Crossing from offside - masked by  parked or statioNot a Pedestrianry vehicle'
 'In carriageway, statioNot a Pedestrianry - not crossing  (standing or
playing)'
 'Walking along in carriageway, back to traffic'
 'Walking along in carriageway, facing traffic'
 'In carriageway, statioNot a Pedestrianry - not crossing  (standing or playing)
- masked by parked or statioNot a Pedestrianry vehicle']
Cause_of_accident: There are 20 unique items in this list.
 ['Moving Backward' 'Overtaking' 'Changing lane to the left'
 'Changing lane to the right' 'Overloading' 'Other'
 'No priority to vehicle' 'No priority to pedestrian' 'No distancing'
 'Getting off the vehicle improperly' 'Improper parking' 'Overspeed'
 'Driving carelessly' 'Driving at high speed' 'Driving to the left'
 'Unknown' 'Overturning' 'Turnover' 'Driving under the influence of drugs'
 'Drunk driving']
Accident_severity: There are 3 unique items in this list.
 ['Slight Injury' 'Serious Injury' 'Fatal injury']
```

1.3 Data Quality Report

1.3.1 Discrete Features

[ ]:
```
data_quality_report_cont(df1, disc_feat)
```

```
Data Quality for Continous Features
Total Features: 2
                      Feature  Count  Missing  % missing  Cardinality
0  Number_of_vehicles_involved   8210        0        0.0            6
1         Number_of_casualties   8210        0        0.0            8


Descriptive Stats
                             count  mean   std  min  25%  50%  75%  max
Number_of_vehicles_involved  8210.0  2.01  0.64  1.0  2.0  2.0  2.0  7.0
Number_of_casualties         8210.0  1.51  0.97  1.0  1.0  1.0  2.0  8.0
```

There are no missing values in the discrete variables. Therefore, no imputation or data cleaning required.

1.3.2 Categorical Features

[ ]:
```
data_quality_report_cat(df1, cat_feat)
```

```
Data Quality Report for Categorical Features
============================================
Stats
-----
                Feature  Count  Missing  % Missing  Cardinality
0      Vehicle_movement   8026      184       2.29           14
1        Casualty_class   8210        0       0.00            4
2       Sex_of_casualty   8210        0       0.00            3
3  Age_band_of_casualty   8210        0       0.00            6
4     Casualty_severity   8210        0       0.00            4
5     Work_of_casuality   6062     2148      35.43            8
6  Fitness_of_casuality   6440     1770      27.48            6
7   Pedestrian_movement   8210        0       0.00            9
8     Cause_of_accident   8210        0       0.00           20
9     Accident_severity   8210        0       0.00            3


Mode 1
------
                Feature            Mode 1  Mode 1 Freq.  Mode 1 %
0      Vehicle_movement    Going straight          5481     68.29
1        Casualty_class   Driver or rider          3201     38.99
2       Sex_of_casualty              Male          3491     42.52
3  Age_band_of_casualty                na          2907     35.41
4     Casualty_severity                 3          4715     57.43
```

```
5     Work_of_casuality            Driver          3923     64.71
6  Fitness_of_casuality            Normal          6391     99.24
7   Pedestrian_movement  Not a Pedestrian          7571     92.22
8     Cause_of_accident     No distancing          1520     18.51
9     Accident_severity     Slight Injury          7082     86.26
```

```
Mode 2
------
                Feature                      Mode 2  Mode 2 Freq.  Mode 2 %
0      Vehicle_movement             Moving Backward           642      8.00
1        Casualty_class                          na          2907     35.41
2       Sex_of_casualty                          na          2907     35.41
3  Age_band_of_casualty                       18-30          2013     24.52
4     Casualty_severity                          na          2907     35.41
5     Work_of_casuality                Self-employed          1343     22.15
6  Fitness_of_casuality                NormalNormal            13      0.20
7   Pedestrian_movement             Unknown or other           231      2.81
8     Cause_of_accident  Changing lane to the right          1233     15.02
9     Accident_severity               Serious Injury          1046     12.74
```

```
Descriptive Stats
-----------------
                     count unique                top  freq
Vehicle_movement      8026     13    Going straight  5481
Casualty_class        8210      4   Driver or rider  3201
Sex_of_casualty       8210      3              Male  3491
Age_band_of_casualty  8210      6                na  2907
Casualty_severity     8210      4                 3  4715
Work_of_casuality     6062      7            Driver  3923
Fitness_of_casuality  6440      5            Normal  6391
Pedestrian_movement   8210      9  Not a Pedestrian  7571
Cause_of_accident     8210     20     No distancing  1520
Accident_severity     8210      3     Slight Injury  7082
```

Three features have missing values. Since less than 60% of the values are missing, missing values will be imputed with the mode.

[ ]:
```
df1.info(memory_usage='deep')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8210 entries, 0 to 8209
Data columns (total 12 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   Number_of_vehicles_involved  8210 non-null   int64
 1   Number_of_casualties         8210 non-null   int64
 2   Vehicle_movement             8026 non-null   object
```

```
 3   Casualty_class               8210 non-null   object
 4   Sex_of_casualty              8210 non-null   object
 5   Age_band_of_casualty         8210 non-null   object
 6   Casualty_severity            8210 non-null   object
 7   Work_of_casuality            6062 non-null   object
 8   Fitness_of_casuality         6440 non-null   object
 9   Pedestrian_movement          8210 non-null   object
 10  Cause_of_accident            8210 non-null   object
 11  Accident_severity            8210 non-null   object
dtypes: int64(2), object(10)
memory usage: 5.2 MB
```

[ ]:
```
# Impute mode into missing values
col_impute = ['Vehicle_movement', 'Work_of_casuality', 'Fitness_of_casuality']

for col in col_impute:
    most_frequent = df1[col].mode()[0]
    df1[col] = df1[col].replace(np.nan, most_frequent)
```

