Contents

An Analysis of Traffic Accident Severity in New York

Felipe Da Silva - 3366048, Jared Nichols - 3261253, Samuel Robley - 3391645, Ryan Joseph Tolentino - 3392094, Tim Virga - 3214448

1. Executive summary

The social and economic tolls, along with the clear harmful and potentially deadly consequences, of traffic congestion make the issue an important area for research. This investigation of traffic accident severity in the state of New York aims to determine predictors of high severity accidents. High severity accidents are those which cause large delays to traffic. The investigation found that mean accident duration was longer in 2020 when compared to 2019. It was also discovered that important predictors for accident severity were weather, the time of day and time of year.

2. Introduction

New York state is a state notorious for its traffic congested roads. Commuters rely on the roads to reach their workplaces, homes, commercial needs and a variety of social gatherings and events a with meeting the shipping and delivery needs of the state. Traffic accidents occurring on the streets of New York can lead to further congestion and delays, along with the potential for serious harm or death. The cause of these crashes is often complex, and a study into the crashes in New York city found the top three reasons for accidents were driver distraction, failure to yield right of way and exceeding the speed limit (Shaaban & Ibrahim, 2021). Efforts to improve road safety include ensuring clear road marking, installation of roadside barriers, drainage optimization and intersection channelisation (Wu et al., 2015). Of particular interest to this study, is the effects of traffic accidents on the delay of traffic.

The aims of this investigation are to determine the predictors of high severity accidents using a range of analytic and statistical tools. High severity accidents are defined within the dataset of having large impacts on the delay of traffic. The dataset has a severity index between 1 and 4, with 4 being the largest impact on traffic, with 1 being the least impact. Prior research has shown that traffic congestion has social and economic implications, mainly a loss of productivity (Mohan Rao & Ramachandra Rao, 2012). The implementation of congestion reducing measures must include accommodations for the impacts of traffic accidents. Determining predictors for these high severity incidents is important for the prevention of traffic congestion. It has been found from a review of the literature that an increase in traffic accidents results from increasing levels of congestion or traffic volume (Retallack & Ostendorf, 2019). This cyclic relationship of traffic accidents cause congestion and that congestion increasing the risk of traffic accidents highlights the importance of understanding the compounding factors for traffic congestion.

Through data exploration and statistical analysis, the dataset explored below will attempt to provide inferences on predictors to high severity incidents, as well as provide some recommendations on potential accident severity reduction techniques. This analysis will explore differences in mean accident duration across two different year groups, 2019 and 2020. The effect of weather condition on accident duration will be explored through a one-way analysis of variance. A logistic regression model will be fitted to identify key predictors of accident severity, as well as a range of classification methods to predict accident severity.

3. Dataset description

The dataset for this investigation is a subset of a dataset containing nationwide traffic accident data covering 49 states in the United States (Moosavi, 2021). The larger dataset was originally published in 2019 and has been updated on a number of occasions since. The subset used for this investigation uses data collected specifically in the state of New York. The collected data contains a range of continuous, discrete, and categorical variables describing the various accidents contained in the dataset.

The data ranges from the 1st of January 2019 to the 31st of December 2020 and contains 39537 accidents observations across 47 variables. Each accident record consists of a variety of intrinsic and contextual attributes such as location, time, natural language description, weather, period-of-day, and points-of-interest (Moosavi, 2021).

Table 1 illustrates variables with missing values numbers. Data pre-processing was performed to clean the dataset and remove missing values.

Table 1. Summary of variables with missing values

Variable	Description	Variable type	Missing values
Number	Shows the street number in address field.	Numerical (discrete)	27692
City	Shows the city in address field.	Categorical (nominal)	8
Airport_Code	Denotes an airport-based weather station which is the closest one to location of the accident.	Numerical (discrete)	66
Weather_Timestamp	Shows the time-stamp of weather observation record (in local time).	Date.Time	312
Temperature.F	Shows the temperature (in Fahrenheit).	Numerical (continuous)	390
Wind_Chill.F	Shows the wind chill (in Fahrenheit).	Numerical (continuous)	1463
Humidity	Shows the humidity (in percentage).	Numerical (discrete)	400
Pressure.in	Shows the air pressure (in inches).	Numerical (continuous)	375
Visibility.mi	Shows visibility (in miles).	Numerical (continuous)	406
Wind_Direction	Shows wind direction.	Categorical (nominal)	939
Wind_Speed.mph	Shows wind speed (in miles per hour).	Numerical (continuous)	1188
Precipitation.in	Shows precipitation amount in inches if there is any.	Numerical (discrete)	1943
Sunrise_Sunset	Shows the period of day (i.e. day or night) based on sunrise/sunset.	Categorical (nominal)	8
Civil_Twilight	Shows the period of day (i.e. day or night) based on civil twilight.	Categorical (nominal)	8
Nautical_Twilight	Shows the period of day (i.e. day or night) based on nautical twilight.	Categorical (nominal)	8
Astronomical_Twilight	Shows the period of day (i.e. day or night) based on astronomical twilight.	Categorical (nominal)	8

3.1 Data pre-processing prior to analyses

Table 2 summarises the alterations made to the dataset. After pre-processing and removing the missing values there are 36779 accidents with 29 columns.

Table 2. Alterations to Dataset (pre – processing)

Alterations	Details
Removed unneeded columns	Variables such as ID, Start_Lat, Start_Lng, Start_Lng, End_Lat End_Lng, Description, Number, Street, State, Zipcode, Country, Timezone, Airport_code, Weather_Timestamp, Wind_Direction, Nautical_Twilight,

	Astronomical_Twilight were removed as we assumed they are not needed in the analyses.
Removed observations with missing data	Observations with at least one missing data were removed.
Created new a new version of categorical variable 'Weather Condition'.	Variable 'Weather_Condition' was collapsed into five broad categories (Fair, Cloudy, Fog, Rain and Others), for use in analyses.
Created new numerical variables 'Year', 'Month' and 'Hour'	Variables 'Start_time' and 'End_time' were converted to date.time type to create timestamps (Year, Month and Hour) for use in analysis.
Created new numerical variable 'Duration'.	Variable 'Start_time' was subtracted from 'End_time' in minutes as integer type to create a variable 'Duration' of incident.

4. Descriptives

Summary tables were created for 9 numerical variables which were identified as areas of interest for the investigation. The variables included distance, duration, temperature, wind chill, humidity, air pressure, visibility, wind speed and precipitation. The distance variable is the length of road affected by the accident, in miles. This was calculated using Haversine formula on start and end longitude and latitude co-ordinates collected from Google Bing Maps and MapQuest traffic Application Programming Interfaces (API's). Duration was calculated as the difference between start and end time and is measured in minutes. Temperature shows the outside air temperature in Fahrenheit. Wind chill shows the wind chill in Fahrenheit. Humidity shows the amount of water vapor in the air as a percentage. Air pressure shows air pressure in inches. Visibility shows the furthest distance at which prominent objects can be seen and is measured in miles. Wind speed is in miles per hour. Precipitation measures precipitation amount in inches.

4.1 Summary Tables

	Distance	Duration	Temp	Wind Chill	Humidity	Air Pressure	Visibility	Wind Speed	Precipitation
count	39537	39537	39147	38074	39137	39162	39131	38349	37594
mean	0.654307	209.54703	51.915442	49.177806	65.716253	29.713701	9.040011	9.132535	0.005389
std	1.551176	3479.8111	16.787521	19.667776	20.599997	0.399174	2.668705	5.870588	0.027952
min	0	5	-12	-30.4	13	27.55	0	0	0
25%	0	30	39.9	34	49	29.47	10	5	0
50%	0.192	79	51	51	67	29.74	10	8	0
75%	0.687	144	65	65	84	29.99	10	13	0
max	49.24	224923	96	96	100	30.71	20	40	0.83

4.2 Outlier removal

4.2.a. Traffic incidents with recorded distance value of zero

Data points with a value of zero for distance of road affected were removed, as only non-zero distances are of interest for statistical testing and regression; 9926 observations of this criterion were removed.

4.2.b. Filtering by Interquartile Range

Outliers for numeric values of interest were also identified according to the following criteria:

$$x < Q1 - (1.5 \times IQR) \mid x > Q3 + (1.5 \times IQR)$$

where Q1 is the first quantile, IQR is the interquartile range, and Q3 is the third quantile. The Table below summarizes the number of outliers for each numeric variable of interest.

