Calanuya road safety analyzes and Report

Malihe Alavi

30 jan 2024 m.alavi.649@gmail.com **+989370336116**

Introduction:

Road safety is a critical issue that affects millions of people worldwide. The World Health Organization (WHO) reports that road traffic accidents are the leading cause of death among young people aged 5-29 years 1. Machine learning (ML) has emerged as a promising technology to address road safety problems. ML algorithms can analyze large amounts of data and identify patterns that can help predict and prevent accidents. However, the use of ML in safety-critical systems poses significant engineering challenges.

Too many studies have worked on road safety problems for dataset of different countries. In most of these researches, authors have tried to train a classification model to predict if an accident results in death or sever injuries or not [7][8][9][10]. These studies have used common classification models like XGboost, logistic regression, KNN, random forest, SVM and neural network for classification task. On the other hand, several studies have worked on road-safety problem and tried to learn the pattern and predict future events by time series models. Most of these studies have used ARIMA model for this type of challenges.[1][2][3]

In this study, first we investigate the general trends and patterns that effect on the accidents, then we develop 3 ML models for predicting the accident based on environmental factors like weather, light, road type, speed limit and … and also based on the time series pattern between days and accidents types. Despite of the previous study that developed a classification model, the target value of Catalunya dataset has different range of integer values which leads us to train regression models with random forest regressor and SVM regressor, also we have developed a time series model for predicting the upcoming year.

In order to train regression models, we had to first preprocess the data because it had missing values that should be replaces with mean or median or should be removed from dataset, also there were too many categorical features which should be encoded to numeric features.

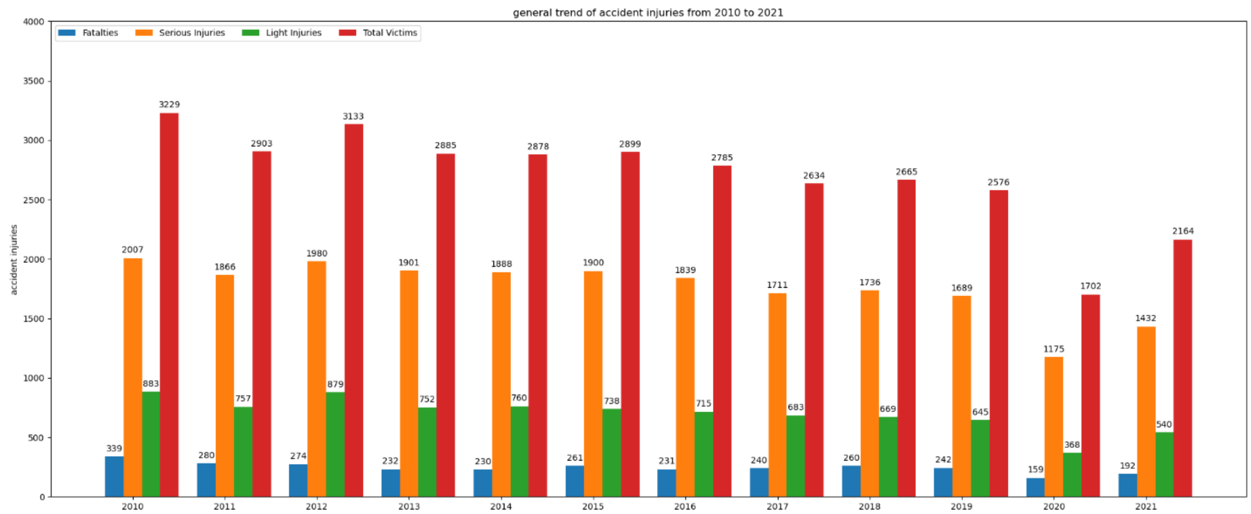
Key finding:

- Random forest regressor outperforms SVR in terms of time complexity, accuracy and simplicity for Calanuya dataset
- The most important factors on accidents were: hour of the day, wind condition, weather condition, light condition and the type of vehicle that was involved in the accident.
- Based on yearly trend, one can concludes that the amounts of total victims and light injuries in accidents has slightly decreased over 10 years
- Most of the accidents have occurred in Barcelona which is about 4 times more than the other province.
- Most of the accidents occurred from 10 a.m. to 8 p.m. and during this period, the evenings about (4p.m. to 8p.m.) are the pic of crashes
- Type of road and speed limit info also have great impact on the accidents
- the accident rates are more in months May, June, July and October respect to serious Injuries.
- Fatalities rates are relatively the same in all months
- light injuries also increases in July.

Data Visualizations and Analyzes:

1. General Trends

What are the overall trends in traffic accidents, fatalities, and serious injuries in Catalonia from 2010-2021?



As can be seen from the chart, the overall total victims have been slightly decreased during 2010 to 2021, the number of total victims in 2010 was 3229 and it has reduced to 2164 in 2021. However, there is no significant trend in fatalities rate but a slight reduction also can be seen in serious injuries and light injuries rate. As they reduced from 2007 and 883 in 2010 to 1432 and 540 in 2021 respectively.

2. Accident Characteristics

What common characteristics (time of day, type of road, etc.) are observed in the most severe accidents?

Number of fatalities, serious Injuries, light Injuries and total victims respect to **Road** (top 20)

Road	Tot. vic.
SE	14351
AP-7	951
N-II	947
N-340	662
A-2	650
C-31	642
N-260	546
CR	530
C-14	527
C-58	444
C-55	440
C-12	429
C-17	366
C-32	304

road	fatalities
SE	814
AP-7	140
N-340	136
N-II	124
A-2	80
C-31	78
C-14	76
CR	74
C-12	62
C-55	58
N-260	57
C-25	51
C-17	49
C-16	39

road	Ser. Inj
SE	10786
N-II	501
AP-7	440
C-31	356
N-340	335
CR	326
N-260	324
A-2	319
C-14	254
C-58	240
C-55	231
C-17	199
C-12	198
C-32	184

Road	Li. Inj
SE	2751
AP-7	371
N-II	322
A-2	251
C-31	208
C-14	197
N-340	191
C-12	169
C-58	165
N-260	165
C-55	151
CR	130
C-17	118
N-240	116

C-16	298
N-240	277
C-25	271
C-35	246
B-10	219
C-13	209

C-58	39
N-240	39
C-32	37
N-420	32
C-63	30
C-15	30

C-16	175
C-35	129
C-25	128
B-10	128
N-240	122
C-13	121

C-25	92
C-35	89
C-16	84
C-32	83
C-65	70
B-10	69

Number of fatalities, serious Injuries, light Injuries and total victims respect to **Day Type**

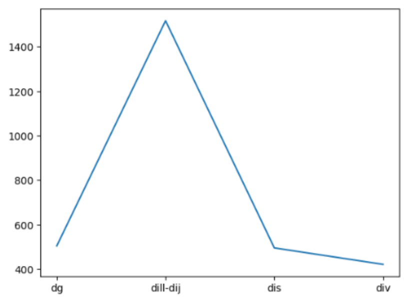
Day Type	Fatality
dg	505
dill-dij	1517
dis	496
div	422

Day Type	Ser. Injuries
Dg	3093
dill-dij	11404
dis	3308
div	3319

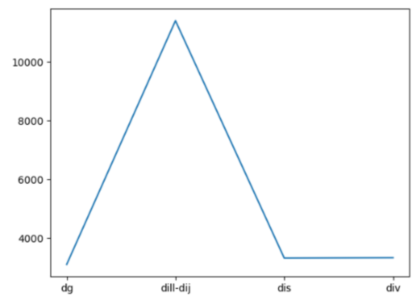
Day Type	Total Vic.
Dg	5183
dill-dij	16953
dis	5383
div	4934

Day Type	Lig. Injuries
dg	1585
dill-dij	4032
dis	1579
div	1193

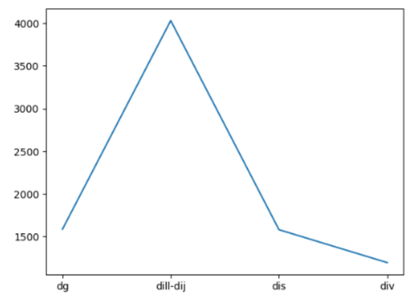
fatalities



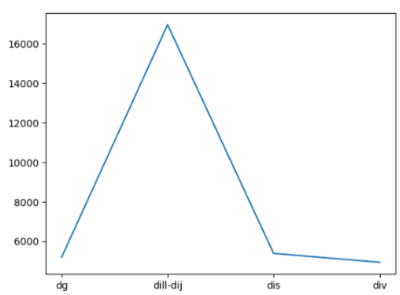
serious injuries



Light injuries



total victims



Number of fatalities, serious Injuries, light Injuries and total victims respect to **Time of Day Grouping**

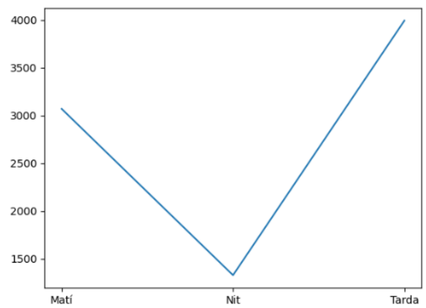
TOD Grouping	Lig. Inj.
Matí	3069
Nit	1327
Tarda	3993

TOD Grouping	Tot. vic.
Matí	12758
Nit	4387
Tarda	15308

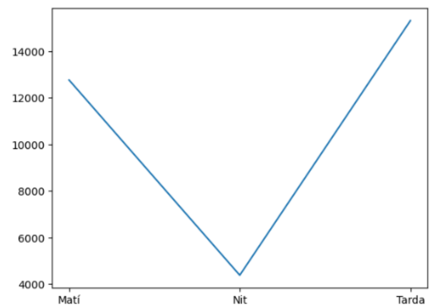
TOD Grouping	fatality
Matí	1097
Nit	558
Tarda	1285

TOD Grouping	Ser. Inj.
Matí	8592
Nit	2502
Tarda	10030

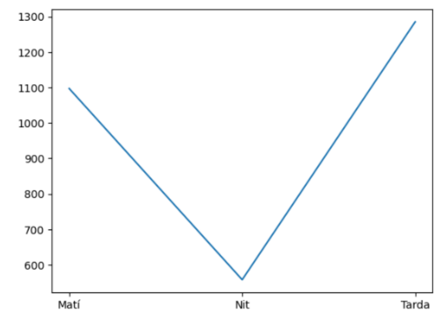
Light injuries



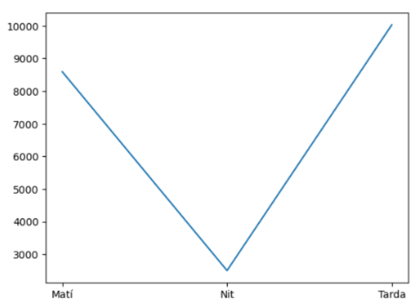
total victims



fatalities



Serious Injuries



3. Geographical Insights

Which municipalities or counties in Catalonia have the highest incidence of traffic accidents? How does this correlate with population density or road network characteristics?