```
C:\Users\xxkjx\AppData\Local\Temp\ipykernel_46252\408343183.py:6:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1[col] = df1[col].replace(np.nan, most_frequent)
```

[ ]:
```
# Check imputation
data_quality_report_cat(df1, col_impute)
```

```
Data Quality Report for Categorical Features
============================================
Stats
-----
                Feature  Count  Missing  % Missing  Cardinality
0      Vehicle_movement   8210        0        0.0           13
1     Work_of_casuality   8210        0        0.0            7
2  Fitness_of_casuality   8210        0        0.0            5


Mode 1
------
                Feature          Mode 1  Mode 1 Freq.  Mode 1 %
0      Vehicle_movement  Going straight          5665     69.00
1     Work_of_casuality          Driver          6071     73.95
2  Fitness_of_casuality          Normal          8161     99.40
```

```
Mode 2
------
                Feature           Mode 2  Mode 2 Freq.  Mode 2 %
0      Vehicle_movement  Moving Backward           642      7.82
1     Work_of_casuality    Self-employed          1343     16.36
2  Fitness_of_casuality     NormalNormal            13      0.16


Descriptive Stats
-----------------
                     count unique             top  freq
Vehicle_movement      8210     13  Going straight  5665
Work_of_casuality     8210      7          Driver  6071
Fitness_of_casuality  8210      5          Normal  8161
```

[ ]:
```
dfj = df1.copy()
dfj.head()
```

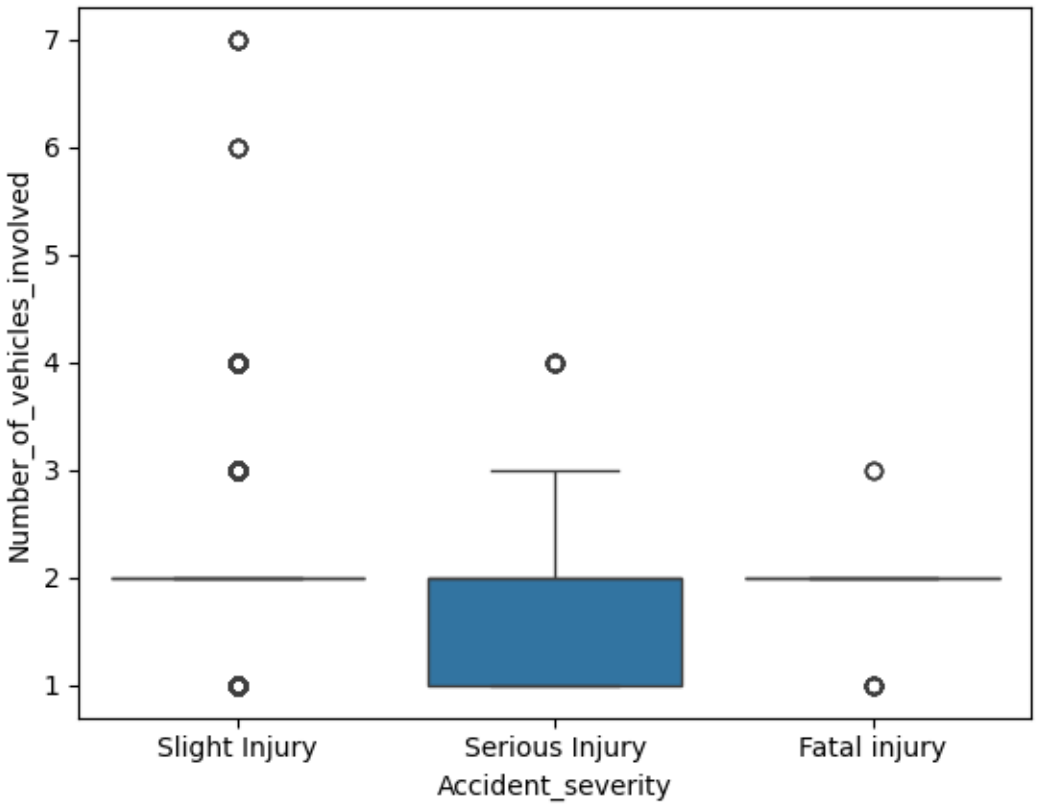
[ ]:
```
   Number_of_vehicles_involved  Number_of_casualties Vehicle_movement  \
0                            2                     2   Going straight
1                            2                     2   Going straight
2                            2                     2   Going straight
3                            2                     2   Going straight
4                            2                     2   Going straight

    Casualty_class Sex_of_casualty Age_band_of_casualty Casualty_severity  \
0               na              na                   na               na
1               na              na                   na               na
2  Driver or rider            Male                31-50                3
3       Pedestrian          Female                18-30                3
4               na              na                   na               na

  Work_of_casuality Fitness_of_casuality Pedestrian_movement  \
0            Driver               Normal    Not a Pedestrian
1            Driver               Normal    Not a Pedestrian
2            Driver               Normal    Not a Pedestrian
3            Driver               Normal    Not a Pedestrian
4            Driver               Normal    Not a Pedestrian