Numeric variable	Number of outliers	Lowest value outlier	Highest value outlier
Distance	2422	2.209	49.24
Duration	2598	366	224,923

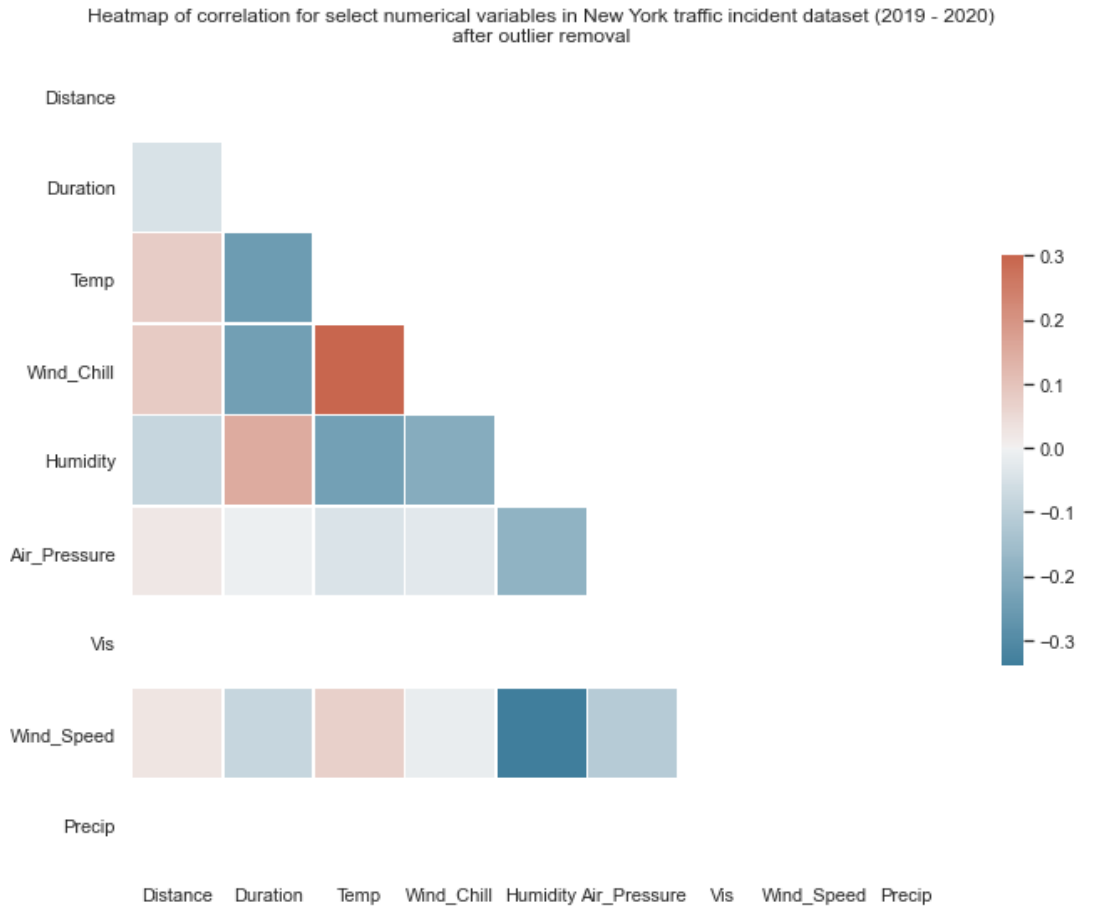
Temperature	42	-12.0	0.0
Wind Chill	16	-30.4	-14.8
Humidity	0	none	none
Air Pressure	467	27.55	28.71
Visibility	5617	0	20
Wind Speed	590	23	40
Precipitation	2637	0.01	0.83

4.2.c. Exclusion of variables

After filtering outliers from the data, visualizing correlations via heatmap reveals that Visibility and Precipitation values are absent from the heatmap (Figure below). Checking the table of associated correlation coefficients (Table) confirms this finding, with both Visibility and Precipitation displaying no correlation coefficients (specifically, NaN values). This is likely due to the nature of the filtering criteria for outliers (according to interquartile range) and the abnormal distribution of Visibility and Precipitation, where approximately 80% of visibility values were one value (Visibility = 10.0 miles) and nearly 90% of Precipitation values being the same as well (Precipitation = 0.0 inches). Due to no variation in the remaining values for Precipitation and Visibility, these numerical variables were decidedly excluded as features of interest for the remaining statistical analyses.

Wind Chill was also excluded as a variable of interest due to its dependence on the variable Temperature (as wind Chill values are derived based off air Temperature).

Exclusion of these variables post removal of outliers should not prove problematic for the remaining analyses due to the reasons mentioned above (abnormal distribution and collinearity), large number of sample size, and incidentally, the stepwise nature of removing outliers implemented in Python. Outliers were sequentially removed in the following order: Distance, Duration, Temperature, Wind Chill, Humidity, Air Pressure, Visibility, Wind Speed, and Precipitation. With Visibility and Precipitation being nearly last in sequence of outlier removal, it is assumed that most of the variation in these values were considered outliers in other variables and highlighting problematic distribution.



	Distance	Duration	Temp	Wind_Chill	Humidity	Air_Pressure	Vis	Wind_Speed	Precip
Distance	1.000000	-0.045107	0.082169	0.082971	-0.081921	0.021078	NaN	0.025168	NaN
Duration	-0.045107	1.000000	-0.251755	-0.240357	0.152462	-0.009756	NaN	-0.080439	NaN
Temp	0.082169	-0.251755	1.000000	0.990589	-0.237391	-0.044315	NaN	0.070578	NaN
Wind_Chill	0.082971	-0.240357	0.990589	1.000000	-0.203822	-0.027404	NaN	-0.012749	NaN
Humidity	-0.081921	0.152462	-0.237391	-0.203822	1.000000	-0.183194	NaN	-0.339667	NaN
Air_Pressure	0.021078	-0.009756	-0.044315	-0.027404	-0.183194	1.000000	NaN	-0.113314	NaN
Vis	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Wind_Speed	0.025168	-0.080439	0.070578	-0.012749	-0.339667	-0.113314	NaN	1.000000	NaN
Precip	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

5. Analysis & results

5.1 Levene's test for homoscedasticity and Welch t-test for comparing means

A Levene's test was used to determine if the variance in traffic accident duration was similar for events occurring in the years 2019 and 2020, prior to comparison of mean duration via Student's *t*-test or Welch *t*-test pending results.

Assumptions for the Levene's test were as follows:

- Independent observations
- Variable in question is quantitative in nature

Both assumptions were met as incidents occurring in the year 2019 are separate from those occurring in the year 2020, and duration of traffic accidents are considered continuous numerical data.

The hypotheses for the Levene's test were as follows:

$$H_0: \sigma_1^2 = \sigma_2^2$$

$$H_1: \sigma_1^2 \neq \sigma_2^2$$

Where σ_1^2 is the variance of the first group (accidents in the year 2019), and σ_2^2 is the variance of the second group (accidents occurring in the year 2020). The null hypothesis, H_0 , was that the variance in accident duration was the same for events occurring in the year 2019 and those in the year 2020. The null hypothesis, H_1 , is that the variance in accident duration was not the same for both years.

Results of the Levene's test indicate that the difference in variance between traffic accident duration occurring in the year 2019 (M = 30.28, SD = 22.44) and duration for those occurring in the year 2020 (M = 129.43, SD = 68.19) was statistically significant ($t(16504) = 3755.33$, $p < .001$), thus homogeneity of variance in duration between the year 2019 and year 2020 group of accidents is rejected.

A Welch's *t*-test was then used to analyse difference in mean accident duration due to non-equal variance between the year 2019 and year 2020 groups. The Welch's *t*-test assumes normal distribution, which is not the case for this traffic dataset as accident duration was right-skewed. However, as more samples in the future are collected this distribution should theoretically approximate normal distribution as proposed by the Central Limit Theorem (CLT). Independence of variables was also assumed, as mentioned before.

The hypotheses for the Welch's *t*-test were as follows:

$$H_0: \mu_1 = \mu_2$$

$$H_1: \mu_1 \neq \mu_2$$

where $\mu_1$is the mean duration of traffic accidents in the year 2019, and μ_2 is the mean duration of accidents in the year 2020. The null hypothesis is that the mean duration of traffic accidents is the same for both year 2019 and year 2020 groups; the alternative hypothesis is that the mean accident duration is different between the year 2019 and year 2020 groups.

After performing the Welch's *t*-test, the mean duration of traffic accidents occurring in the year 2019 (M = 30.28, SD = 22.44) and the mean duration of traffic accidents occurring in the year 2020 (M = 129.43, SD = 68.19) was found to have a statistically significant difference ($t(16504) = -139.46$, $p < .001$). The duration of a traffic accident event, on average, was found to persist about 100 minutes longer in the year 2020 than an event occurring in the year 2019.