Number of fatalities, serious Injuries, light Injuries and total victims respect to **Municipality Name** (top 20)

Municipality Name	**Tot. vic.**
BARCELONA	4029
TERRASSA	746
LLEIDA	732
SABADELL	708
BADALONA	544
TARRAGONA	527
HOSPITALET DE LLOBREGAT	504
GIRONA	444
REUS	395
MATARO	390
MANRESA	372
SANT BOI DE LLOBREGAT	318
VILANOVA I LA GELTRU	293
VENDRELL, EL	292
SANT CUGAT DEL VALLES	263
VIC	243
CERDANYOLA DEL VALLES	240
GRANOLLERS	233
LLORET DE MAR	233
PRAT DE LLOBREGAT, E	212

Municipality Name	fatality
BARCELONA	302
LLEIDA	52
BADALONA	46
TARRAGONA	44
SABADELL	30
TERRASSA	29
REUS	28
GIRONA	28
HOSPITALET DE LLOBREGAT	26
MANRESA	26
VENDRELL, EL	25
ROCA DEL VALLES, LA	25
AMPOSTA	24
VILANOVA I LA GELTRU	21
MATARO	21
SANT CUGAT DEL VALLES	20
MONT-ROIG DEL CAMP	19
GRANOLLERS	19
VILAFRANCA DEL PENEDES	18
BALAGUER	17

Municipality Name	Ser. inj.
BARCELONA	2656
TERRASSA	574
SABADELL	530
LLEIDA	507
HOSPITALET DE LLOBREGAT	384
TARRAGONA	378
BADALONA	360
GIRONA	320
MATARO	303
REUS	293
MANRESA	264
SANT BOI DE LLOBREGAT	243
VILANOVA I LA GELTRU	213
SANT CUGAT DEL VALLES	187
VIC	187
LLORET DE MAR	184
CERDANYOLA DEL VALLES	179
GRANOLLERS	176
VENDRELL, EL	172
CORNELLA DE LLOBREGAT	162

Municipality Name	Li. Inj.
BARCELONA	1701
LLEIDA	173
SABADELL	148
TERRASSA	143
BADALONA	138
TARRAGONA	105
GIRONA	96
VENDRELL, EL	95
HOSPITALET DE LLOBREGAT	94
MANRESA	82
REUS	74
MATARO	66
SANT QUIRZE DEL VALLES	63
ROCA DEL VALLES, LA	62
AMPOSTA	62
SANT BOI DE LLOBREGAT	59
VILANOVA I LA GELTRU	59
SANT CUGAT DEL VALLES	56
CERDANYOLA DEL VALLES	51
PRAT DE LLOBREGAT, E	50

Number of fatalities, serious Injuries, light Injuries and total victims respect to **<u>County Name</u>** (top 20)

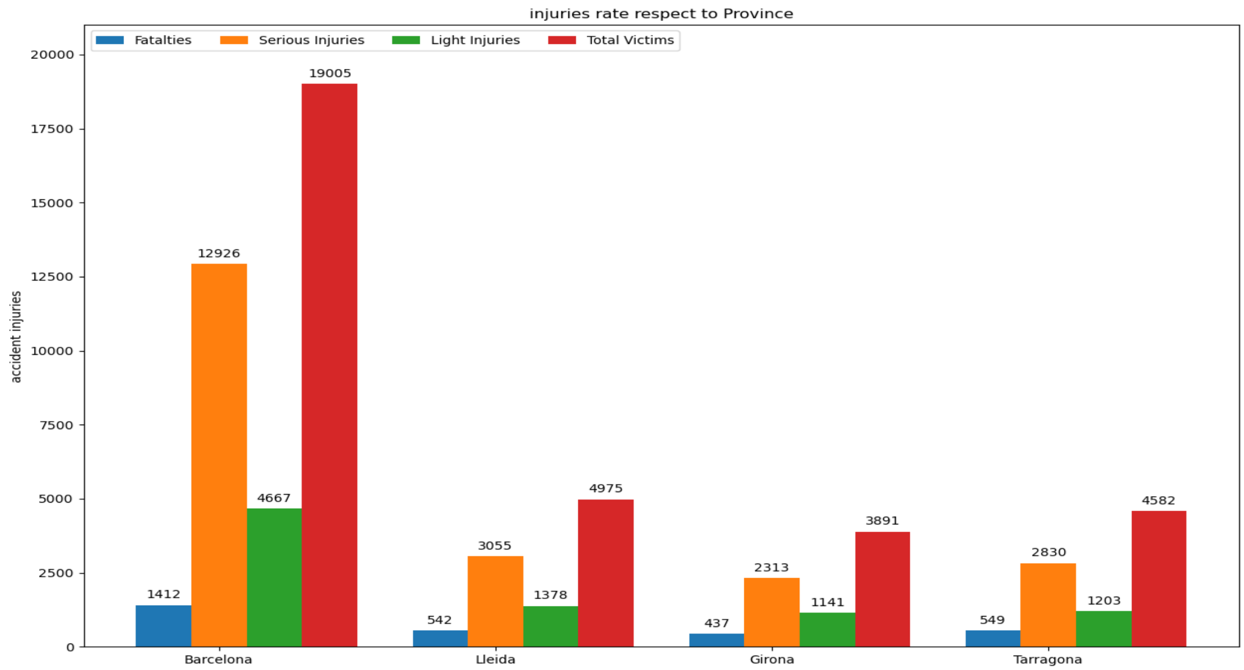
County Name	Total vic.
Barcelones	5371
Valles Occidental	3279
Baix Llobregat	2439
Valles Oriental	1799
Maresme	1395
Segria	1300
Selva	1214
Alt Emporda	1208
Bages	1159
Girones	1052

County Name	Ser. Inj.
Barcelones	3626
Valles Occidental	2335
Baix Llobregat	1737
Valles Oriental	1155
Maresme	1007
Segria	814
Bages	745
Selva	737
Alt Emporda	724
Tarragones	693

County Name	Light Inj.
Barcelones	1354
Valles Occidental	757
Baix Llobregat	545
Valles Oriental	488
Segria	363
Alt Emporda	347
Selva	344
Girones	306
Bages	303
Maresme	296

County Name	fatalities
Barcelones	391
Valles Occidental	187
Baix Llobregat	157
Valles Oriental	156
Alt Emporda	137
Selva	133
Segria	123
Bages	111
Girones	110
Baix Ebre	109

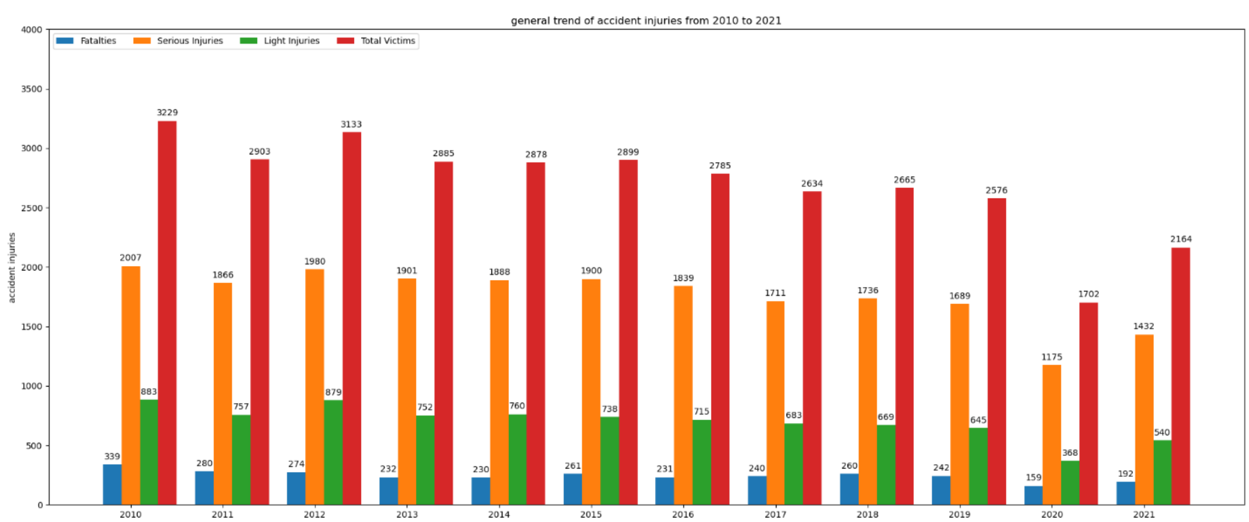
Number of fatalities, serious Injuries, light Injuries and total victims respect to **<u>Province Name</u>**



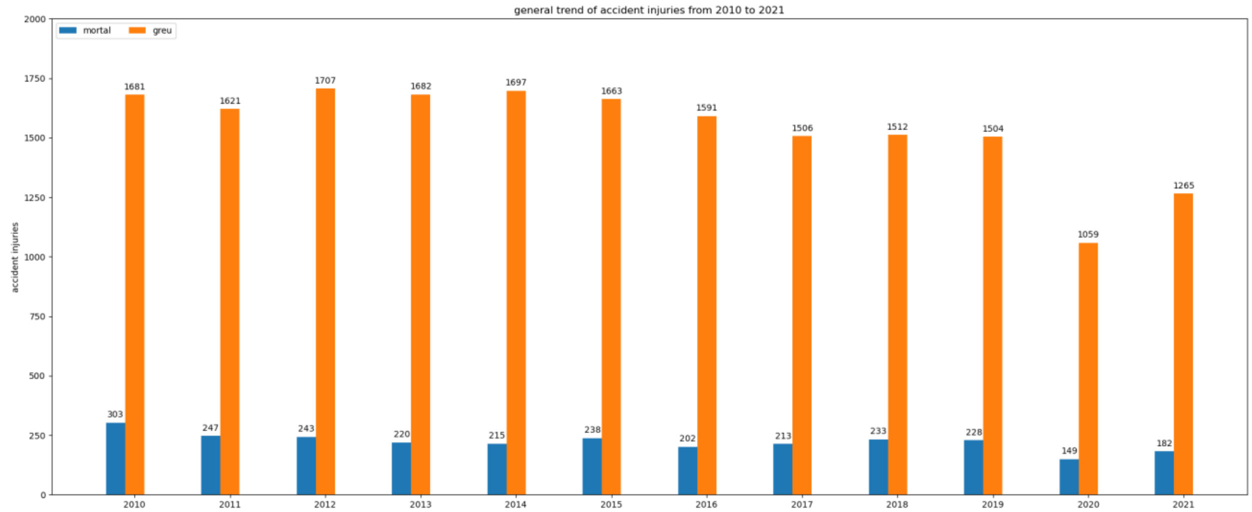
4. **<u>Yearly Trends</u>**

How have traffic accident patterns (frequency, severity) changed yearly from 2010 to 2021?

Number of fatalities, serious Injuries, light Injuries and total victims respect to **<u>Year</u>**



Number of mortal and greu (severity of accident) respect to **<u>Year</u>**



5. **Day and Time Patterns**

On what days of the week and times of day do most accidents occur? Are there notable differences between weekdays and weekends?