           Cause_of_accident Accident_severity
0             Moving Backward     Slight Injury
1                  Overtaking     Slight Injury
2   Changing lane to the left    Serious Injury
3  Changing lane to the right     Slight Injury
4                  Overtaking     Slight Injury
```

1.4 Data Analysis of Discrete Features

1.4.1 Descriptive Statistics and Box Plots

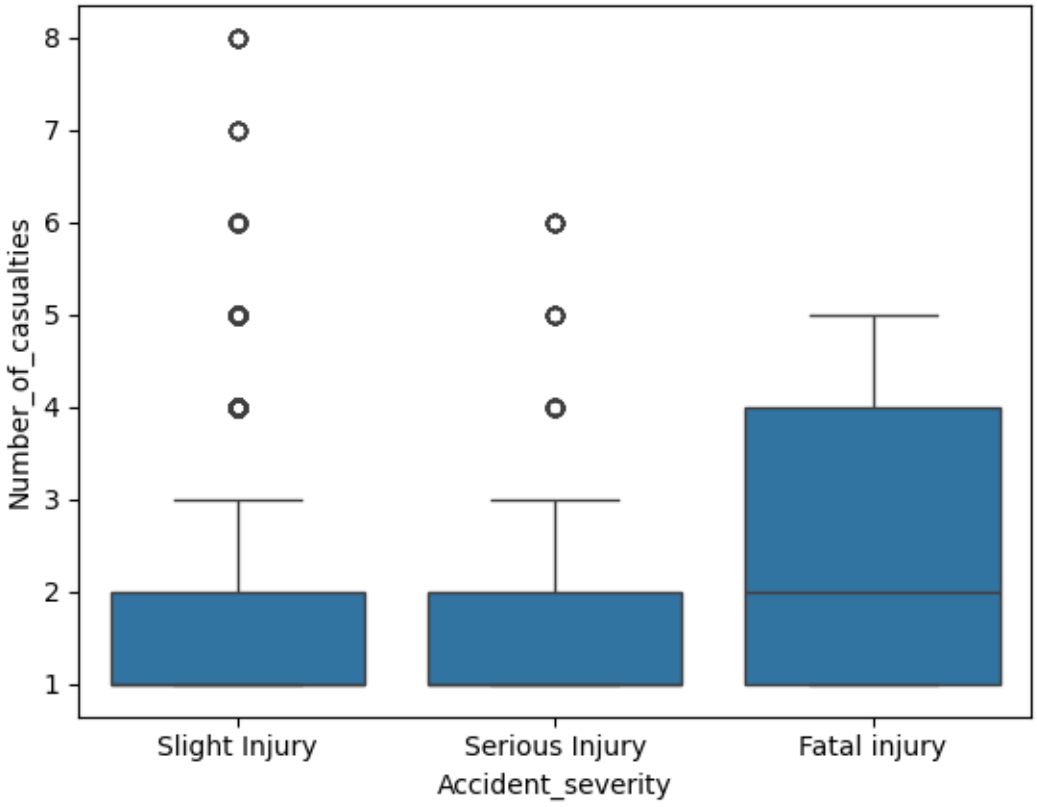
[ ]:
```
# Descriptive stats and box plots
for col in disc_feat:
    descriptive_stats = df1.groupby(target)[col].describe().round()
    print(f"Descriptive Statistics for {col}:\n {descriptive_stats}")
    sns.boxplot(x=target, y=col, data = df1)
    plt.show()
```

```
Descriptive Statistics for Number_of_vehicles_involved:
                     count  mean  std  min  25%  50%  75%  max
Accident_severity
Fatal injury          82.0   2.0  0.0  1.0  2.0  2.0  2.0  3.0
Serious Injury      1046.0   2.0  1.0  1.0  1.0  2.0  2.0  4.0
Slight Injury       7082.0   2.0  1.0  1.0  2.0  2.0  2.0  7.0
```




```
Descriptive Statistics for Number_of_casualties:
                     count  mean  std  min  25%  50%  75%  max
Accident_severity
Fatal injury          82.0   2.0  1.0  1.0  1.0  2.0  4.0  5.0
Serious Injury      1046.0   2.0  1.0  1.0  1.0  1.0  2.0  6.0
```

```
Slight Injury      7082.0   1.0  1.0  1.0  1.0  1.0  2.0  8.0
```



1.4.2 Chi-Squre Test of Independence

[ ]:
```python
# Create function for cramers_v
def cramers_v(chi2, n, min_dim):
    return np.sqrt(chi2 / (n * (min_dim - 1)))
```

[ ]:
```python
# Contingency tables
sig_disc = []
for col in disc_feat:
    ct = pd.crosstab(df1[col], df1[target])
    chi2, p, dof, expected = chi2_contingency(ct)
    # Calculate Cramer's V
    n = ct.sum().sum()
    min_dim = min(ct.shape) - 1
    cramer_v = cramers_v(chi2, n, min_dim)
    # Print if p-value is less than .5
    if p <= 0.5:
        sig_disc.append(col)
        print(f"Chi-Square Test for '{col}':")
```

```
        print(f"Chi2: {chi2:.02f}")
        print(f"P-values: {p:.02f}")
        print(f"Degrees of Freedom: {dof:.02f}")
        print(f"Cramér's V: {cramer_v}\n")

print(sig_disc)
```

```
Chi-Square Test for 'Number_of_vehicles_involved':
Chi2: 142.81
P-values: 0.00
Degrees of Freedom: 10.00
Cramér's V: 0.13189001744457038

Chi-Square Test for 'Number_of_casualties':
Chi2: 116.66
P-values: 0.00
Degrees of Freedom: 14.00
Cramér's V: 0.1192037549718662