5.2 One-way Analysis of variance for comparison of means and Post Hoc Analysis using Tukey's test

To test equality of mean accident duration between groups of different weather categories, a one-way Analysis of variance (ANOVA) test was performed. The assumptions for a one-way ANOVA are similar to the *t*-tests: variable independence, normal distribution, and homogeneity of variance. Variable independence is met as individual incidents are recoded as separate events. Large sample size tends to normality as explained by the CLT. Assumption of homogeneity of variances have been violated, likely due to different distribution caused by outlier removal and truncation of data, as well as disparate sample sizes in the different subgroups of category – however, at this stage, selecting a more suitable method of statistical analysis is beyond the capabilities of the authors.

The hypotheses for the one-way ANOVA were as follows:

$$H_0: \mu_f = \mu_c = \mu_r = \mu_s = \mu_g$$

$$H_1: \neg H_0$$

where μ_f is the mean duration of a traffic accident for the 'fair' weather category, μ_c is mean traffic duration for cloudy category, μ_r is mean traffic duration under 'rain' category, μ_s is the mean accident duration for 'snow' category, and μ_g the mean accident duration for 'fog' category. The null hypothesis is equality of means among the six different subgroups under weather category; the alternative hypothesis is that at least 2 of the 5 groups have different mean traffic accident duration.

The result shows there is a statistically significant relationship between the mean duration of a traffic accident and the weather condition ($F(23380) = 10.5540$, $p < 0.001$). Post Hoc analysis using Tukey's HSD (honestly significant difference) test reveals that there is a statistically significant difference in mean duration for the following weather category pairs:

- 'cloudy' (*M* = 110.48, *SD* = 74.38) and 'other' (*M* = 59.64, *SD* = 51.64) with $p < .001$
- 'fair' (*M* = 109.46, *SD* = 72.16) and 'other' (*M* = 59.64, *SD* = 51.64) with $p < .001$
- 'fog' (*M* = 158.78, *SD* = 102.63) and 'other' (*M* = 59.64, *SD* = 51.64) with $p = .0032$
- 'fair' (*M* = 109.46, *SD* = 72.16) and 'rain' (*M* = 104.21, *SD* = 73.30) with $p < .001$
- 'other' (*M* = 59.64, *SD* = 51.64) and 'rain' (*M* = 104.21, *SD* = 73.30) with $p = .0030$
- 'other' (*M* = 59.64, *SD* = 51.64) and 'snow' (*M* = 121.02, *SD* = 60.24) with $p < .001$

5.3 Kruskal-Wallis H-test and pair-wise Mann-Whitney U-tests for comparison of ranks

A Kruskal-Wallis H-test was performed to determine if the median severity level of a traffic accident was different between 5 groups of weather categories: 'fair', 'cloudy', 'rain', 'snow', and 'fog'. Severity was treated as an ordinal variable ranging from 1 to 4 with increasing importance and/or impact of the accident. Assumptions of the test include sample independence and normal distribution. Both assumptions are met as independent observations of traffic incidents were only associated under one of the four different severity levels, and the large sample size approaches a normal distribution due to the CLT.

The hypotheses for the Kruskal-Wallis H-test were as follows:

$$H_0\colon \tilde{x}_f = \tilde{x}_c = \tilde{x}_r = \tilde{x}_s = \tilde{x}_g$$

$$H_1\colon \neg H_0$$

where $\tilde{x}_f$ is the median severity of a traffic accident for the 'fair' weather category, $\tilde{x}_c$ is the median severity for 'cloudy' weather, $\tilde{x}_r$ is the median severity under 'rain', $\tilde{x}_s$ is the median severity for 'snow' category, and $\tilde{x}_g$ the median severity for the 'fog' category. The null hypothesis is that the median severity is equal for all weather categories; the alternative hypothesis is that the median severity is different for at least 2 of the groups of different weather categories.

The results of the Kruskal-Wallis H-test yielded a significant difference ($H(5, 16504) = 23.29$, $p < .001$) in the median severity levels of traffic incidents between at least two of the six different subcategories of weather classification.

Further analysis using pair-wise Mann-Whitney U-tests to determine which subgroups of weather category had different median severities revealed the following group pairings:

- 'cloudy' ($\tilde{x}$ = 2) and 'fair' ($\tilde{x}$ = 2) with $p = .0068$
- 'rain' ($\tilde{x}$ = 2) and 'snow' ($\tilde{x}$ = 2) with $p = .0053$

The results of the Kruskal-Wallis H-test and pairwise seem to indicate a type I error; however, more formally, the Kruskal-Wallis H-test and Mann-Whitney tests are used to compare ranks and not medians. A more appropriate interpretation for these results would be that samples of accident severity come from different distributions for when comparing 'cloudy-fair' pairing, and 'rain-snow' pairing.

6. Classification models

After having identified a subset of NY traffic incident data, we are seeking to determine the most effective algorithm for classifying traffic incidents into one of the four Severity values.

For this we have computed four classification models; Ordinal Logistic Regression, K-Nearest Neighbors, Decision Tree and Random Forest.

Let's first explore the model interpretations to understand what steps were taken in our analysis, followed by a summary plot at the end of 6. highlighting the best performing model.

6.1 Logistic regression

Logistic regression is a versatile classification algorithm which serves as an extension of the well-known Linear Regression algorithm.

Our logistic regression model (see Appendix A) utilises a 'lbfgs' solver (Limited-memory Broyden–Fletcher–Goldfarb–Shanno) which seeks to minimise the cost function; it approximates the second derivative matrix updates with gradient evaluations and supports multi-label classification as in our case with Severity.

Selecting 'Severity' as our target 'X' and all selected explanatory variables subset from our outlier removal dataframe renamed 'dfpreds', we compute the logistic regression model with train and test sets.

```
# Logistic regression
lr = LogisticRegression(solver='lbfgs', random_state=0, max_iter=10000, multi_class='auto')
lr.fit(X_train,y_train)
y_pred=lr.predict(X_test)

# Get the accuracy score of y pred against y
acc=accuracy_score(y_test, y_pred)

# Append to the accuracy list
accuracy_lst.append(acc)

print("Logistic regression accuracy is: {:.3f}.".format(acc))
```

```
Logistic regression accuracy is: 0.838.
```

Figure 6.1.1: Logistic regression model using multi_class='auto' for multiple class logistic regression

Our Logistic regression accuracy is modelled at 84% accuracy, with a confusion matrix label distribution as follows:

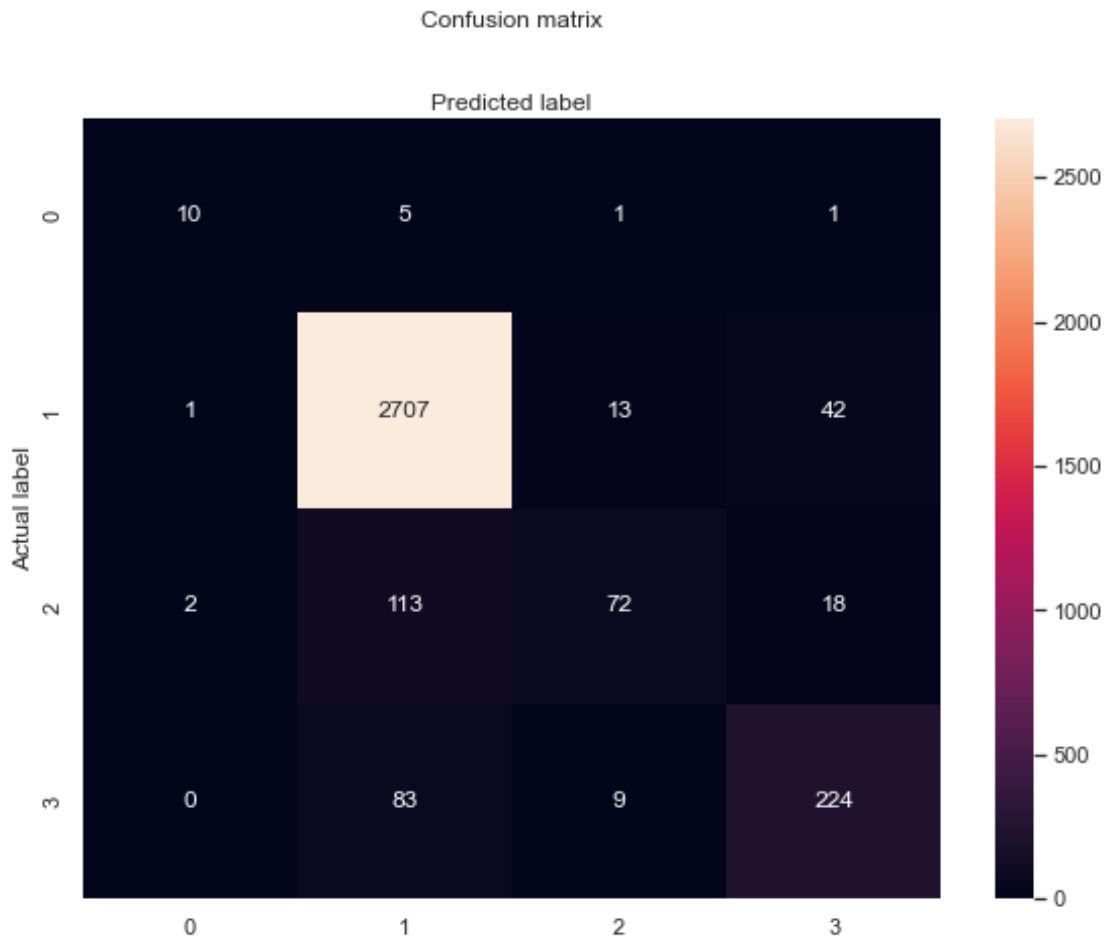


Figure 6.1.2: Confusion matrix for Severity class pred vs actual

For the logistic model we can further report measurement with Accuracy, Precision and Recall as follows:

Accuracy: 0.9127537109966677

Precision: 0.9054324775953668

Recall: 0.9127537109966677

6.2 K-Nearest Neighbors

Our second model chosen (see Appendix B) for classification modelling is K-Nearest Neighbors (KNN). KNN is a supervised learning algorithm that attempts to predict the correct class for test data by employing a Euclidean distance calculation between 'K' number of points in a two-dimensional space.

$$\boldsymbol{Euclidean\ distance}\ (\boldsymbol{a}, \boldsymbol{b}) = \sqrt{(\boldsymbol{a_1} - \boldsymbol{b_1})^2 + (\boldsymbol{a_2} - \boldsymbol{b_2})^2}$$

We first chose a random K value of 8 to get a feel for how the dataset would model and report an accuracy of y_test against y_pred of 85%.

To further improve the KNN model performance on test data, we perform cross-validation to perform many iterations of a range of k-values to determine a mean accuracy across K folds with different selections of test and train data. To do this we leveraged the GridSearchCV library and utilised the GridSearchCV() function with an 'accuracy' scoring parameter and 'cv=5'.

```
#create new a knn model
knn2 = KNeighborsClassifier()
#create a dictionary of all values we want to test for n_neighbors
param_grid = {'n_neighbors': np.arange(1, 20)}
#use gridsearch to test all values for n_neighbors
knn_gscv = GridSearchCV(knn2, param_grid, cv=5, scoring='accuracy')
#fit model to data
knn_gscv.fit(X, y)
print('KNN best score is: {}.'.format(knn_gscv.best_score_))
print('KNN optimal k value is: {}.'.format(knn_gscv.best_params_))
```

```
KNN best score is: 0.7065737655255983.
KNN optimal k value is: {'n_neighbors': 12}.
```

Figure 6.2.1: KNN GridSearchCV method

Over the course of 5 cross validations, our revised KNN model suggested a k value of 12 as being most optimal, with a score of 71%.

6.3 Decision Tree

Our third model used for classification is a Decision Tree (see Appendix C) from the scikit library which is a non-parametric supervised learning method. We've chosen to include a decision tree model because of ease of interpretability and as an alternative view for understanding the important features within our dataset.

The only limitation of decision trees is with a large amount of variables, at deeper nodes they can become quite dense and therefore, require a large resolution to interpret with maximum precision. In our figure below, we are merely showing the key features as part of the decision tree with the full code including a text extract (see Appendix D) of the feature split decisions in the Appendix.

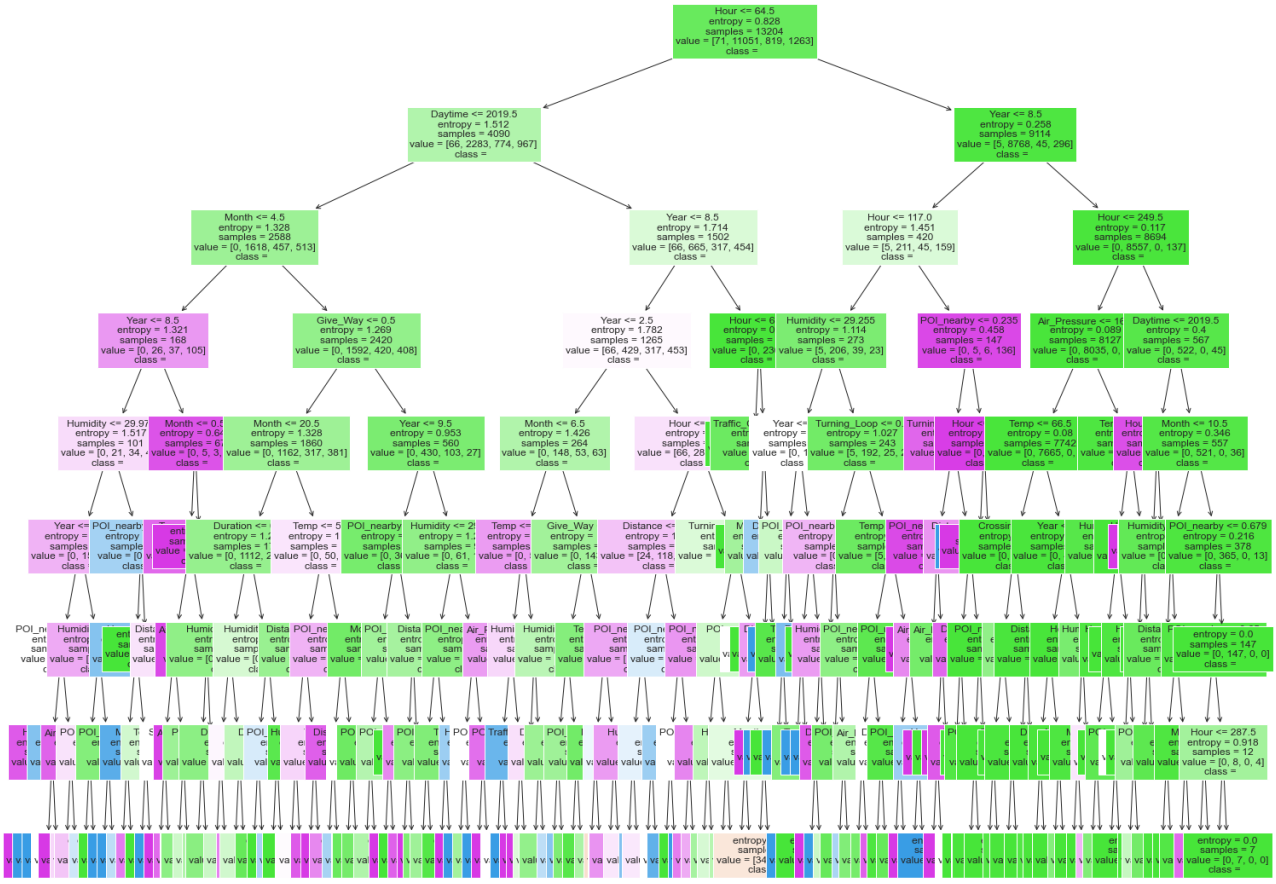


Figure 6.3.1: Decision feature classification visualisation

We note for our Decision Tree model the most important features selected by the algorithm are Hour, Daytime, Year and Month. Our choice for determining node splits was decided by 'criterion=entropy'; where entropy is a measure of information that indicates the disorder of the features with the target, the optimum split is chosen by the feature with less entropy.

6.4 Random Forest

Our final model for classification analysis is Random Forest (see Appendix D) which is a flexible and powerful supervised learning algorithm that is comprised of an appointed number of trees, hence forest. Random forests as the name suggests employ a degree of randomness by selecting random data samples. These random samples combined with a voting mechanic across many trees - in our case, 100 – produce a prediction for each tree using the gini impurity criterion within each tree to select the most optimal split decision, the optimal split decisions are voted on across the total number of trees computed to produce the best possible prediction.