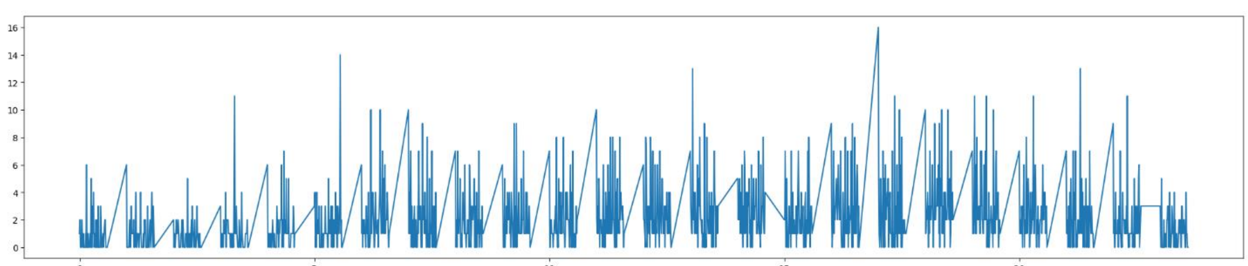
DOw	Serious Inj.
CapDeSetmana	8006
Feiners	13118

DOw	Light Inj.
CapDeSetmana	3742
Feiners	4647

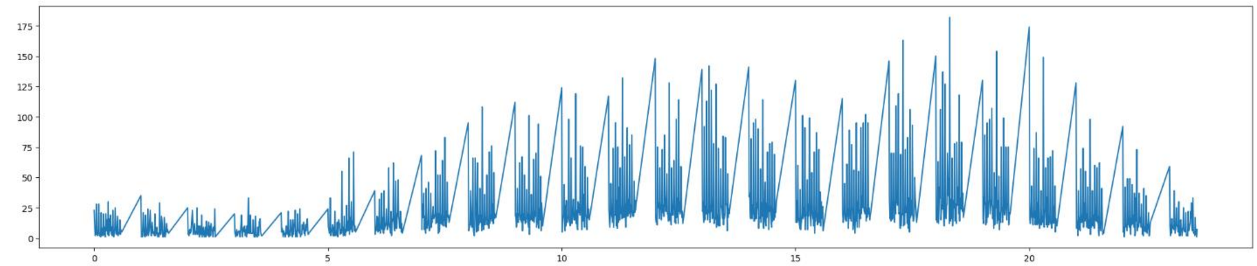
DOw	Total vic.
CapDeSetmana	12942
Feiners	19511

DOw	Fatalties
CapDeSetmana	1194
Feiners	1746

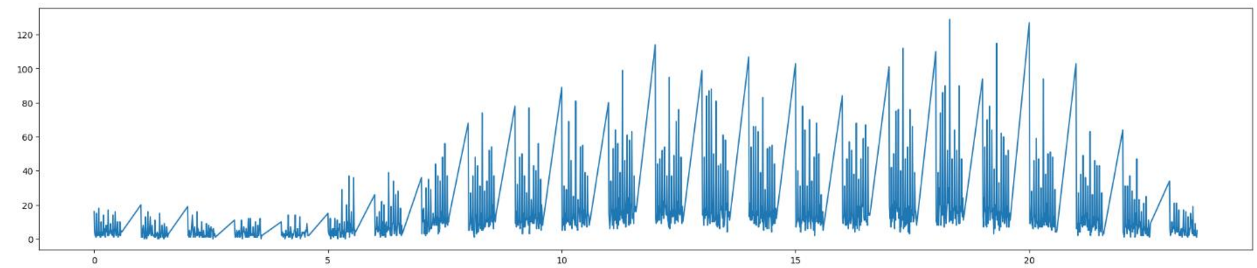
Number of fatalities respect to Hour of Day



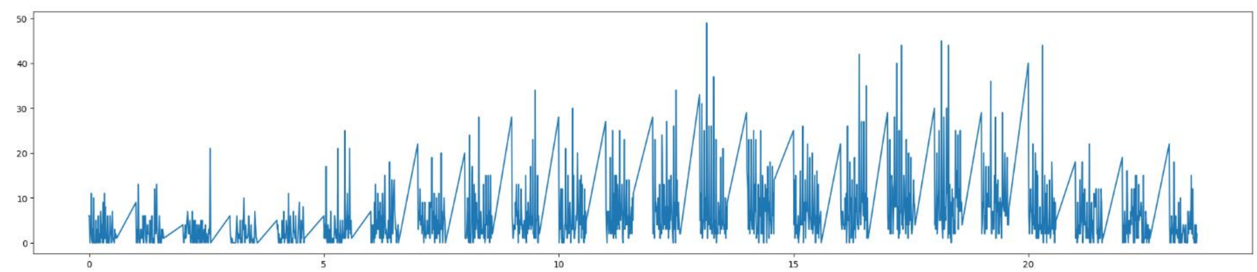
Number of total Victims respect to Hour of Day



Number of serious inj. respect to Hour of Day



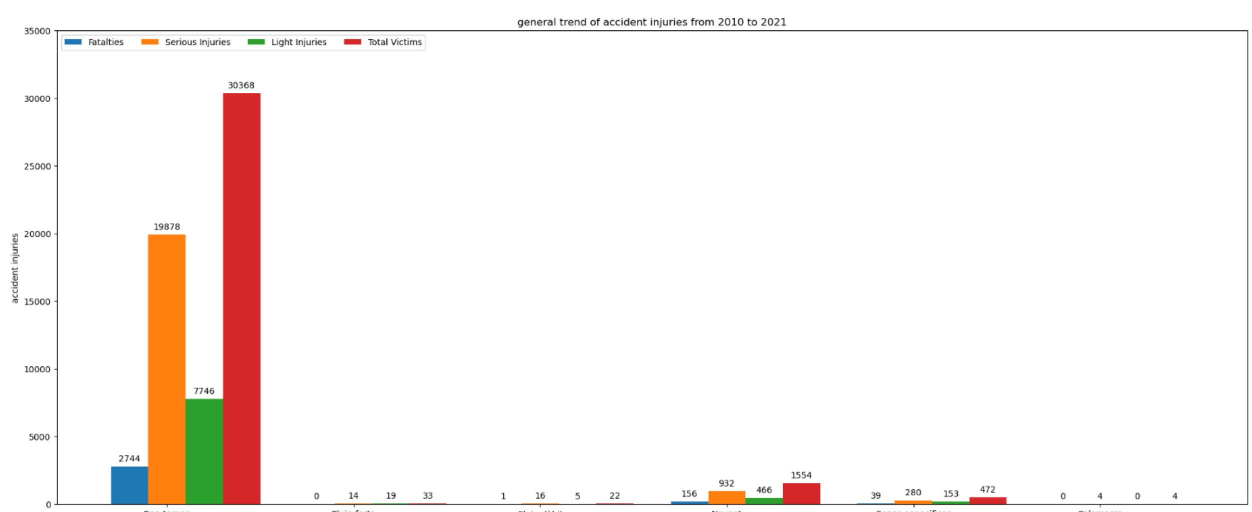
Number of light inj. respect to Hour of Day



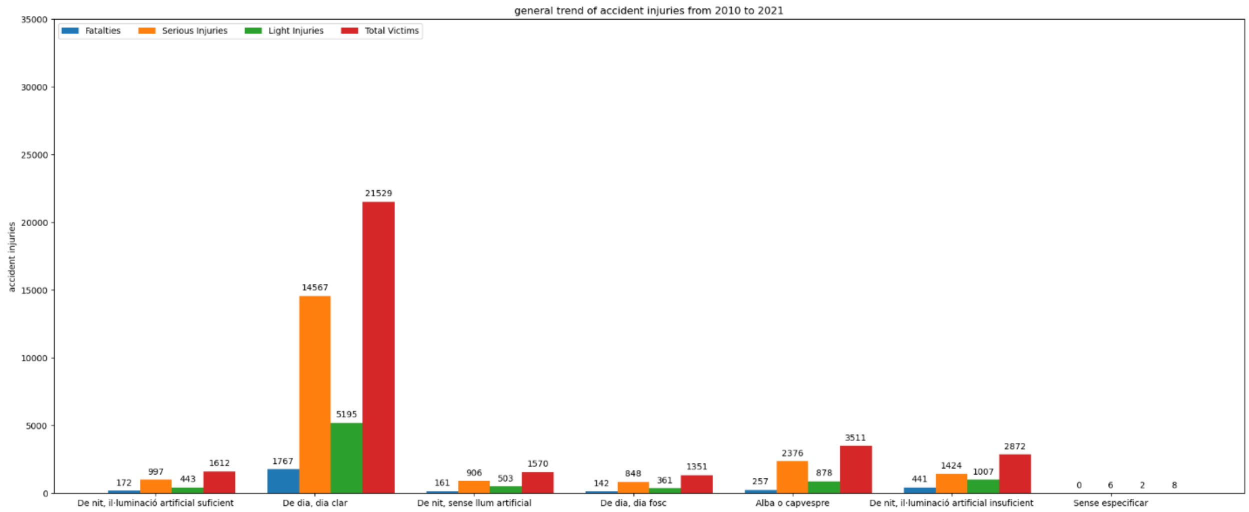
6. **<u>Environmental Impact</u>**

How do different weather conditions affect the likelihood of accidents? Is there a correlation between visibility, road conditions, and accident severity?

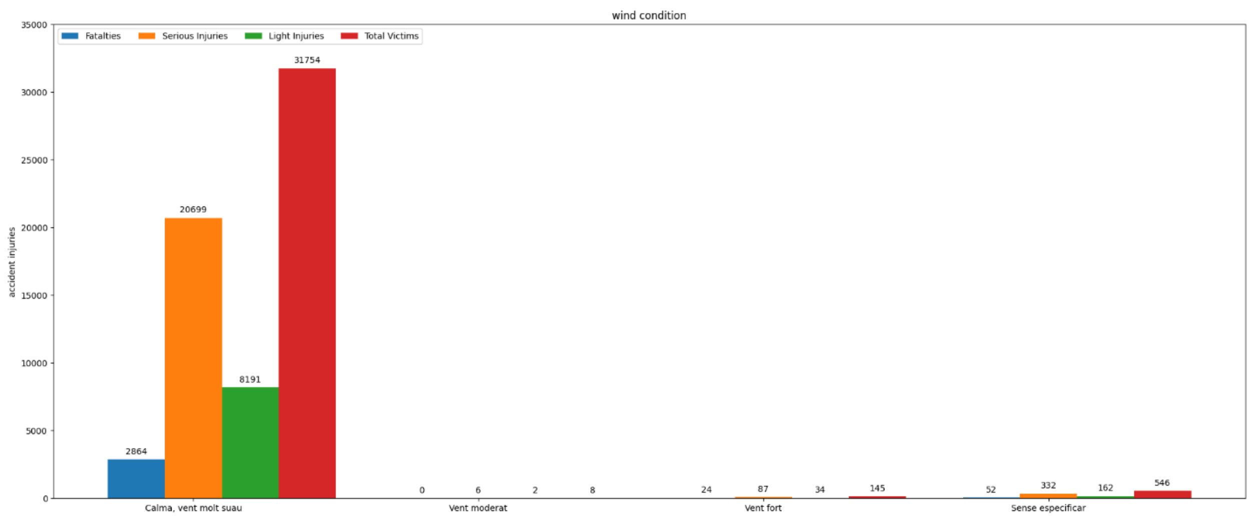
Number of fatalities, serious Injuries, light Injuries and total victims respect to **weather conditions**



Number of fatalities, serious Injuries, light Injuries and total victims respect to **Lighting condition**