['Number_of_vehicles_involved', 'Number_of_casualties']
```

1.4.3 Conclusion of Analysis of Discrete Features

Based on the p-values, these features are statistically significant to the target variable: - Number_of_vehicles_involved - Number_of_casualties

1.4.4 Data Analysis of Categorical Features

[ ]:
```
df_cat = df1[cat_feat]
df_cat.head()
```

[ ]:
```
  Vehicle_movement   Casualty_class Sex_of_casualty Age_band_of_casualty  \
0   Going straight              na              na                   na
1   Going straight              na              na                   na
2   Going straight  Driver or rider            Male                31-50
3   Going straight       Pedestrian          Female                18-30
4   Going straight              na              na                   na

  Casualty_severity Work_of_casuality Fitness_of_casuality  \
0                na           Driver               Normal
1                na           Driver               Normal
2                 3           Driver               Normal
3                 3           Driver               Normal
4                na           Driver               Normal

  Pedestrian_movement          Cause_of_accident Accident_severity
0    Not a Pedestrian            Moving Backward     Slight Injury
1    Not a Pedestrian                 Overtaking     Slight Injury
```

```
2    Not a Pedestrian   Changing lane to the left    Serious Injury
3    Not a Pedestrian  Changing lane to the right     Slight Injury
4    Not a Pedestrian                  Overtaking     Slight Injury
```

[ ]:
```
cat_feat
```

[ ]:
```
['Vehicle_movement',
 'Casualty_class',
 'Sex_of_casualty',
 'Age_band_of_casualty',
 'Casualty_severity',
 'Work_of_casuality',
 'Fitness_of_casuality',
 'Pedestrian_movement',
 'Cause_of_accident',
 'Accident_severity']
```

1.4.5 Chi-Square Test of Independence

Contingency Tables

[ ]:
```
# Contingency tables
sig_cat = []
for col in cat_feat[:-1]:
    ct = pd.crosstab(df_cat[col], df_cat[target])
    chi2, p, dof, expected = chi2_contingency(ct)
    # Calculate Cramer's V
    n = ct.sum().sum()
    min_dim = min(ct.shape) - 1
    cramer_v = cramers_v(chi2, n, min_dim)
    # Print if p-value is less than .5
    if p <= 0.5:
        sig_cat.append(col)
        print(f"Chi-Square Test for '{col}':")
        print(f"Chi2: {chi2:.02f}")
        print(f"P-values: {p:.02f}")
        print(f"Degrees of Freedom: {dof:.02f}")
        print(f"Cramér's V: {cramer_v}\n")

print(sig_cat)
```

```
Chi-Square Test for 'Vehicle_movement':
Chi2: 28.24
P-values: 0.25
Degrees of Freedom: 24.00
Cramér's V: 0.05864406781565965

Chi-Square Test for 'Casualty_class':
Chi2: 7.62
```

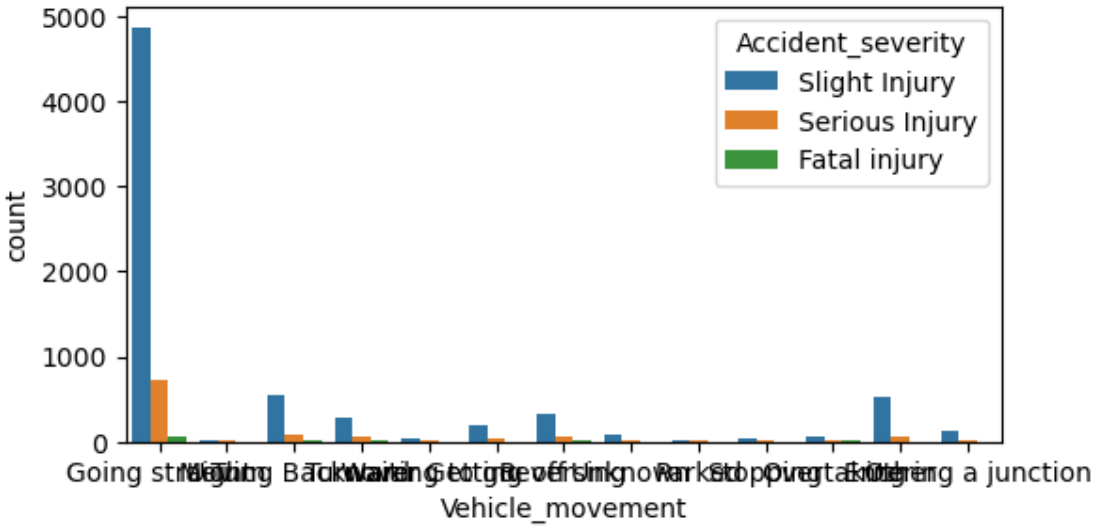
```
P-values: 0.27
Degrees of Freedom: 6.00
Cramér's V: 0.030467697968743344

Chi-Square Test for 'Age_band_of_casualty':
Chi2: 13.97
P-values: 0.17
Degrees of Freedom: 10.00
Cramér's V: 0.04124580260715501

Chi-Square Test for 'Pedestrian_movement':
Chi2: 18.03
P-values: 0.32
Degrees of Freedom: 16.00
Cramér's V: 0.046861619874994875

Chi-Square Test for 'Cause_of_accident':
Chi2: 57.20
P-values: 0.02
Degrees of Freedom: 38.00
Cramér's V: 0.08346792519076886