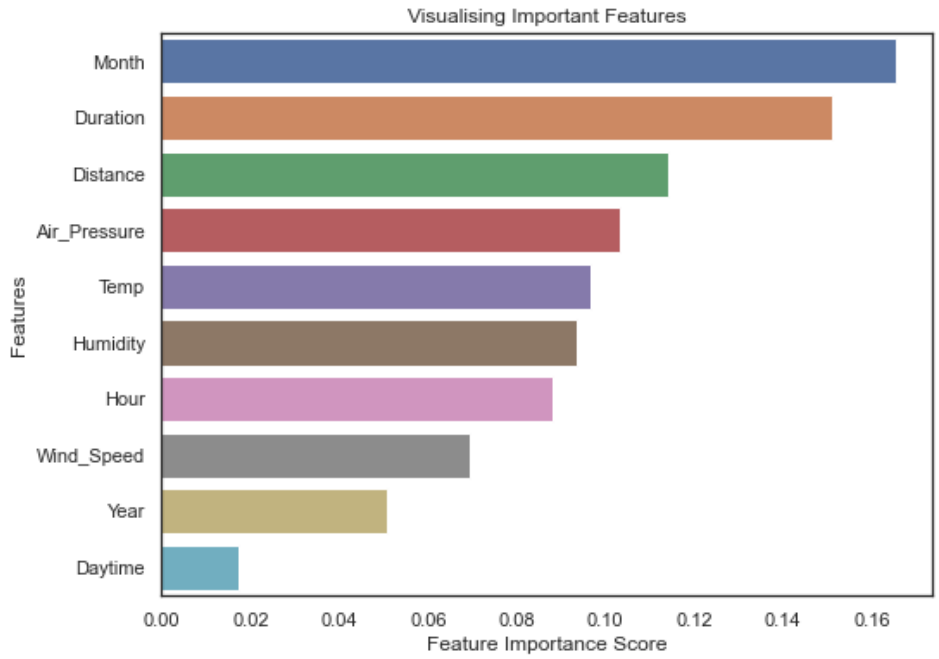


Figure X: Random forest feature importance

We note that somewhat different to our Decision Tree classification, the Random Forest algorithm has highlighted the most important features as being: Month, Duration and Distance. With a high degree of insignificance for Daytime, which was favoured by the previous model.

Random Forest is also capable of feature selection, we used a selection threshold of .3 to subset the original features above to produce a Limited Features Random Forest to further improve model accuracy.

We report our Random Forest algorithm with accuracy score of 91% and our Limited Random Forest algorithm with a score of 92% respectively.

6.5 Algorithm classification accuracy results

In summary, we have produced results for 4 models with our target response variable as Severity. We present a highly accurate Random Forest model with active feature selection that can predict the Severity classification of NY Traffic Incidents based on the explanatory variables (see Appendix D).

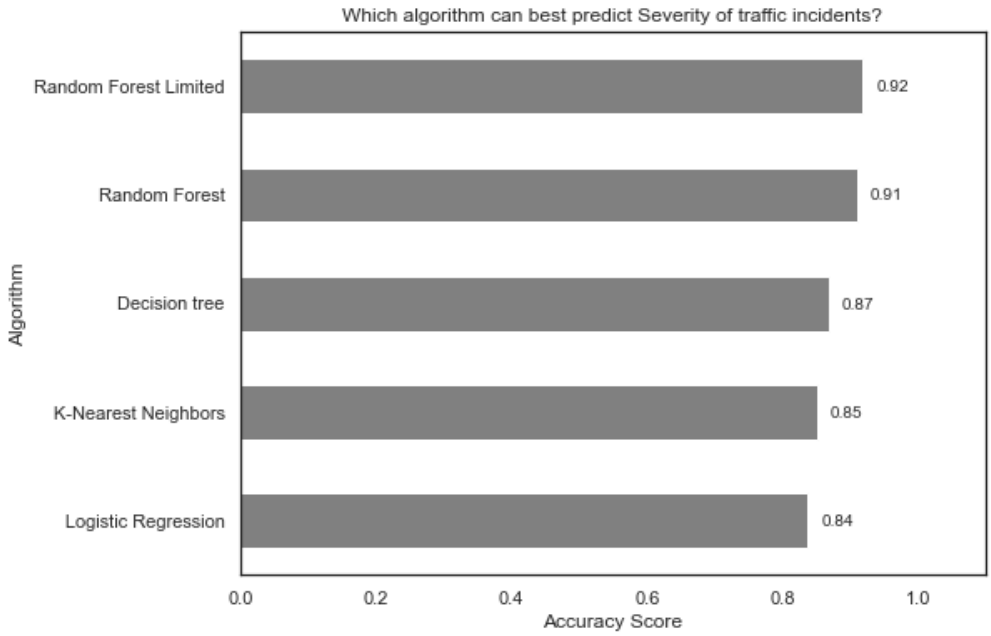


Figure X: Regression algorithm comparison for all selected predictors

7. Discussion

Traffic accidents are not a new problem, it is an ongoing issue with an increase in population and covers pedestrians, motorists, human powered vehicles (bicycles, etc.). There has been a lot of research on the subject of accident severity reduction. A range of research could be considered particularly relevant to the analysis conducted on traffic accident severity based on weather conditions. Research by Green, Heywood & Navarro (2015) into the traffic accident rate in London, has shown the introduction of a congestion tariff to the CBD area has had a considerable impact in reducing the quantity and rate of accidents. This was attributed to a reduction in traffic quantity as well as the tariff potentially creating an incentive for commuters to choose alternative methods of transport. There are ongoing developments in energy-absorbing guard rails on highways, as referenced by Dorokhin et al (2020), the evolution of guardrails to not only absorb the vehicle impact but provide additional features to help reduce the mechanical damage to the vehicle and lower the severity of injury to the occupants provide another approach. Murad & Abaza (2006) approached the issue of wet weather accidents by proposing a surface upgrade at high accident intersections, this would create an increase in skid-resistance creating a potential reduction in accident rates and severity.

8. Conclusion

Based on the analysis completed above, it's clear that there was an increase in mean duration of a traffic accidents delay of traffic. The Welch's T Test found there to be significant difference in the mean accident duration of 2019 and of 2020. The difference between the means of these groups was almost 100 minutes. Weather conditions were found to have an effect on the length of time an accident would impact traffic congestion. The most significant of these differences were between fair weather and the 'other' category, between fair weather and snow, and between snow and the 'other' category. Furthermore, the month of the accident, along with the length of the traffic delay were key predictors of the traffic accidents severity. There were disagreements among the classification's models about the importance of some factors, in particular, time of day.

Some limitations of this dataset are that it doesn't take driver age, experience, speed prior to accident, vehicle type or vehicle safety features which could provide further insight into the accident determinants and severity. Further research could be conducted to establish correlations to these additional predictors.

Other problems encountered in regards to the dataset are the right skewed distribution of values for traffic accident duration and distance of road affected. Skewness persists even after outlier removal and exclusion of zero distance values, which may impact the results of statistical analysis.

Poorly defined variables in the dataset also remains a problem for interpreting statistical results. For example, how a traffic incident is classified by severity is not clearly defined in both Moosavi's work (2021) from which the dataset originated, nor the direct sources of traffic data from Microsoft Bing Maps Traffic API or MapQuest. Another example is distance, which was calculated using Haversine formula on GPS coordinates corresponding to the traffic accident start and end point. Due to the measurement units stated in miles, there is low resolution for distances that are low in value or near zero miles. A quick internet search indicates the average accuracy of GPS is within 4 meters and the average length of a car is also a similar length; as such, distances recorded as zero may, in fact, refer

to a single vehicle accident covering that vehicle's length of road space. Otherwise, zero distance accidents may also refer to off-road accidents, as no extent of road will be affected by such an event. It is not possible to make a distinction at this stage until further data about such incidents are collected.

Cost-benefit analysis regarding installation or upgrading of highway barriers should be undertaken, as well as NY City considering a congestion tariff. Similarly, implementing measures to encourage car pooling, improved public transport and other transport methods can reduce traffic congestion.