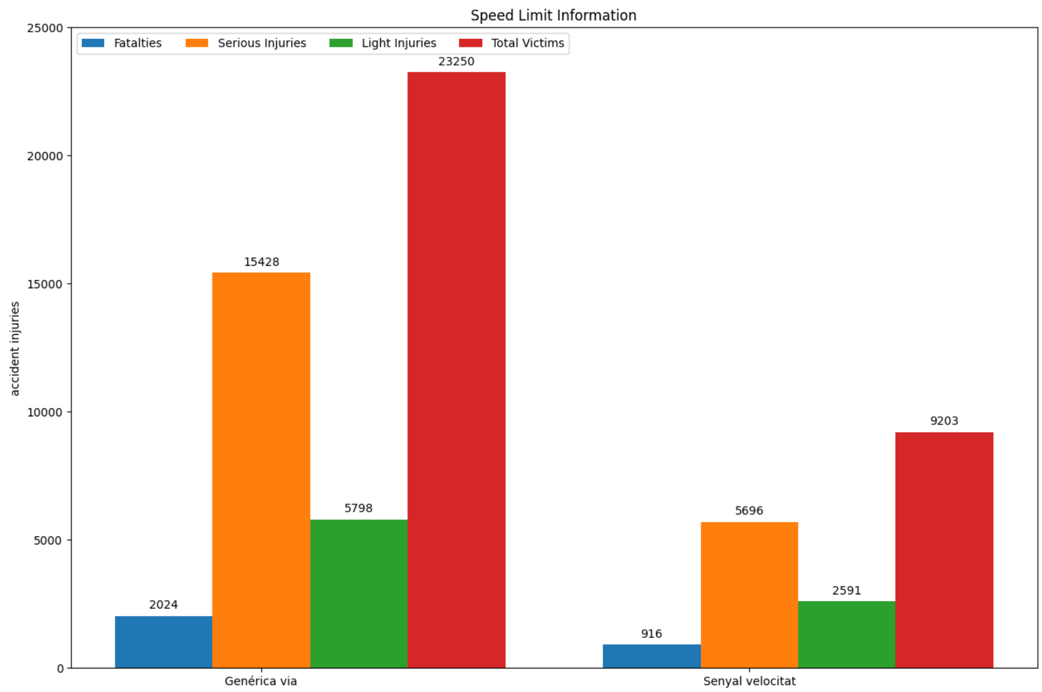
Number of fatalities, serious Injuries, light Injuries and total victims respect to **wind condition**



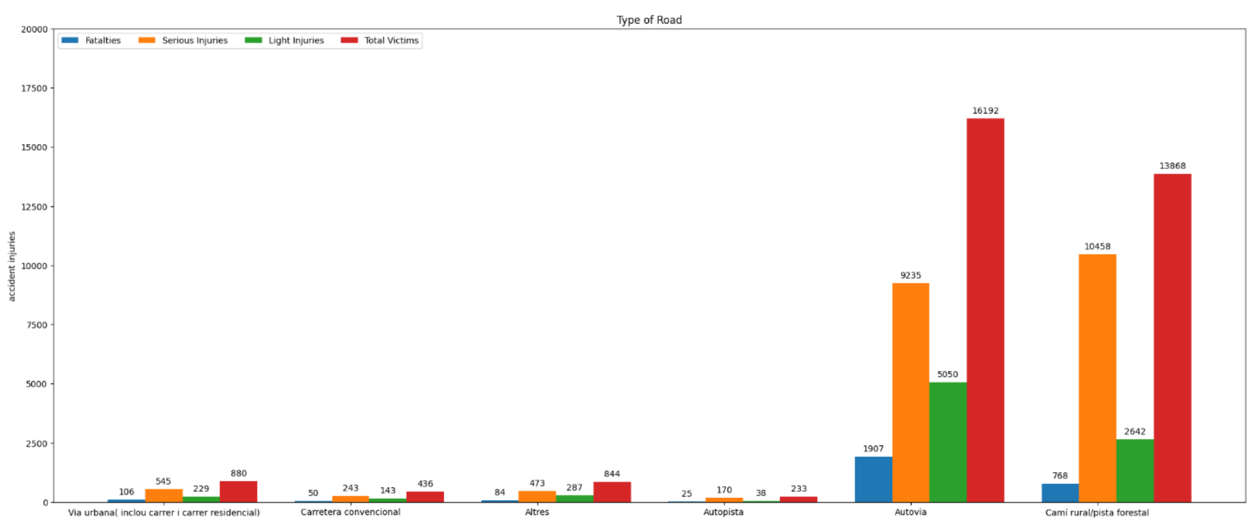
7. **Road and Traffic Features**

What impact do road features (such as speed limits and road types) and traffic density have on the occurrence of accidents?

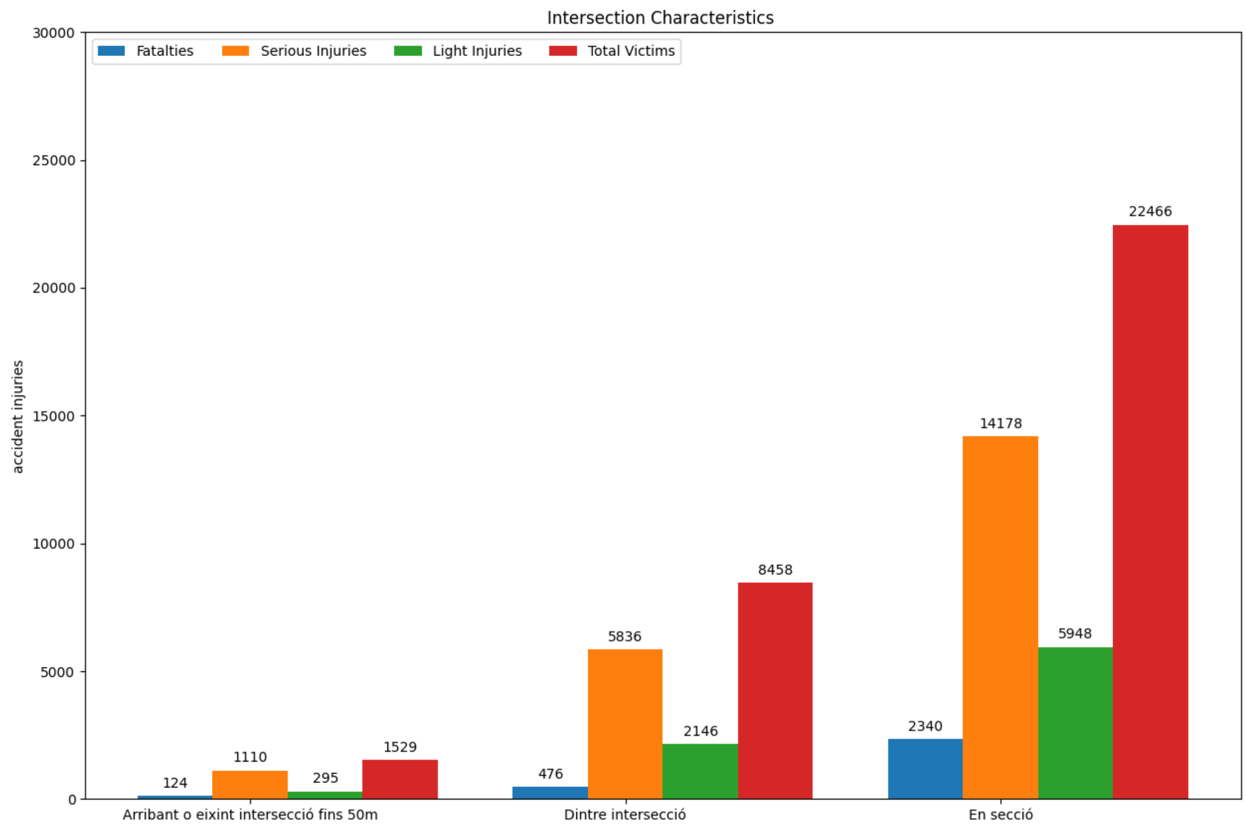
Number of fatalities, serious Injuries, light Injuries and total victims respect to **speed limit information**



Number of fatalities, serious Injuries, light Injuries and total victims respect to **type of road**



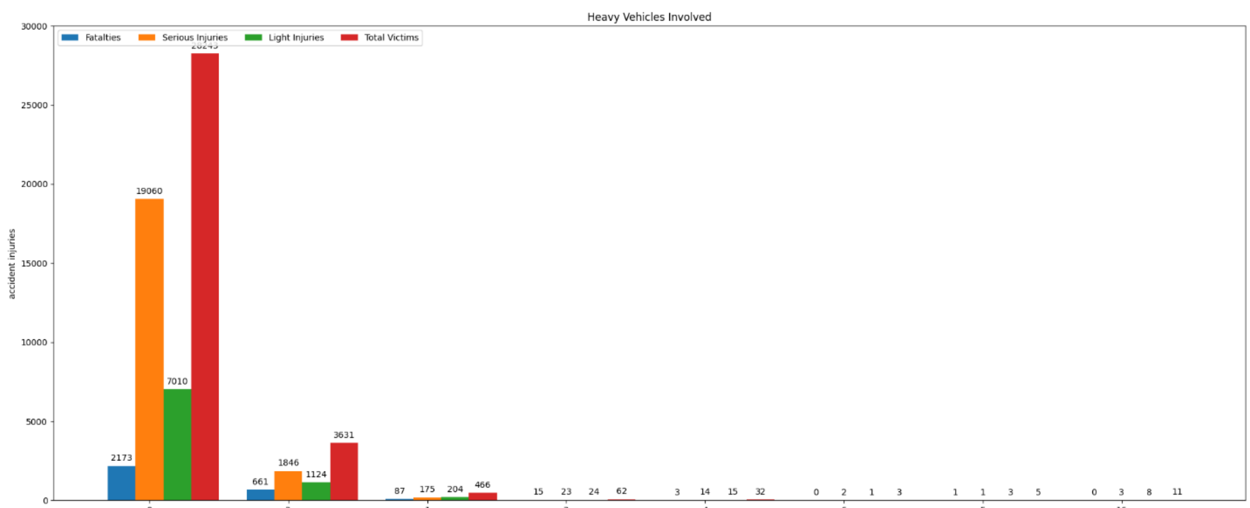
Number of fatalities, serious Injuries, light Injuries and total victims respect to **intersection characteristics**



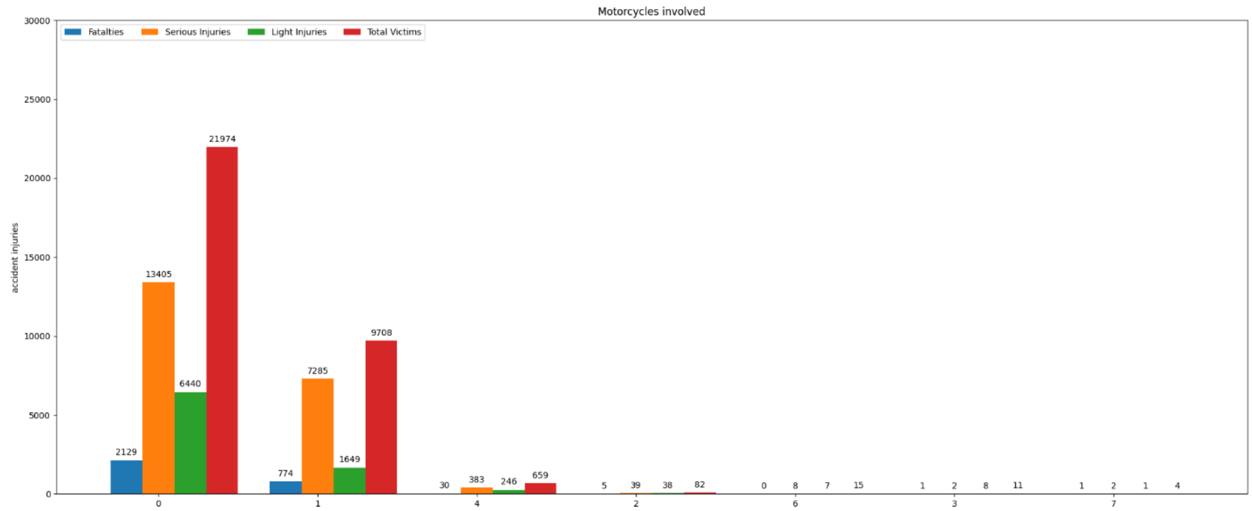
8. **Vehicle Types and Accident Severity**

Does the involvement of specific types of vehicles (like heavy trucks and motorcycles) correlate with more severe accidents?