['Vehicle_movement', 'Casualty_class', 'Age_band_of_casualty',
'Pedestrian_movement', 'Cause_of_accident']
```

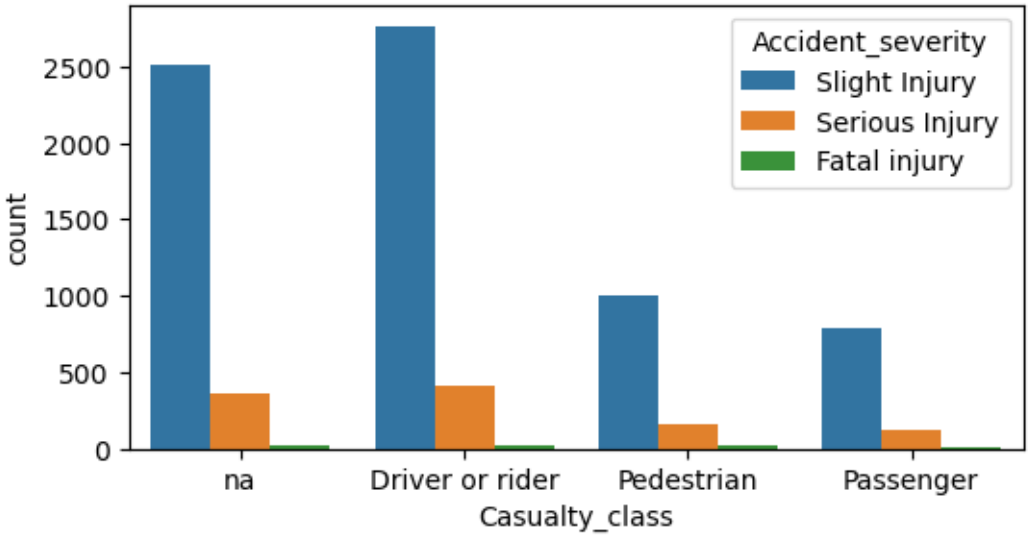
1.4.6 Analysis of Contingency Table and Chi-Square Test of Independence

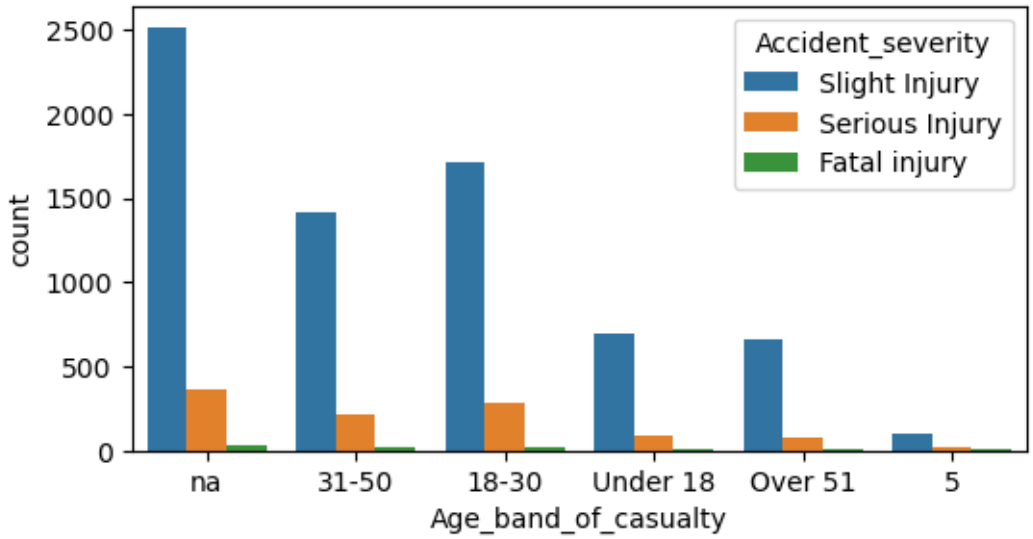
All categorical features were analyzed to determine if there was a significant association between each feature and the target variable. Features that had p-value ≤ 0.5, are shown in the output.

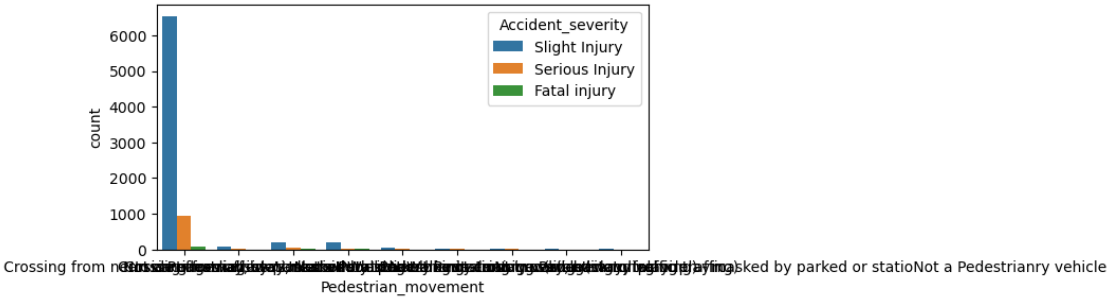
Considering a confidence level of 95%, only one feature met this requirement: * Cause_of_accident

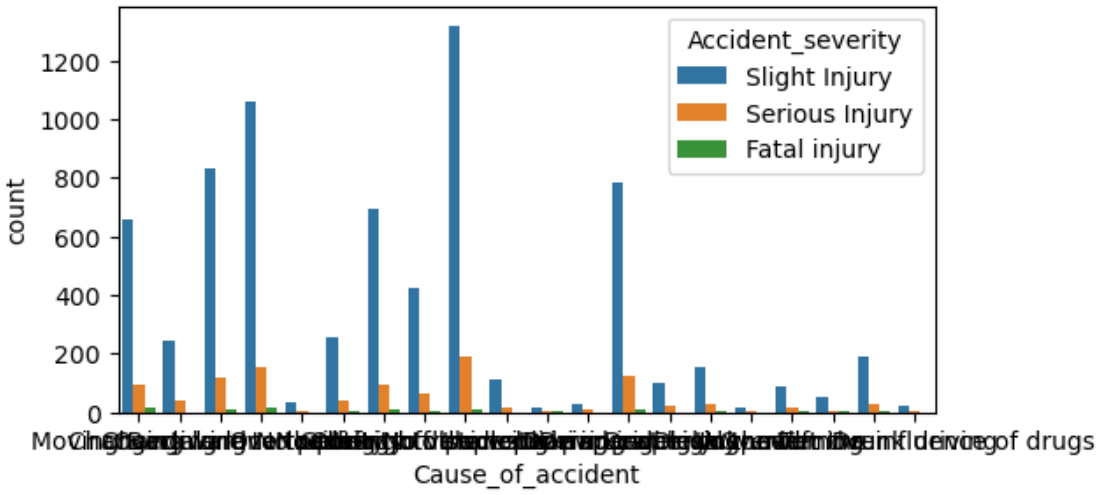
[ ]:
```
for col in sig_cat:
    plt.figure(figsize=(6, 3))
    sns.countplot(x=col, hue=target, data=df_cat)
    plt.show()
```


Accident_severity
Slight Injury
Serious Injury
Fatal injury
count
5000
4000
3000
2000
1000
0
Going straight
Vehicle_movement


Accident_severity
Slight Injury
Serious Injury
Fatal injury
count
2500
2000
1500
1000
500
0
na
Driver or rider
Pedestrian
Passenger
Casualty_class


Accident_severity
Slight Injury
Serious Injury
Fatal injury
count
2500
2000
1500
1000
500
0
na
31-50
18-30
Under 18
Over 51
5
Age_band_of_casualty


Accident_severity
Slight Injury
Serious Injury
Fatal injury
count
6000
5000
4000
3000
2000
1000
0
Pedestrian_movement


Accident_severity
Slight Injury
Serious Injury
Fatal injury
count
1200
1000
800
600
400
200
0
Cause_of_accident

1.5 EDA Results

Based on EDA by the team, nine features of the original dataset were determined to have a correlation to the target variable: - Area_accident_occured - Types_of_Junction - Light_conditions - Number_of_vehicles_involved - Number_of_casualties - Cause_of_accident, - Day_of_week - Sex_of_driver - Age_band_of_driver

Target variable: - Accident_severity

2 Preparation of Dataset / Feature Selection

[ ]:
```
df = pd.read_csv(dataset/"accidents_train.csv")
df.head()
```

[ ]:

	Num	Time	Day_of_week	Age_band_of_driver	Sex_of_driver	\
0	1	17:02:00	Monday	18-30	Male	
1	2	17:02:00	Monday	31-50	Male	
2	3	17:02:00	Monday	18-30	Male	
3	4	1:06:00	Sunday	18-30	Male	
4	5	1:06:00	Sunday	18-30	Male	

	Educational_level	Vehicle_driver_relation	Driving_experience	\
0	Above high school	Employee	1-2yr	
1	Junior high school	Employee	Above 10yr	
2	Junior high school	Employee	1-2yr	
3	Junior high school	Employee	5-10yr	
4	Junior high school	Employee	2-5yr	

	Type_of_vehicle	Owner_of_vehicle	…	Vehicle_movement	\
0	Automobile	Owner	…	Going straight	
1	Public (> 45 seats)	Owner	…	Going straight	
2	Lorry (41?100Q)	Owner	…	Going straight	
3	Public (> 45 seats)	Governmental	…	Going straight	
4	NaN	Owner	…	Going straight	

	Casualty_class	Sex_of_casualty	Age_band_of_casualty	Casualty_severity	\
0	na	na	na	na	
1	na	na	na	na	
2	Driver or rider	Male	31-50	3	
3	Pedestrian	Female	18-30	3	
4	na	na	na	na	

	Work_of_casuality	Fitness_of_casuality	Pedestrian_movement	\
0	NaN	NaN	Not a Pedestrian	
1	NaN	NaN	Not a Pedestrian	
2	Driver	NaN	Not a Pedestrian	
3	Driver	Normal	Not a Pedestrian	
4	NaN	NaN	Not a Pedestrian	