Appendix

Appendix A

Logistic regression model

Logistic regression

In [55]:
```
# Assign the data
df = dfpreds

# Set the target for the prediction
target = 'Severity'

# Create arrays for the features and the response variable

# set X and y
y = df[target]
X = df.drop(target, axis=1)

# Split the data set into training and testing data sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=5, stratify=y)
```

In [56]:
```
# List of classification algorithms
algo_lst=['Logistic Regression',' K-Nearest Neighbors','Decision tree','Random Forest', 'Random Forest Limited']

# Initialize an empty list for the accuracy for each algorithm
accuracy_lst=[]
```

In [57]:
```
# Logistic regression
lr = LogisticRegression(solver='lbfgs', random_state=0, max_iter=10000, multi_class='auto')
lr.fit(X_train,y_train)
y_pred=lr.predict(X_test)

# Get the accuracy score of y pred against y
acc=accuracy_score(y_test, y_pred)

# Append to the accuracy list
accuracy_lst.append(acc)

print("Logistic regression accuracy is: {:.3f}.".format(acc))
```

```
Logistic regression accuracy is: 0.838.
```

In [58]:
```
print('intercept', lr.intercept_[0])
print('classes', lr.classes_)
pd.DataFrame({'Coefficients': lr.coef_[0]},
             index=X.columns)

lr.predict_proba(X)
```

```
intercept -0.0004072231463108777
classes [1 2 3 4]
```

Out[58]:
```
array([[6.54925749e-05, 9.72589276e-01, 4.56784747e-03, 2.27773838e-02],
       [1.74523646e-03, 7.66336222e-01, 1.67040097e-01, 6.48784444e-02],
       [1.18192619e-04, 9.59859221e-01, 8.04166305e-03, 3.19809236e-02],
       ...,
       [2.61127371e-03, 8.00229708e-01, 1.57709620e-01, 3.94493976e-02],
       [1.62368373e-02, 6.24151421e-01, 2.21890380e-01, 1.37721362e-01],
       [5.39699013e-03, 7.08542502e-01, 1.81354148e-01, 1.04706361e-01]])
```

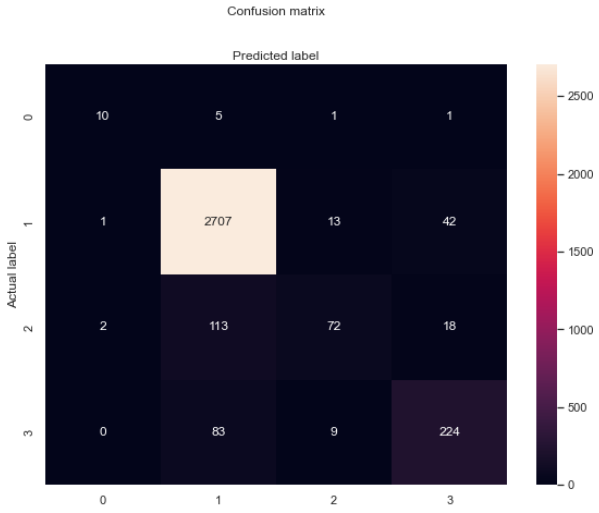
In [95]:
```
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

Out[95]:
```
array([[  10,    5,    1,    1],
       [   1, 2707,   13,   42],
       [   2,  113,   72,   18],
       [   0,   83,    9,  224]], dtype=int64)
```

In [96]:
```
class_names=[1,2,3,4] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')

```

Out[96]:
```
Text(0.5, 384.16, 'Predicted label')
```



In [93]:
```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print("Precision:",metrics.precision_score(y_test, y_pred, average='weighted'))
print("Recall:",metrics.recall_score(y_test, y_pred, average='weighted'))
```

```
Accuracy: 0.9127537109966677
Precision: 0.9054324775953668
Recall: 0.9127537109966677
```

Appendix B

K-Nearest Neighbors model

KNN classification

In [62]:
```
# Create a k-NN classifier with a randomly selected NN=
knn = KNeighborsClassifier(n_neighbors=8)

# Fit the classifier to the data
knn.fit(X_train,y_train)

# Predict the labels for the training data X
y_pred = knn.predict(X_test)

# Get the accuracy score
acc=accuracy_score(y_test, y_pred)

# Append to the accuracy list
accuracy_lst.append(acc)

print('KNN score is: {:.3f}.'.format(knn.score(X_test, y_test)))
print('KNN accuracy is: {:.3f}.'.format(acc))
```

```
KNN score is: 0.852.
KNN accuracy is: 0.852.
```

KNN with CV

Selected a random value for k

In [63]:
```
#create a new KNN model
knn_cv = KNeighborsClassifier(n_neighbors=8)
#train model with cv of 5
cv_scores = cross_val_score(knn, X, y, cv=10)
#print each cv score (accuracy) and average them
print(cv_scores)
print('cv_scores mean:{}'.format(np.mean(cv_scores)))
```

```
[0.8413083  0.85584494 0.85645064 0.84191399 0.8407026  0.84545455
 0.84909091 0.84121212 0.37575758 0.32727273]
cv_scores mean:0.7475008351228825
```

KNN with grid search CV

Using sklearn.model_selection GridSearchCV for cross-validation to select k

In [64]:
```
#create new a knn model
knn2 = KNeighborsClassifier()
#create a dictionary of all values we want to test for n_neighbors
param_grid = {'n_neighbors': np.arange(1, 20)}
#use gridsearch to test all values for n_neighbors
knn_gscv = GridSearchCV(knn2, param_grid, cv=5, scoring='accuracy')
#fit model to data
knn_gscv.fit(X, y)
print('KNN best score is: {}.'.format(knn_gscv.best_score_))
print('KNN optimal k value is: {}.'.format(knn_gscv.best_params_))
```

```
KNN best score is: 0.7065737655255983.
KNN optimal k value is: {'n_neighbors': 12}.
```

```
In [65]: 1 knn_gscv.cv_results_  # get KNN cv results
```

```
Out[65]: {'mean_fit_time': array([0.00359936, 0.00339999, 0.0039989 , 0.0035974 , 0.00359902,
        0.00369892, 0.00359659, 0.00349832, 0.00349898, 0.00339904,
        0.00350041, 0.00369906, 0.00349965, 0.00379715, 0.00359893,
        0.00359879, 0.00309935, 0.00350003, 0.00369864]),
 'std_fit_time': array([3.74370784e-04, 1.99987266e-04, 7.74819044e-04, 2.00206916e-04,
        1.98560547e-04, 2.45010124e-04, 1.98223415e-04, 1.53034273e-06,
        1.04904175e-06, 3.73503097e-04, 1.12234137e-06, 3.98397460e-04,
        5.35248383e-07, 2.43024098e-04, 2.00272855e-04, 3.73630580e-04,
        1.99724176e-04, 2.78041453e-07, 2.43296944e-04]),
 'mean_score_time': array([0.5842998 , 0.65519948, 0.69830022, 0.72440014, 0.73509927,
        0.71980042, 0.72929969, 0.7261003 , 0.73230023, 0.71219993,
        0.73319969, 0.73740001, 0.72579994, 0.72179999, 0.72929974,
        0.76510043, 0.74690018, 0.7354001 , 0.7322001 ]),
 'std_score_time': array([0.0151152 , 0.05048128, 0.03894165, 0.031195  , 0.02227743,
        0.02310138, 0.02085102, 0.02563467, 0.03067029, 0.02706435,
        0.02257501, 0.02463219, 0.02649037, 0.03016385, 0.02011384,
        0.0244923 , 0.03862315, 0.02957422, 0.01515426]),
 'param_n_neighbors': masked_array(data=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
                    17, 18, 19],
              mask=[False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False],
        fill_value='?',
             dtype=object),
 'params': [{'n_neighbors': 1},
  {'n_neighbors': 2},
  {'n_neighbors': 3},
  {'n_neighbors': 4},
  {'n_neighbors': 5},
  {'n_neighbors': 6},
  {'n_neighbors': 7},
  {'n_neighbors': 8},
  {'n_neighbors': 9},
  {'n_neighbors': 10},
  {'n_neighbors': 11},
  {'n_neighbors': 12},
  {'n_neighbors': 13},
  {'n_neighbors': 14},
  {'n_neighbors': 15},
  {'n_neighbors': 16},
  {'n_neighbors': 17},
  {'n_neighbors': 18},
  {'n_neighbors': 19}],
 'split0_test_score': array([0.84853075, 0.84943956, 0.84701606, 0.84731899, 0.8458043 ,
        0.84368373, 0.8418661 , 0.84307786, 0.84550136, 0.84489549,
        0.8458043 , 0.84550136, 0.84459255, 0.84277492, 0.84338079,
        0.8418661 , 0.84307786, 0.83913965, 0.84065435]),
 'split1_test_score': array([0.84004847, 0.84307786, 0.84095729, 0.84459255, 0.84398667,
        0.84641018, 0.84822781, 0.84701606, 0.8497425 , 0.84913663,
        0.84792487, 0.85004544, 0.84883369, 0.84701606, 0.84610724,
        0.84641018, 0.84338079, 0.84247198, 0.84338079]),
 'split2_test_score': array([0.83489852, 0.84004847, 0.84095729, 0.84338079, 0.84004847,
        0.8418661 , 0.84035141, 0.83974553, 0.84035141, 0.83883672,
        0.83883672, 0.83883672, 0.84035141, 0.84156316, 0.84126022,
        0.83974553, 0.84065435, 0.83913965, 0.84004847]),
 'split3_test_score': array([0.84307786, 0.83853378, 0.84004847, 0.83853378, 0.8418661 ,
        0.84216904, 0.84247198, 0.84095729, 0.84216904, 0.84247198,
        0.84126022, 0.8418661 , 0.83944259, 0.83883672, 0.83944259,
        0.83883672, 0.84095729, 0.83823084, 0.83823084]),
 'split4_test_score': array([0.12662829, 0.13571645, 0.14692517, 0.14965162, 0.14541048,
        0.14783399, 0.1463193 , 0.14995456, 0.14783399, 0.15056044,
        0.15237807, 0.15661921, 0.15601333, 0.15995153, 0.16267798,
        0.16237504, 0.16388973, 0.16146622, 0.16510148]),
 'mean_test_score': array([0.69863678, 0.70136322, 0.70318085, 0.70469555, 0.70342321,
        0.70439261, 0.70384732, 0.70415026, 0.70511966, 0.70518025,
        0.70524084, 0.70657377, 0.70584671, 0.70602848, 0.70657377,
        0.70584671, 0.706392  , 0.70408967, 0.70548319]),
 'std_test_score': array([0.28603834, 0.28284811, 0.27813895, 0.27753654, 0.2790131 ,
        0.27828396, 0.27877682, 0.27710887, 0.27866118, 0.27733013,
        0.27645007, 0.27500273, 0.27493701, 0.2730512 , 0.27195696,
        0.27174843, 0.27125334, 0.27131558, 0.2701959 ]),
 'rank_test_score': array([19, 18, 17, 11, 16, 12, 15, 13, 10,  9,  8,  1,  5,  4,  1,  5,  3,
        14,  7])}
```

Appendix C

Decision Tree model

Decision tree

Decision tree with entropy info criterion

In [66]:
```
# Decision tree algorithm