Number of fatalities, serious Injuries, light Injuries and total victims respect to the num. of **heavy vehicle involved**

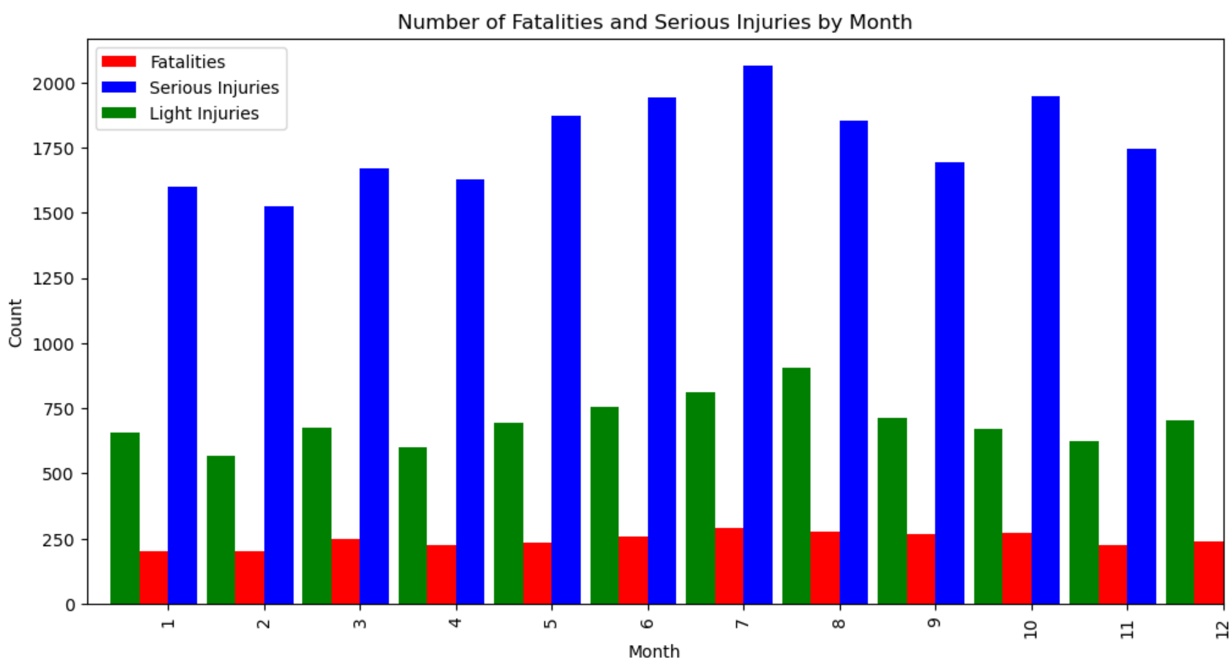


Number of fatalities, serious Injuries, light Injuries and total victims respect to the num. of **motorcycle involved**



9. **Temporal Clustering**

Are there specific periods (months, years) where accident patterns cluster significantly? What might be the causes for these clusters?



Generally, the accident rates are more in months May, June, July and October respect to serious Injuries.

Fatalities rates are relatively the same in all months and light injuries also increases in July.

10. **Time-Series Forecasting**

Based on past trends, create a model to forecast the number of accidents, fatalities, or serious injuries for the upcoming year. Clearly describe the forecasting model you have developed. This should include the type of model, its structure, and any specific features or techniques it utilizes. Discuss the factors that influenced your decision, such as the model's accuracy, efficiency, suitability to the data characteristics, or its ability to handle the complexities of the dataset.

For this purpose, after loading the data, we first convert the ***Date*** column format from dd/mm/yyyy to yyyy/mm/dd to be sorted based on days not years then we grouped '***Date'*** base on each target values (fatality, light injuries, serious injuries) and the output were 3 series with the length of 4308. Each series indicates in each day of the Date how many fatalities /light injuries/ serious injuries have happened.

Before applying any model to data we needed to be sure that our data is stationary which means it's mean and variance is consistent during the time. The ***adfuller*** python library did this for us. It takes the time series data and analyzes it and returns p-value and ADF Statistic which indicates whether the ts_data is stationary or not.

```
In [103]: from statsmodels.tsa.stattools import adfuller
          # Assuming your time series data is in the variable 'ts_data'
          result_fat = adfuller(train_fat)
          print('ADF Statistic: %f' % result_fat[0])
          print('p-value: %f' % result_fat[1])
          result_si = adfuller(train_si)
          print('ADF Statistic: %f' % result_si[0])
          print('p-value: %f' % result_si[1])
          result_li = adfuller(train_li)
          print('ADF Statistic: %f' % result_li[0])
          print('p-value: %f' % result_li[1])

          ADF Statistic: -56.895906
          p-value: 0.000000
          ADF Statistic: -57.681715
          p-value: 0.000000
          ADF Statistic: -58.753172
          p-value: 0.000000
```

For three series the ADF Statistic indicate that the time series is stationary. The p-value is very less than the significance level of 0.05, which means that we can reject the null hypothesis that the series is non-stationary [6]. In other words, the time series does not exhibit a unit root, and it is stationary. This result is consistent with the ADF test, which is used to test the null hypothesis that a time series has a unit root and is non-stationary.

Based on studies [1],[2],[3], ARIMA time series model outperforms the other model in accidents and road safeties problems. Therefore, in this work we used ARIMA model to predict the upcoming year. In mentioned studies the ARIMA parameters (p,q,d) which are [4,5] :

- **p** represents the autoregressive term, which incorporates the effect of past values into the model. It signifies the number of lag observations included in the model.
- **d** denotes the differencing term, which is the number of differences needed to make the time series data stationary. It reflects the degree of differencing.
- **q** stands for the moving average term, which accounts for the influence of past white noise or error terms. It indicates the size of the moving average window.

are tuned to (1,1,1) and (1,0,1). In addition to these parameters, we obtained a few other parameters using ***auto_arima*** library in python. This library takes a min and max range for each of these parameters and explore all the possible value to gain the best accuracy. ***auto_arima*** suggests these parameters for training ARIMA model:

given parameter to auto_arima: start_p = 1, start_q=1, max_p = 10, max_q = 10, seasonal=False

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 3446
Model:              SARIMAX(10, 1, 0)   Log Likelihood               -4921.105
Date:                Mon, 29 Jan 2024   AIC                           9864.209
Time:                        19:45:26   BIC                           9931.801
Sample:                             0   HQIC                          9888.351
                               - 3446
Covariance Type:                  opg
==============================================================================
```

given parameter to auto_arima: start_p = 1, start_q=1, max_p = 10, max_q = 10, seasonal=True, m=7

```
                                     SARIMAX Results
=========================================================================================
Dep. Variable:                                 y   No. Observations:                 3446
Model:             SARIMAX(6, 1, 0)x(2, 0, 0, 7)   Log Likelihood               -4880.219
Date:                           Mon, 29 Jan 2024   AIC                           9778.438
Time:                                   19:42:42   BIC                           9833.740
Sample:                                        0   HQIC                          9798.190
                                          - 3446
Covariance Type:                             opg
=========================================================================================
```

given parameter to auto_arima: start_p = 1, start_q=1, max_p = 5, max_q = 5, seasonal=False

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 3446
Model:               SARIMAX(5, 1, 0)   Log Likelihood               -5053.277
Date:                Mon, 29 Jan 2024   AIC                          10118.555
Time:                        19:46:52   BIC                          10155.423
Sample:                             0   HQIC                         10131.723
                               - 3446
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
```

In order to examine the accuracy of each model on our time series data, we first split each series into train and test with 0.8 and 0.2 portion of primary data, respectively. We trained the model on train part and then test it on the test part to see the performance.

Evaluation metrics: we use three metrics to evaluate the model

- mean square error (MSE)
- mean absolute error (MAE)
- root mean square error (RMSE)

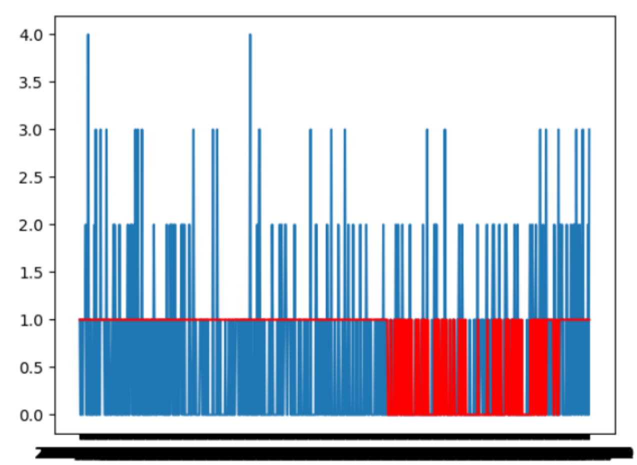
RESULTS

Fatality time series: as shown in the picture below, the index of this series indicates the Date which is sorted by day and the values are the number of fatalities in each day

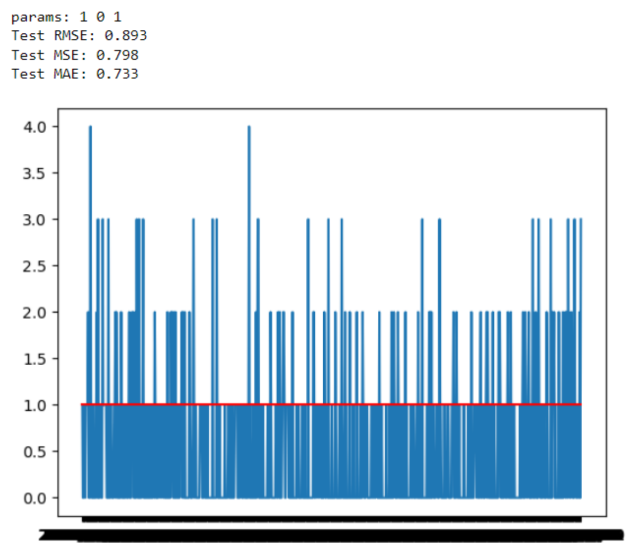
```
Fatalties_grouped_by_date

Date
2010/01/01    0
2010/01/02    0
2010/01/03    0
2010/01/04    3
2010/01/05    1
             ..
2021/12/27    1
2021/12/28    0
2021/12/29    2
2021/12/30    0
2021/12/31    3
Name: Fatalties, Length: 4308, dtype: int64
```

p=1, q=1, d=1:

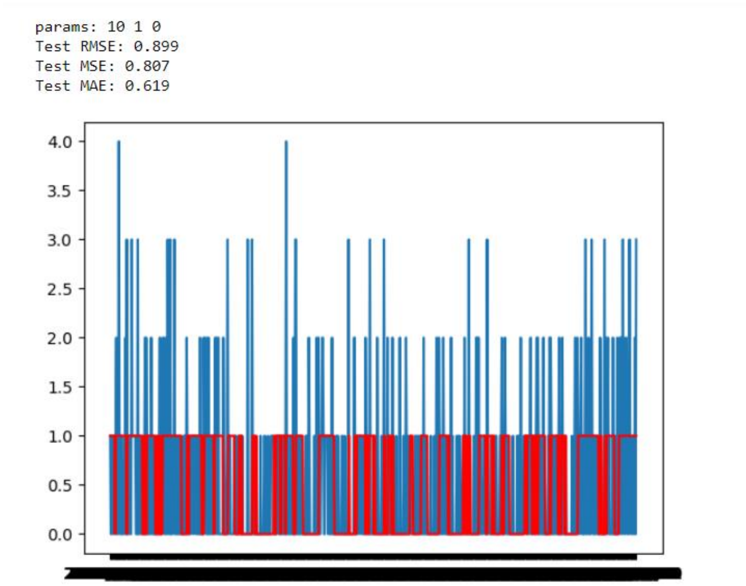
```
params: 1 1 1
Test RMSE: 0.905
Test MSE: 0.819
Test MAE: 0.691
```



p=1 , q=0, d=1 :

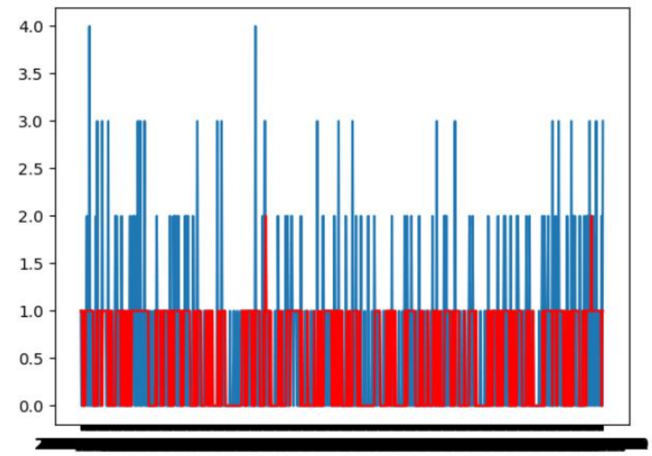


p=10, q=1, d=0:



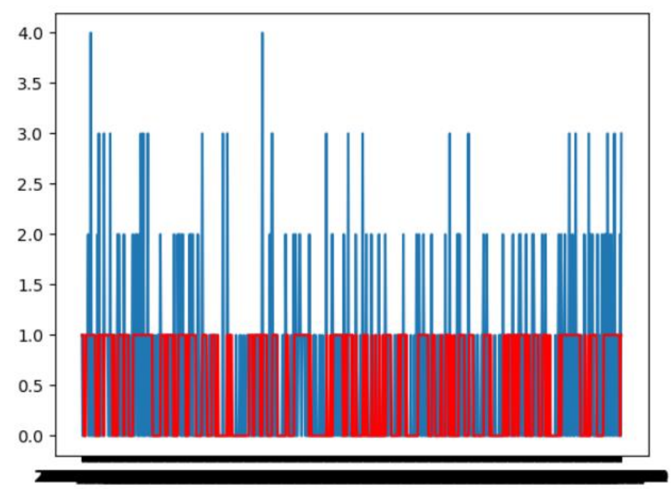
p=5, q=1, d=0:

```
params: 5 1 0
Test RMSE: 0.917
Test MSE: 0.841
Test MAE: 0.630
```



p=6, q=1, d=0:

```
params: 6 1 0
Test RMSE: 0.914
Test MSE: 0.835
Test MAE: 0.636
```

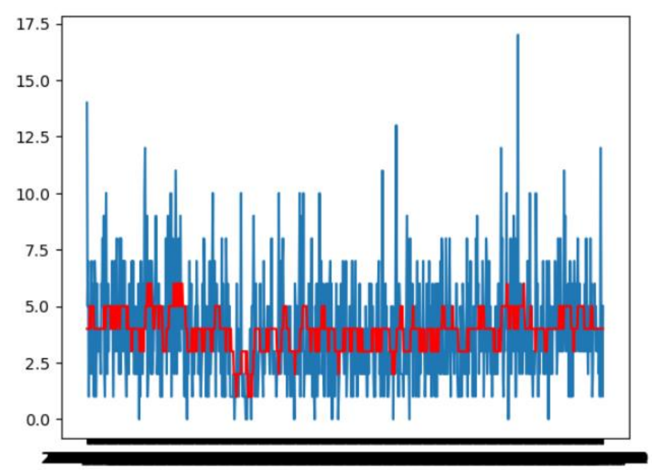


Serious Injuries time series: as shown in the picture below the index of this series indicates the Date which is sorted by day and the values are the number of Serious Inj. in each day

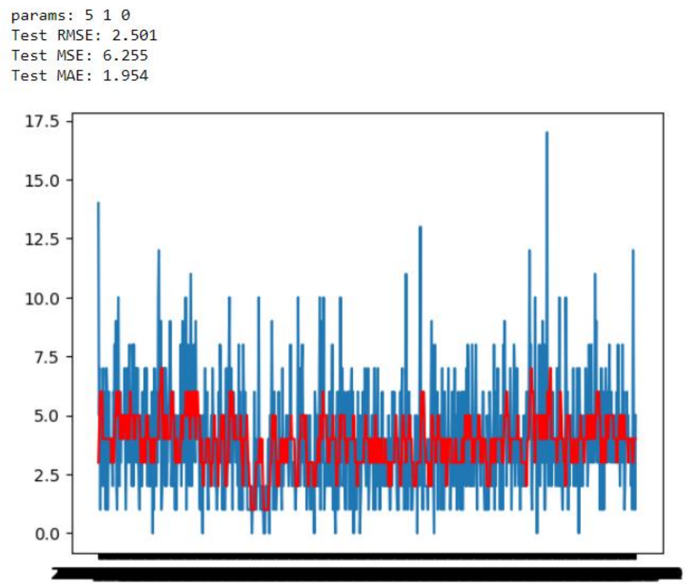
```
Serious_Injuries_grouped_by_date

Date
2010/01/01     7
2010/01/02     7
2010/01/03    10
2010/01/04     2
2010/01/05     2
              ..
2021/12/27    12
2021/12/28     4
2021/12/29     5
2021/12/30     1
2021/12/31     5
Name: Serious Injuries , Length: 4308, dtype: int64
```

p=10, q=1, d=0:

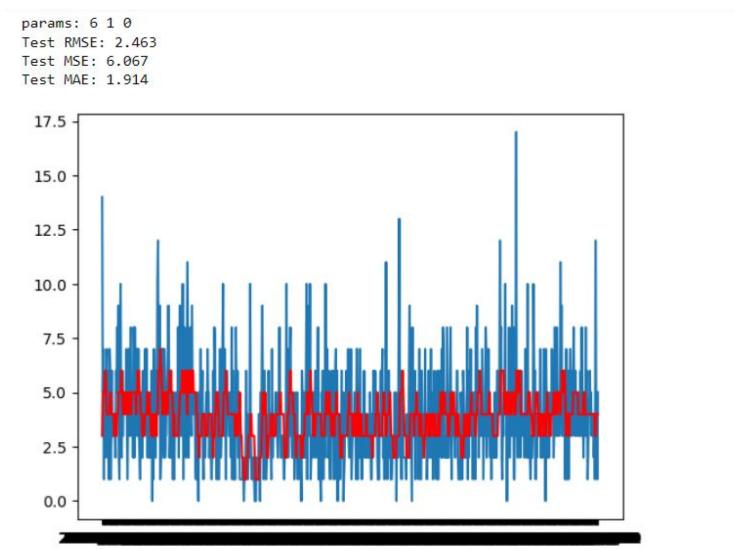
```
params: 10 1 0
Test RMSE: 2.415
Test MSE: 5.832
Test MAE: 1.897
```



p=5, q=1, d=0:



p=6, q=1, d=0:

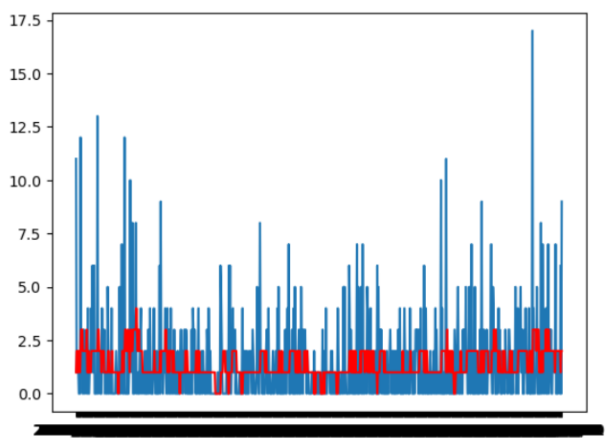


Light Injuries time series: as shown in the picture below, the index of this series indicates the Date which is sorted by day and the values are the number of Light Inj. in each day

```
Light_Injuries_grouped_by_date

Date
2010/01/01    3
2010/01/02    4
2010/01/03    5
2010/01/04    3
2010/01/05    3
             ..
2021/12/27    2
2021/12/28    0
2021/12/29    6
2021/12/30    0
2021/12/31    9
Name: Light Injuries , Length: 4308, dtype: int64
```

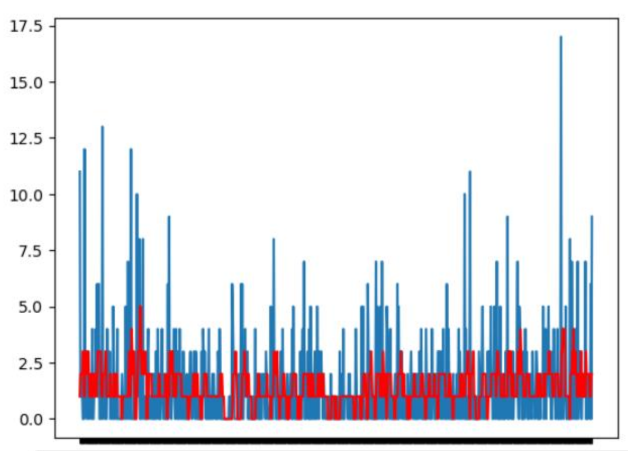
p=10, q=1, d=0:

```
params: 10 1 0
Test RMSE: 2.042
Test MSE: 4.172
Test MAE: 1.383
```

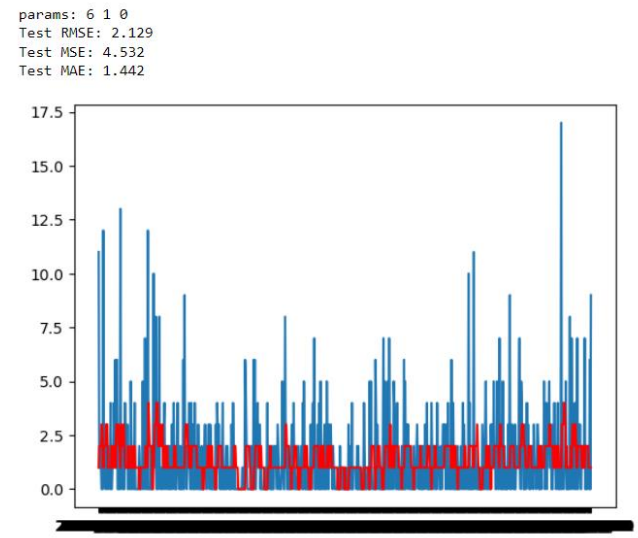


p=5, q=1, d=0:

```
params: 5 1 0
Test RMSE: 2.131
Test MSE: 4.542
Test MAE: 1.437
```



p=6, q=1, d=0:



Based on the results on test data for different series the best parameter is (10,1,0) because it leads to lower mse and mae. Therefore to predict the data of the upcoming year (365 days) we used ARIMA(10,1,0). The results are stored in 3 csv file with the name fat_pred.csv, si_pred.csv and li_pred.csv which are related with fatalities, serious Injuries and light injuries.