```
             Cause_of_accident Accident_severity
0              Moving Backward     Slight Injury
1                   Overtaking     Slight Injury
2   Changing lane to the left    Serious Injury
3  Changing lane to the right     Slight Injury
4                   Overtaking     Slight Injury

[5 rows x 33 columns]
```

2.1 Features Selected per EDA

[ ]:
```
columns = ["Area_accident_occured", "Types_of_Junction", "Light_conditions",
           "Number_of_vehicles_involved", "Number_of_casualties",␣
 ↪"Cause_of_accident",
           "Day_of_week", "Sex_of_driver", "Age_band_of_driver",
           "Accident_severity"]
df1 = df[columns]
```

[ ]:
```
df1.shape
```

[ ]:
```
(8210, 10)
```

[ ]:
```
df1.info(memory_usage='deep')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8210 entries, 0 to 8209
Data columns (total 10 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   Area_accident_occured        8050 non-null   object
 1   Types_of_Junction            8210 non-null   object
 2   Light_conditions             8210 non-null   object
 3   Number_of_vehicles_involved  8210 non-null   int64
 4   Number_of_casualties         8210 non-null   int64
 5   Cause_of_accident            8210 non-null   object
 6   Day_of_week                  8210 non-null   object
 7   Sex_of_driver                8210 non-null   object
 8   Age_band_of_driver           8210 non-null   object
 9   Accident_severity            8210 non-null   object
dtypes: int64(2), object(8)
memory usage: 4.3 MB
```

[ ]:
```
# Discrete features
disc_feat = ['Number_of_vehicles_involved', 'Number_of_casualties']
# Categorical features
cat_feat = ["Area_accident_occured", "Types_of_Junction", "Light_conditions",
```