# Instantiate dt_entropy, set 'entropy' as the information criterion
dt_entropy = DecisionTreeClassifier(max_depth=8, criterion='entropy', random_state=1)


# Fit dt_entropy to the training set
dt_entropy.fit(X_train, y_train)

# Use dt_entropy to predict test set labels
y_pred= dt_entropy.predict(X_test)

# Evaluate accuracy_entropy
accuracy_entropy = accuracy_score(y_test, y_pred)


# Print accuracy_entropy
print('[Decision Tree -- entropy] accuracy_score: {:.3f}.'.format(accuracy_entropy))



# Instantiate dt_gini, set 'gini' as the information criterion
dt_gini = DecisionTreeClassifier(max_depth=8, criterion='gini', random_state=1)


# Fit dt_entropy to the training set
dt_gini.fit(X_train, y_train)

# Use dt_entropy to predict test set labels
y_pred= dt_gini.predict(X_test)

# Evaluate accuracy_entropy
accuracy_gini = accuracy_score(y_test, y_pred)

# Append to the accuracy list
acc=accuracy_gini
accuracy_lst.append(acc)

# Print accuracy_gini
print('Decision Tree accuracy with gini index: {:.3f}.'.format(accuracy_gini))
```

```
[Decision Tree -- entropy] accuracy_score: 0.869.
Decision Tree accuracy with gini index: 0.869.
```

In [68]:
```
# decision tree visualisation plot
dfpreds.feature_names = list(dfpreds)
fig = plt.figure(figsize=(25,20))
_ = tree.plot_tree(dt_entropy,
                   feature_names=dfpreds.feature_names,
                   class_names=str(dfpreds['Severity']),
                   filled=True,fontsize=12)
```

```
|--- feature_17 <= 64.50|   |--- feature_14 <= 2019.50|   |   |--- feature_16 <= 4.50|   |   |   |---
feature_15 <= 8.50|   |   |   |   |--- feature_22 <= 29.97|   |   |   |   |   |--- feature_15 <= 4.50|
|   |   |   |   |   |--- feature_19 <= 0.19|   |   |   |   |   |   |--- feature_17 <= 28.50|   |
|   |   |   |   |   |   |--- class: 4|   |   |   |   |   |   |--- feature_17 >  28.50|   |   |   |
|   |   |   |   |--- class: 3|   |   |   |   |   |--- feature_19 >  0.19|   |   |   |   |   |   |
|--- feature_16 <= 0.50|   |   |   |   |   |   |   |   |--- class: 3|   |   |   |   |   |   |   |---
feature_16 >  0.50|   |   |   |   |   |   |   |   |--- class: 3|   |   |   |   |--- feature_15 >
4.50|   |   |   |   |   |--- feature_22 <= 29.54|   |   |   |   |   |   |--- feature_23 <=
4.00|   |   |   |   |   |   |   |--- class: 4|   |   |   |   |   |   |--- feature_23 >  4.00|
|   |   |   |   |   |   |   |--- class: 4|   |   |   |   |   |--- feature_22 >  29.54|   |   |   |
|   |   |   |--- feature_19 <= 0.60|   |   |   |   |   |   |   |--- class: 4|   |   |   |   |   |
|   |--- feature_19 >  0.60|   |   |   |   |   |   |   |--- class: 3|   |   |   |--- 
feature_22 >  29.97|   |   |   |   |--- feature_19 <= 1.01|   |   |   |   |   |--- feature_17
<= 27.50|   |   |   |   |   |   |--- feature_19 <= 0.78|   |   |   |   |   |   |   |--- class:
2|   |   |   |   |   |   |--- feature_19 >  0.78|   |   |   |   |   |   |   |--- class: 3|   |
|   |   |   |--- feature_17 >  27.50|   |   |   |   |   |   |--- feature_16 <= 3.50|   |   |
|   |   |   |   |--- class: 3|   |   |   |   |   |   |--- feature_16 >  3.50|   |   |   |   |
|   |   |--- class: 3|   |   |   |   |--- feature_19 >  1.01|   |   |   |   |   |--- 
class: 2|   |   |--- feature_15 >  8.50|   |   |   |--- feature_16 <= 0.50|   |   |   |   |
|--- feature_21 <= 66.50|   |   |   |   |   |--- feature_20 <= 45.00|   |   |   |   |   |   |-
-- feature_21 <= 56.00|   |   |   |   |   |   |   |--- class: 4|   |   |   |   |   |   |--- 
feature_21 >  56.00|   |   |   |   |   |   |   |--- class: 2|   |   |   |   |   |--- 
feature_20 >  45.00|   |   |   |   |   |   |--- feature_0 <= 0.50|   |   |   |   |   |   |   |
|--- class: 3|   |   |   |   |   |   |--- feature_0 >  0.50|   |   |   |   |   |   |   |---
```