Regression with random forest and svm :

Preprocessing and model training

After importing the data with *pandas.read_csv* , we use the following command to observe al NaN values.

the output of this command shows which columns have nan values and how many NaNs they have

```
In [6]: dataset.isnull().sum()
```

```
Kilometer Point                      1
Road Speed Limit                  2642
Surrounding Environment             32
Special Lane Presence             1349
Special Traffic Measures            40
Traffic Regulation and Priority  14970
Direction of Road                 3496
Subtype of Road Section          14212
Road Ownership                   10712
Road's Altimetric Layout          7637
```

Except of two columns (***Kilometer Point , Road Speed Limit***) , all other columns with NaN, have categorical values. To remove these nan value from the dataset first for all categorical columns the NaN values replaced with ‘ nan_values ’ to be able then easily delete these value after one-hot encoding.

Next, in order to convert all categorical features into useful numerical features. First, we counted all unique (cathedral) values of columns to decide what kind of encoding method should be used for them.

the 2 most common encoding methods are one-hot encoding and label encoding. label encoding is used for ordered features and one-hot is good for unordered and independent features. since all of our features (except hour of the day) are unordered we use one-hot encoding. But we first need to be sure that the number of unique values is low. one-hot encoding is the best solution only when the number of unique values is low since it leads to high dimensionality and memory issues. with the large number of unique values, **Binary Encoding** or **Hash Encoding** are often used.

```
In [4]: for (columnName, columnData) in dataset.iteritems():
            print(columnName,len(columnData.unique()),columnData.unique(),  '\n')
```

This command is used to count the unique values of each feature

Based on the output of this command these features were chosen to be encoded with one-hot encoding

'Area', 'Accident with Hit and Run', 'Fog Presence ', 'Surrounding Environment', 'Special Lane Presence', 'Special Traffic Measures', 'Weather Conditions ', 'Special Road Functions ','Influence of Road Objects', 'Severity of Accident ', 'Influence of Fog ', 'Influence of Environment', 'Influence of Traffic', 'Influence of Weather', 'Influence of Wind Intensity ' , 'Influence of Lighting', 'Influence of Special Measures', 'Influence of Road Surface ', 'Influence of Visibility ', 'Intersection Characteristics ', 'Speed Limit Information', 'Lighting Conditions ', 'Traffic Regulation and Priority ', 'Direction of Road', 'Subtype of Accident', 'Subtype of Road Section ','Subzone within Area', 'Road Surface Conditions ', 'Type of Road', 'Road Ownership ', "Road's Altimetric Layout" , 'Wind Conditions ', 'Day of the Week Grouping ', 'Time of Day Grouping ', 'Type of Accident ', 'Day Type'

Among these features that are given to one_hot encoding, *Special Lane Presence* and *Subtype of Accident* have most unique values with 14 and 13 respectively. The other categorical features with high unique values like *Road*, *Municipality Name and County Name* removed because these features are redundant and have no influence on predicting.

To do one-hot encoding we used get_dummies method which is a methos of pandas library

data = pd.get_dummies(dataset, selected_features)

After that, we deleted all columns that have nan_value. Notice that before encoding we replaced all NaN with ‘nan_value’. After one_hot encoding all these values were separated from primary values in a separate column and we easily could delete them. To do so, we used this command:

```
In [5]: data = data.drop(columns = ['Surrounding Environment_nan_value', 'Special Lane Presence_nan_value', 'Special Traffic Measures_nar
                                   'Traffic Regulation and Priority _nan_value', 'Direction of Road_nan_value' , 'Subtype of Road Sectior
                                    , 'Road Ownership _nan_value', "Road's Altimetric Layout_nan_value"])
```

For the 2 numeric columns that include NaN (***Kilometer Point , Road Speed Limit***) NaN values were replaced with the mean of that column.

```
In [6]: data[['Road Speed Limit']] = data[['Road Speed Limit']].fillna(data[['Road Speed Limit']].mean())
        data[['Kilometer Point']] = data[['Kilometer Point']].fillna(data[['Kilometer Point']].mean())
```

Since the two columns include a large range of numeric values we normalize their values with **Z-Score Normalization** method also known as **standardization scaling.** this method centers the values around the mean with a unit standard deviation, resulting in a mean of 0 and a standard deviation of 1.

```
In [8]: columns_to_normalize = ['Kilometer Point', 'Road Speed Limit']
        mean = data[columns_to_normalize].mean()
        std = data[columns_to_normalize].std()
        data[columns_to_normalize] = (data[columns_to_normalize] - mean) / std
```

Then the redundant features like *'Year', 'Date ','Road ', 'Municipality Name ', 'County Name ', 'Province Name '* that have no impact on the result were removed in order to gain more accurate result and also less computation.

```
In [9]: data = data.drop(columns = ['Year', 'Date ','Road ', 'Municipality Name ', 'County Name ', 'Province Name '])
```

And then we separate the target value including *'Fatalties', 'Serious Injuries ', 'Light Injuries ', 'Total Victims '* from the feature values.

```
In [13]: target_columns = ['Fatalties', 'Serious Injuries ', 'Light Injuries ', 'Total Victims ']
         target = data[target_columns]
         data = data.drop(columns = target_columns)
```

Now preprocessing process is done and it's time to split train and test data. With *train_test_split* method we allocate 0.8 of data to train and 0.2 of data to test data

```
In [40]: print(X_train.shape,X_test.shape, y_train.shape,y_test.shape)

         (13542, 173) (4233, 173) (13542, 4) (4233, 4)
```

The size of train and test data

In order to identify important and influential features we used random forest regressor feature_importance [1]. Since we have three target values (i.e. *'Fatalties', 'Serious Injuries ', 'Light Injuries'*), we calculated important features respect of each of these targets separately:

```
In [15]: regr = RandomForestRegressor(max_depth=4, random_state=1)
         regr.fit(X_train, y_train['Fatalties'])
         imp_feature_fatalties = regr.feature_importances_

         # sorted(imp_feature_rfr,reverse=True )
         feature_importance_fatalties = pd.DataFrame({
             'Feature': X_train.columns,
             'Importance': imp_feature_fatalties
         })

         # Select the top N most important features
         top_n = 30  # For example, select the top 5 most important features
         top_features_fatalties = feature_importance_fatalties.nlargest(top_n, 'Importance')
         print(top_features_fatalties)
```

```
In [16]: regr = RandomForestRegressor(max_depth=4, random_state=1)
         regr.fit(X_train, y_train['Serious Injuries '])
         imp_feature_serious_injuries = regr.feature_importances_

         # sorted(imp_feature_rfr,reverse=True )
         feature_importance_serious_injuries = pd.DataFrame({
             'Feature': X_train.columns,
             'Importance': imp_feature_serious_injuries
         })

         # Select the top N most important features
         top_n = 30  # For example, select the top 5 most important features
         top_features_serious_injuries = feature_importance_serious_injuries.nlargest(top_n, 'Importance')
         print(top_features_serious_injuries)
```

```
In [17]: regr = RandomForestRegressor(max_depth=4, random_state=1)
         regr.fit(X_train, y_train['Light Injuries '])
         imp_feature_light_injuries = regr.feature_importances_

         # sorted(imp_feature_rfr,reverse=True )
         feature_importance_light_injuries = pd.DataFrame({
             'Feature': X_train.columns,
             'Importance': imp_feature_light_injuries
         })

         # Select the top N most important features
         top_n = 10  # For example, select the top 5 most important features
         top_features_light_injuries = feature_importance_light_injuries.nlargest(top_n, 'Importance')
         print(top_features_light_injuries)
```

These are the top 10 important features that rfr recognized for each target with their importance rate

Fatalities:

Severity of Accident _Accident greu	0.53402
Severity of Accident _Accident mortal	0.43376
Hour of Day	0.00516
Light Vehicles Involved	0.00327
Heavy Vehicles Involved	0.00287
Kilometer Point	0.00214
Direction of Road_Un sol sentit	0.00209
Road Speed Limit	0.00181
Subtype of Accident_Col·lisió frontal	0.00155
Day Type_dg	0.00125

Units Involved	0.00122
Wind Conditions _Vent fort	0.00120
Weather Conditions _Pluja dèbil	0.00083
Bicycles Involved	0.00082

Serious Injuries:

Severity of Accident _Accident greu	0.355884
Severity of Accident _Accident mortal	0.284307
Light Vehicles Involved	0.185400
Subtype of Accident_Col·lisió frontal	0.043756
Pedestrains Involved	0.025954
Units Involved	0.015471
Motorcycles	0.010850
Hour of Day	0.009748
Direction of Road_Un sol sentit	0.008250
Type of Accident _Sortida de la calcada sense ...	0.006639
Heavy Vehicles Involved	0.006219
Type of Road_Autopista	0.004351
Day Type_dill-dij	0.004129
Day Type_dg	0.003674

Light Injuries:

Light Vehicles Involved	0.636904
Units Involved	0.130238
Subtype of Road Section _Giratòria	0.062038
Motorcycles	0.042355
Hour of Day	0.013374
Heavy Vehicles Involved	0.013069
Special Lane Presence_Carril bus	0.012731
Pedestrains Involved	0.008859
Kilometer Point	0.008572
Weather Conditions _Pluja forta	0.005350

<u>Training process with Random Forest regressor:</u>

we used 10 fold cross validation to more accurately train the model. For each of three targets we used different number of top important features (10,20, 30, 40) and no significant differences between the accuracy of each, was observed, so at last 30 important features were chosen for training and predicting.