```
            "Number_of_vehicles_involved", "Number_of_casualties",␣
 ↪"Cause_of_accident",
            "Day_of_week", "Sex_of_driver", "Age_band_of_driver"]
# Target variable
target = ["Accident_severity"]
```

2.2 Data Quality Reports

[ ]:
```
data_quality_report_cont(df1, disc_feat)
```

```
Data Quality for Continous Features
Total Features: 2
                       Feature  Count  Missing  % missing  Cardinality
0  Number_of_vehicles_involved   8210        0        0.0            6
1         Number_of_casualties   8210        0        0.0            8


Descriptive Stats
                              count  mean   std  min  25%  50%  75%  max
Number_of_vehicles_involved  8210.0  2.01  0.64  1.0  2.0  2.0  2.0  7.0
Number_of_casualties         8210.0  1.51  0.97  1.0  1.0  1.0  2.0  8.0
```

[ ]:
```
data_quality_report_cat(df1, cat_feat)
```

```
Data Quality Report for Categorical Features
============================================
Stats
-----
                       Feature  Count  Missing  % Missing  Cardinality
0        Area_accident_occured   8050      160       1.99           15
1            Types_of_Junction   8210        0       0.00            8
2             Light_conditions   8210        0       0.00            4
3  Number_of_vehicles_involved   8210        0       0.00            6
4         Number_of_casualties   8210        0       0.00            8
5            Cause_of_accident   8210        0       0.00           20
6                  Day_of_week   8210        0       0.00            7
7                Sex_of_driver   8210        0       0.00            3
8           Age_band_of_driver   8210        0       0.00            5


Mode 1
------
                       Feature         Mode 1  Mode 1 Freq.  Mode 1 %
0        Area_accident_occured          Other          2511     31.19
1            Types_of_Junction        Y Shape          3118     37.98
2             Light_conditions       Daylight          5924     72.16
3  Number_of_vehicles_involved              2          5740     69.91
4         Number_of_casualties              1          5786     70.48
```

```
5            Cause_of_accident  No distancing          1520     18.51
6                  Day_of_week         Friday          1326     16.15
7               Sex_of_driver           Male          7582     92.35
8           Age_band_of_driver          18-30          2728     33.23


Mode 2
------
                      Feature                     Mode 2  Mode 2 Freq.  \
0        Area_accident_occured                Office areas          2323
1            Types_of_Junction                 No junction          2924
2             Light_conditions       Darkness - lights lit          2171
3  Number_of_vehicles_involved                           1          1323
4         Number_of_casualties                           2          1440
5            Cause_of_accident  Changing lane to the right          1233
6                  Day_of_week                    Thursday          1288
7               Sex_of_driver                      Female           462
8           Age_band_of_driver                       31-50          2688

   Mode 2 %
0     28.86
1     35.62
2     26.44
3     16.11
4     17.54
5     15.02
6     15.69
7      5.63
8     32.74


Descriptive Stats
-----------------
                      count unique            top  freq
Area_accident_occured  8050     14          Other  2511
Types_of_Junction      8210      8        Y Shape  3118
Light_conditions       8210      4       Daylight  5924
Cause_of_accident      8210     20  No distancing  1520
Day_of_week            8210      7         Friday  1326
Sex_of_driver          8210      3           Male  7582
Age_band_of_driver     8210      5          18-30  2728
```

2.2.1 Data clean up (Missing records)

[ ]:
```
df1['Area_accident_occured'] = df1['Area_accident_occured'].
↪fillna(df1['Area_accident_occured'].mode()[0])
```

```
C:\Users\xxkjx\AppData\Local\Temp\ipykernel_46252\2051851754.py:1:
```

```
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1['Area_accident_occured'] =
df1['Area_accident_occured'].fillna(df1['Area_accident_occured'].mode()[0])
```

2.2.2 Verify clean up / imputation

[ ]:
```
data_quality_report_cat(df1, cat_feat)
```

```
Data Quality Report for Categorical Features
============================================
Stats
-----
                      Feature  Count  Missing  % Missing  Cardinality
0        Area_accident_occured   8210        0        0.0           14
1            Types_of_Junction   8210        0        0.0            8
2             Light_conditions   8210        0        0.0            4
3  Number_of_vehicles_involved   8210        0        0.0            6
4         Number_of_casualties   8210        0        0.0            8
5            Cause_of_accident   8210        0        0.0           20
6                  Day_of_week   8210        0        0.0            7
7                Sex_of_driver   8210        0        0.0            3
8           Age_band_of_driver   8210        0        0.0            5


Mode 1
------
                      Feature         Mode 1  Mode 1 Freq.  Mode 1 %
0        Area_accident_occured          Other          2671     32.53
1            Types_of_Junction        Y Shape          3118     37.98
2             Light_conditions      Daylight          5924     72.16
3  Number_of_vehicles_involved              2          5740     69.91
4         Number_of_casualties              1          5786     70.48
5            Cause_of_accident  No distancing          1520     18.51
6                  Day_of_week         Friday          1326     16.15
7                Sex_of_driver           Male          7582     92.35
8           Age_band_of_driver          18-30          2728     33.23