The parameters of RFR model set as below:

max_depth=3, n_estimators=100, random_state=1

manual 10fold cross validation for Serious Injuries (30 imp features)

In [183]:
```
top_n = 30  # For example, select the top 5 most important features
top_features_serious_injuries = feature_importance_serious_injuries.nlargest(top_n, 'Importance')

kf = KFold(n_splits=10, shuffle=True, random_state=42)
model = RandomForestRegressor(max_depth=3, n_estimators=100, random_state=1)
mse_sum = 0
mae_sum = 0
for train_index, test_index in kf.split(X_train):

    Xv_train, X_val = X_train.iloc[train_index, top_features_serious_injuries.index], X_train.iloc[test_index, top_features_seri
    yv_train, y_val = y_train.iloc[train_index, 1] , y_train.iloc[test_index, 1]
    model.fit(Xv_train,yv_train )
    predictions = model.predict(X_val[top_features_serious_injuries['Feature']])
    rounded_predictions = np.round(predictions)
    mse = np.square(np.subtract(y_val, rounded_predictions)).mean()
    mae =np.absolute(np.subtract(y_val, rounded_predictions)).mean()
    mse_sum += mse
    mae_sum += mae
#     score = rf.score(X_test, y_test)
    print("mse:", mse , 'mae:', mae)
print('mean mse : ', mse_sum/10 , 'mean mae:', mae_sum/10)
y_hat_test = model.predict(X_test[top_features_serious_injuries['Feature']])
rounded_y_hat_test = np.round(y_hat_test)
mse_test_fat = np.square(np.subtract(y_test['Serious Injuries '], rounded_y_hat_test)).mean()
mae_test_fat =np.absolute(np.subtract(y_test['Serious Injuries '], rounded_y_hat_test)).mean()
print('MSE TEST Serious Injuries : ', mse_test_fat , 'MAE TEST Serious Injuries:', mae_test_fat)
```

Evaluation metrics: MSE , MAE

Result:

MSE and MAE on 10 folds validation set, mean of 10 folds and test set:

Fatalities:

```
mse: 0.01004134672179563 mae: 0.008860011813349085
mse: 0.0035440047253396337 mae: 0.0035440047253396337
mse: 0.029533372711163616 mae: 0.02008269344359126
mse: 0.019492025989367986 mae: 0.013585351447135264
mse: 0.013585351447135264 mae: 0.011222681630242174
mse: 0.04311872415829888 mae: 0.021854695806261076
mse: 0.13998818665091553 mae: 0.026580035440047254
mse: 0.009450679267572357 mae: 0.0070880094506792675
mse: 0.015957446808510637 mae: 0.012411347517730497
mse: 0.016548463356973995 mae: 0.009456264775413711
mean mse :  0.030125960183707355 mean mae: 0.013468509604978923
MSE TEST FATALITIES :  0.014883061658398299 MAE TEST FATALITIES: 0.009685802031656036
```

Serious injuries :

```
mse: 0.19314825753101003 mae: 0.1340815121086828
mse: 0.18015357353809805 mae: 0.12463083284111046
mse: 0.17011222681630242 mae: 0.12640283520378026
mse: 0.1813349084465446 mae: 0.13171884229178973
mse: 0.19492025989367986 mae: 0.13112817483756645
mse: 0.17779090372120496 mae: 0.13290017720023628
mse: 0.4813939751919669 mae: 0.13880685174246898
mse: 0.17306556408741877 mae: 0.1281748375664501
mse: 0.16489361702127658 mae: 0.12470449172576832
mse: 0.18321513002364065 mae: 0.13356973995271867
mean mse :  0.2100028416271143 mean mae: 0.1306118295470572
MSE TEST Serious Injuries :  0.19867706118592016 MAE TEST Serious Injuries: 0.1315851641861564
```

Light Injuries:

```
mse: 0.9438865918487891  mae: 0.3520378027170703
mse: 0.5280567040756055  mae: 0.3142350856467809
mse: 0.538098050797401  mae: 0.3219137625516834
mse: 1.0011813349084466  mae: 0.3821618428824572
mse: 0.8499704666272888  mae: 0.35262847017129356
mse: 0.6455995274660367  mae: 0.3455404607206143
mse: 0.8558771411695215  mae: 0.35499113998818665
mse: 0.7655050206733609  mae: 0.370939161252215
mse: 0.8026004728132388  mae: 0.3557919621749409
mse: 1.1973995271867612  mae: 0.37706855791962174
mean mse 0.812817483756645
mean mae 0.3527308246024864
MSE TEST light Injuries :  0.8169147176943067 MAE TEST light Injuries: 0.36617056461138675
```

<u>Training process with SVM regressor:</u>

Parameter tuning in SVM models is the most important part, because of that we did a greedy search into different values of C for fatalities target to gain the best values and the results are as follows:

```
mse: 0.014883061658398299 | mae: 0.009685802031656036 | C: 0.01 | elapsed_time: 0.484999418258667
mse: 0.014883061658398299 | mae: 0.009685802031656036 | C: 0.1 | elapsed_time: 0.7876248359680176
mse: 0.014883061658398299 | mae: 0.009685802031656036 | C: 1 | elapsed_time: 5.481390476226807
mse: 0.014883061658398299 | mae: 0.009685802031656036 | C: 10 | elapsed_time: 59.00965476036072
```

As can be seen the more C gets higher the more time need for execution, but no differences in accuracy was observed, therefore we chose C=0.1 for the training process with SVR.

The other steps were like before as we did with rfr and the important features were those that with rfr we had chosen.

Results

Fatalities

```
mse: 0.01004134672179563 | mae: 0.008860011813349085 | C: 0.1 | elapsed_time: 0.8355138301849365
mse: 0.0035440047253396337 | mae: 0.0035440047253396337 | C: 0.1 | elapsed_time: 0.8486447334289551
mse: 0.028942705256940343 | mae: 0.019492025989367986 | C: 0.1 | elapsed_time: 0.7368645668029785
mse: 0.019492025989367986 | mae: 0.013585351447135264 | C: 0.1 | elapsed_time: 0.6407837867736816
mse: 0.01299468399291199 | mae: 0.010632014176018901 | C: 0.1 | elapsed_time: 0.9304382801055908
mse: 0.04311872415829888 | mae: 0.021854695806261076 | C: 0.1 | elapsed_time: 1.0706007480621338
mse: 0.13998818665091553 | mae: 0.026580035440047254 | C: 0.1 | elapsed_time: 1.0383474826812744
mse: 0.009450679267572357 | mae: 0.007088009450679267 5 | C: 0.1 | elapsed_time: 1.04744291305542
mse: 0.015957446808510637 | mae: 0.012411347517730497 | C: 0.1 | elapsed_time: 0.9737417697906494
mse: 0.016548463356973995 | mae: 0.009456264775413711 | C: 0.1 | elapsed_time: 1.0244560241699219
Mean Squared Error 10fold: 0.030007826692862706 Mean Absolute Error 10fold: 0.013350376114134265
mse on test: 0.014883061658398299 mae on test: 0.009685802031656036
```

Serious injuries

```
mse: 0.22504430005906675 | mae: 0.1281748375664501 | C: 0.1 | elapsed_time: 3.235952615737915
mse: 0.20141760189013586 | mae: 0.11754282339043119 | C: 0.1 | elapsed_time: 4.542234659194946
mse: 0.20791494388659185 | mae: 0.12285883047844064 | C: 0.1 | elapsed_time: 4.664221286773682
mse: 0.2090962787950384 | mae: 0.132309509746013 | C: 0.1 | elapsed_time: 4.5203166007995605
mse: 0.24571766095688127 | mae: 0.13467217956290609 | C: 0.1 | elapsed_time: 4.626796007156372
mse: 0.22090962787950383 | mae: 0.13112817483756645 | C: 0.1 | elapsed_time: 4.928853750228882
mse: 0.5067926757235677 | mae: 0.1275841701122268 | C: 0.1 | elapsed_time: 4.807656526565552
mse: 0.1854695806261075 | mae: 0.11577082102776137 | C: 0.1 | elapsed_time: 4.71750545501709
mse: 0.18262411347517732 | mae: 0.10933806146572105 | C: 0.1 | elapsed_time: 4.739732027053833
mse: 0.20981087470449172 | mae: 0.12706855791962174 | C: 0.1 | elapsed_time: 4.8174684047698975
Mean Squared Error 10fold: 0.23947976579965619 Mean Absolute Error 10fold: 0.12464479661071386
mse on test: 0.22395464209780297 mae on test: 0.12426175289392866
```

Light injuries

```
mse: 1.1134081512108682 | mae: 0.36207914943886593 | C: 0.1 | elapsed_time: 17.731644868850708
mse: 0.6893089190785587 | mae: 0.325457767277023 | C: 0.1 | elapsed_time: 17.13651752471924
mse: 0.6981689308919079 | mae: 0.3520378027170703 | C: 0.1 | elapsed_time: 16.717671394348145
mse: 1.1535735380980507 | mae: 0.38924985233313647 | C: 0.1 | elapsed_time: 16.39486789703369
mse: 0.9663319551092735 | mae: 0.3614884819846427 | C: 0.1 | elapsed_time: 16.4251925945282
mse: 0.7950383933845245 | mae: 0.370939161252215 | C: 0.1 | elapsed_time: 17.30726981163025
mse: 1.0005906674542233 | mae: 0.3638511518015357 | C: 0.1 | elapsed_time: 17.042555809020996
mse: 1.0242173656231541 | mae: 0.4146485528647372 | C: 0.1 | elapsed_time: 15.24735140800476
mse: 1.0360520094562649 | mae: 0.38475177304964536 | C: 0.1 | elapsed_time: 16.41710877418518
mse: 1.41903073286052 | mae: 0.4107565011820331 | C: 0.1 | elapsed_time: 16.906408071517944
Mean Squared Error 10fold: 0.9895720663167346 Mean Absolute Error 10fold: 0.37352601939009045
mse on test: 1.0370895346090243 mae on test: 0.39782660051972596
```

Conclusion:

in this study different model have been investigated for road safety dataset of Calanuya, time series random forest regressor and support vector machine regressor. The dataset had 3 independent target that for each of them a model has been trained separately. Base on the results and the studies, the rfr models outperforms the svr models for 3 types of targets and also is more efficient for time execution, since the time complexity of svr is very dependent to its parameters. Time series model are the best when there is no information about the datasets but the slot of time. However, choosing the best model and tuning the parameters in time series is very crucial.

For this Calanuya dataset, we only examined ARIMA model based on our studies and the results on different parameters indicates that the model needs to be improved and couldn't well adapted on the dataset.

References:

[1] https://www.linkedin.com/pulse/applied-time-series-forecasts-traffic-accidents-injuries-rami-kanaan/

[2] Getahun, K.A. Time series modeling of road traffic accidents in Amhara Region. *J Big Data* **8**, 102 (2021). https://doi.org/10.1186/s40537-021-00493-z

[3] Nassiri H, Mohammadpour SI, Dahaghin M. Forecasting time trend of road traffic crashes in Iran using the macro-scale traffic flow characteristics. Heliyon. 2023 Mar 11;9(3):e14481. doi: 10.1016/j.heliyon.2023.e14481. PMID: 36967875; PMCID: PMC10036660.

[4] https://www.capitalone.com/tech/machine-learning/understanding-arima-models/

[5] https://people.duke.edu/~rnau/411arim.htm

[6] https://www.machinelearningplus.com/time-series/augmented-dickey-fuller-test/

[7] Megnidio-Tchoukouegno, M.; Adedeji, J.A. Machine Learning for Road Traffic Accident Improvement and Environmental Resource Management in the Transportation Sector. *Sustainability* **2023**, *15*, 2014. https://doi.org/10.3390/su15032014

[8] Silva, Philippe & Andrade, Michelle & Ferreira, Sara. (2020). Machine learning applied to road safety modeling: A systematic literature review. Journal of Traffic and Transportation Engineering (English Edition). 7. 775-790. 10.1016/j.jtte.2020.07.004.

[9] Obasi IC, Benson C. Evaluating the effectiveness of machine learning techniques in forecasting the severity of traffic accidents. Heliyon. 2023 Jul 29;9(8):e18812. doi: 10.1016/j.heliyon.2023.e18812. PMID: 37560691; PMCID: PMC10407198.

[10] Shakil, Ahmed., Md., Akbar, Hossain., Sayan, Kumar, Ray., Mafijul, Islam, Bhuiyan., Saifur, Rahman, Sabuj. (2023). A study on road accident prediction and contributing factors using explainable machine learning models: analysis and performance. Transportation research interdisciplinary perspectives, 19:100814-100814. doi: 10.1016/j.trip.2023.100814