Mode 2
------
                      Feature                    Mode 2  Mode 2 Freq.  \
0        Area_accident_occured             Office areas          2323
1            Types_of_Junction              No junction          2924
```

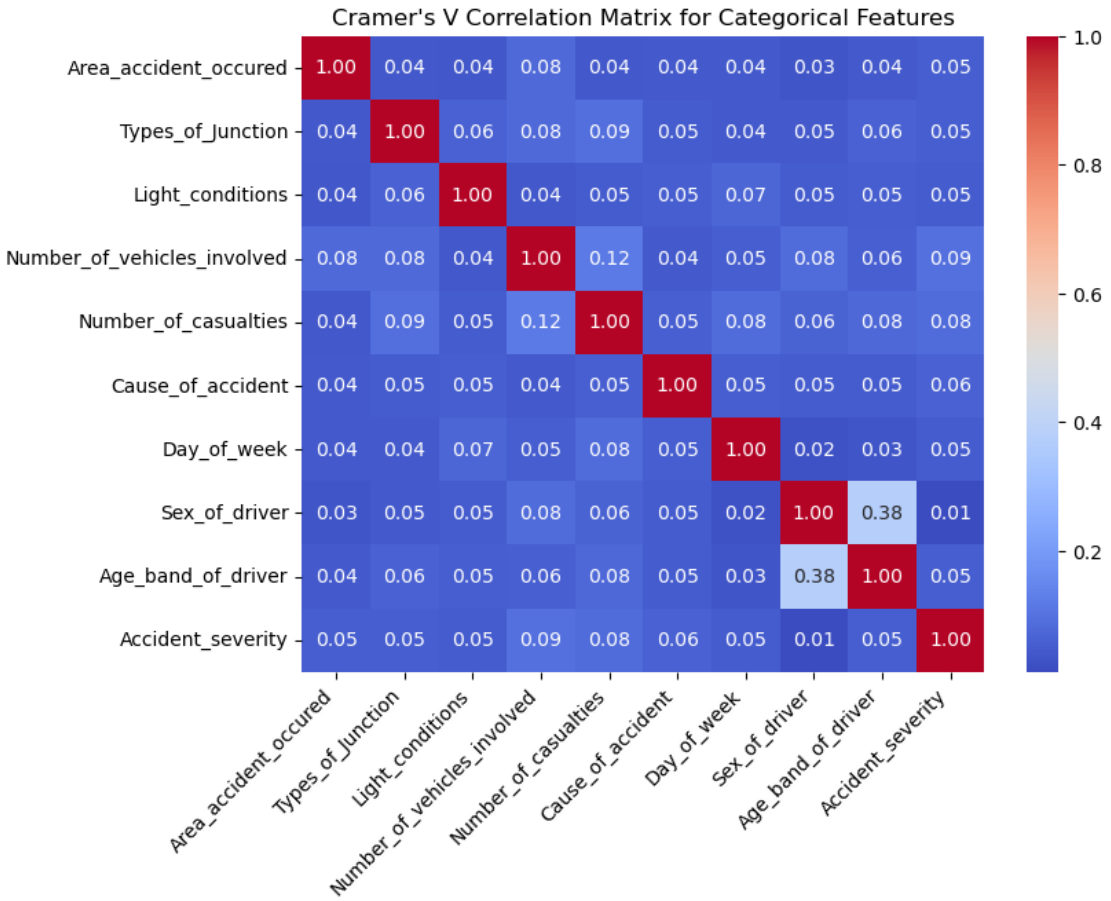
```
2             Light_conditions       Darkness - lights lit          2171
3  Number_of_vehicles_involved                           1          1323
4         Number_of_casualties                           2          1440
5            Cause_of_accident  Changing lane to to the right          1233
6                  Day_of_week                    Thursday          1288
7                Sex_of_driver                      Female           462
8            Age_band_of_driver                       31-50          2688

   Mode 2 %
0     28.29
1     35.62
2     26.44
3     16.11
4     17.54
5     15.02
6     15.69
7      5.63
8     32.74


Descriptive Stats
-----------------
                       count unique            top  freq
Area_accident_occured   8210     14          Other  2671
Types_of_Junction       8210      8        Y Shape  3118
Light_conditions        8210      4       Daylight  5924
Cause_of_accident       8210     20  No distancing  1520
Day_of_week             8210      7         Friday  1326
Sex_of_driver           8210      3           Male  7582
Age_band_of_driver      8210      5          18-30  2728
```

2.2.3 Correlation Matrix

[ ]:
```python
def cramers_v(x, y):
    """Calculate Cramér's V statistic for categorical-categorical association.
    """
    confusion_matrix = pd.crosstab(x, y)
    chi2 = chi2_contingency(confusion_matrix)[0]
    n = confusion_matrix.sum().sum()
    phi2 = chi2 / n
    r, k = confusion_matrix.shape
    return np.sqrt(phi2 / min(k - 1, r - 1))

def cramers_v_matrix(df):
    """Create a correlation matrix for categorical features using Cramér's V."""
    cols = df.columns
    n = len(cols)
```

```
    corr_matrix = pd.DataFrame(np.zeros((n, n)), columns=cols, index=cols)

    for i in range(n):
        for j in range(i, n):
            v = cramers_v(df[cols[i]], df[cols[j]])
            corr_matrix.iat[i, j] = v
            corr_matrix.iat[j, i] = v

    return corr_matrix
```

[ ]:
```
categorical_columns = df1.columns
categorical_df = df[categorical_columns]
cramers_v_corr_matrix = cramers_v_matrix(df1)

plt.figure(figsize=(8, 6))
sns.heatmap(cramers_v_corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.xticks(rotation=45, ha='right')
plt.title('Cramer\'s V Correlation Matrix for Categorical Features')
plt.show()
```



2.3 Export dataset

The dataset is exported with the features and target discovered during the exploratory data analysis.

[ ]:
```
df1.to_csv('../dataset/accidents_clean_train.csv', index=False)
```

[ ]:
```

```