```
class: 4|   |   |   |   |--- feature_21 >  66.50|   |   |   |   |   |--- feature_23 <= 13.50|
|   |   |   |   |   |   |--- feature_23 <= 1.50|   |   |   |   |   |   |   |--- class: 4|   |   |
|   |   |   |   |--- feature_23 >  1.50|   |   |   |   |   |   |   |--- class: 4|   |   |   |   |
|   |--- feature_23 >  13.50|   |   |   |   |   |   |--- class: 3|   |   |   |--- feature_16 >
0.50|   |   |   |   |   |--- class: 4|   |   |--- feature_16 >  4.50|   |   |   |--- feature_4 <=
0.50|   |   |   |   |--- feature_16 <= 20.50|   |   |   |   |--- feature_18 <= 0.50|   |   |   |
|   |   |--- feature_22 <= 29.71|   |   |   |   |   |   |   |--- feature_19 <= 0.61|   |   |   |   |
|   |   |   |   |--- class: 2|   |   |   |   |   |   |   |--- feature_19 >  0.61|   |   |   |   |   |   |
|   |--- class: 2|   |   |   |   |   |   |--- feature_22 >  29.71|   |   |   |   |   |   |   |---
feature_20 <= 43.50|   |   |   |   |   |   |   |   |--- class: 2|   |   |   |   |   |   |   |---
feature_20 >  43.50|   |   |   |   |   |   |   |   |--- class: 2|   |   |   |   |   |--- feature_18 >
0.50|   |   |   |   |   |   |--- feature_22 <= 29.74|   |   |   |   |   |   |   |--- feature_15 <=
7.50|   |   |   |   |   |   |   |   |--- class: 2|   |   |   |   |   |   |   |--- feature_15 >  7.50|
|   |   |   |   |   |   |   |   |--- class: 4|   |   |   |   |   |   |--- feature_22 >  29.74|   |   |   |
|   |   |   |--- feature_20 <= 82.50|   |   |   |   |   |   |   |   |--- class: 2|   |   |   |   |   |
|   |--- feature_20 >  82.50|   |   |   |   |   |   |   |   |--- class: 2|   |   |   |   |---
feature_16 >  20.50|   |   |   |   |   |--- feature_21 <= 56.50|   |   |   |   |   |   |--- feature_20
<= 54.00|   |   |   |   |   |   |   |--- feature_19 <= 0.06|   |   |   |   |   |   |   |   |--- class:
2|   |   |   |   |   |   |   |--- feature_19 >  0.06|   |   |   |   |   |   |   |   |--- class: 3|   |
|   |   |   |   |--- feature_20 >  54.00|   |   |   |   |   |   |   |--- feature_22 <= 30.02|   |   |
|   |   |   |   |   |--- class: 2|   |   |   |   |   |   |   |--- feature_22 >  30.02|   |   |   |   |
|   |   |   |--- class: 2|   |   |   |   |   |--- feature_21 >  56.50|   |   |   |   |   |   |---
feature_19 <= 0.67|   |   |   |   |   |   |   |--- feature_15 <= 10.50|   |   |   |   |   |   |   |
|--- class: 4|   |   |   |   |   |   |   |--- feature_15 >  10.50|   |   |   |   |   |   |   |   |---
class: 4|   |   |   |   |   |   |--- feature_19 >  0.67|   |   |   |   |   |   |   |--- feature_20 <=
64.50|   |   |   |   |   |   |   |   |--- class: 4|   |   |   |   |   |   |   |--- feature_20 >
64.50|   |   |   |   |   |   |   |   |--- class: 4|   |   |   |--- feature_4 >  0.50|   |   |   |   |-
-- feature_15 <= 9.50|   |   |   |   |   |--- feature_19 <= 0.66|   |   |   |   |   |   |---
feature_16 <= 6.50|   |   |   |   |   |   |   |--- feature_21 <= 81.00|   |   |   |   |   |   |   |
|--- class: 3|   |   |   |   |   |   |   |--- feature_21 >  81.00|   |   |   |   |   |   |   |   |---
class: 2|   |   |   |   |   |   |--- feature_16 >  6.50|   |   |   |   |   |   |   |--- feature_19 <=
0.29|   |   |   |   |   |   |   |   |--- class: 2|   |   |   |   |   |   |   |--- feature_19 >  0.29|
|   |   |   |   |   |   |   |   |--- class: 2|   |   |   |   |   |--- feature_19 >  0.66|   |   |   |   |
|   |--- feature_19 <= 1.51|   |   |   |   |   |   |   |--- feature_19 <= 1.45|   |   |   |   |   |
|   |   |--- class: 2|   |   |   |   |   |   |   |--- feature_19 >  1.45|   |   |   |   |   |   |   |
|--- class: 4|   |   |   |   |   |--- feature_19 >  1.51|   |   |   |   |   |   |--- class: 2|
|   |   |   |--- feature_15 >  9.50|   |   |   |   |--- feature_22 <= 29.84|   |   |   |   |   |
|--- feature_20 <= 32.50|   |   |   |   |   |   |--- feature_17 <= 27.50|   |   |   |   |   |   |
|   |--- class: 2|   |   |   |   |   |   |--- feature_17 >  27.50|   |   |   |   |   |   |   |
|--- class: 4|   |   |   |   |   |   |--- feature_20 >  32.50|   |   |   |   |   |   |   |---
feature_19 <= 0.07|   |   |   |   |   |   |   |   |--- class: 4|   |   |   |   |   |   |   |---
feature_19 >  0.07|   |   |   |   |   |   |   |   |--- class: 2|   |   |   |   |   |--- feature_22 >
29.84|   |   |   |   |   |   |--- feature_19 <= 0.92|   |   |   |   |   |   |   |--- feature_21 <=
63.50|   |   |   |   |   |   |   |   |--- class: 2|   |   |   |   |   |   |   |--- feature_21 >
63.50|   |   |   |   |   |   |   |   |--- class: 2|   |   |   |   |   |   |--- feature_19 >  0.92|   |
|   |   |   |   |   |--- feature_22 <= 30.30|   |   |   |   |   |   |   |   |--- class: 4|   |   |   |
|   |   |   |--- feature_22 >  30.30|   |   |   |   |   |   |   |   |--- class: 3|   |--- feature_14 >
2019.50|   |   |--- feature_15 <= 8.50|   |   |   |--- feature_15 <= 2.50|   |   |   |   |---
feature_16 <= 6.50|   |   |   |   |   |--- feature_21 <= 69.50|   |   |   |   |   |   |--- feature_23
<= 8.50|   |   |   |   |   |   |   |--- feature_19 <= 0.39|   |   |   |   |   |   |   |   |--- class:
2|   |   |   |   |   |   |   |--- feature_19 >  0.39|   |   |   |   |   |   |   |   |--- class: 3|   |
|   |   |   |   |--- feature_23 >  8.50|   |   |   |   |   |   |   |--- feature_19 <= 0.35|   |   |
|   |   |   |   |   |--- class: 4|   |   |   |   |   |   |   |--- feature_19 >  0.35|   |   |   |   |
|   |   |   |--- class: 2|   |   |   |   |--- feature_21 >  69.50|   |   |   |   |   |   |---
feature_22 <= 29.86|   |   |   |   |   |   |   |--- class: 4|   |   |   |   |   |   |--- feature_22 >
29.86|   |   |   |   |   |   |   |--- feature_11 <= 0.50|   |   |   |   |   |   |   |   |--- class: 3|
|   |   |   |   |   |   |   |--- feature_11 >  0.50|   |   |   |   |   |   |   |   |--- class: 4|   |   |
|   |--- feature_16 >  6.50|   |   |   |   |--- feature_4 <= 0.50|   |   |   |   |   |   |---
feature_22 <= 29.77|   |   |   |   |   |   |   |--- feature_20 <= 32.00|   |   |   |   |   |   |   |
|--- class: 4|   |   |   |   |   |   |   |--- feature_20 >  32.00|   |   |   |   |   |   |   |   |---
class: 2|   |   |   |   |   |   |--- feature_22 >  29.77|   |   |   |   |   |   |   |--- feature_9 <=
0.50|   |   |   |   |   |   |   |   |--- class: 2|   |   |   |   |   |   |   |--- feature_9 >  0.50|
|   |   |   |   |   |   |   |   |--- class: 3|   |   |   |   |   |--- feature_4 >  0.50|   |   |   |   |
|   |--- feature_21 <= 54.50|   |   |   |   |   |   |--- feature_19 <= 0.11|   |   |   |   |   |
```

```
|   |   |--- class: 3|   |   |   |   |   |   |--- feature_19 >  0.11|   |   |   |   |   |   |   |
|--- class: 2|   |   |   |   |   |--- feature_21 >  54.50|   |   |   |   |   |   |---
feature_16 <= 15.50|   |   |   |   |   |   |   |--- class: 2|   |   |   |   |   |   |---
feature_16 >  15.50|   |   |   |   |   |   |   |--- class: 2|   |   |--- feature_15 >  2.50|
|   |   |   |--- feature_17 <= 50.00|   |   |   |   |--- feature_20 <= 68.50|   |   |   |   |   |
|--- feature_19 <= 0.03|   |   |   |   |   |   |--- feature_21 <= 32.00|   |   |   |   |   |   |
|   |--- class: 1|   |   |   |   |   |   |--- feature_21 >  32.00|   |   |   |   |   |   |   |
|--- class: 2|   |   |   |   |   |--- feature_19 >  0.03|   |   |   |   |   |   |---
feature_22 <= 30.06|   |   |   |   |   |   |   |--- class: 4|   |   |   |   |   |   |---
feature_22 >  30.06|   |   |   |   |   |   |   |--- class: 4|   |   |   |   |--- feature_20 >
68.50|   |   |   |   |   |--- feature_19 <= 1.82|   |   |   |   |   |   |--- feature_17 <=
21.50|   |   |   |   |   |   |   |--- class: 3|   |   |   |   |   |   |--- feature_17 >
21.50|   |   |   |   |   |   |   |--- class: 4|   |   |   |   |   |--- feature_19 >  1.82|   |
|   |   |   |   |--- feature_21 <= 44.00|   |   |   |   |   |   |   |--- class: 1|   |   |   |
|   |   |   |--- feature_21 >  44.00|   |   |   |   |   |   |   |--- class: 3|   |   |   |---
feature_17 >  50.00|   |   |   |   |--- feature_13 <= 0.50|   |   |   |   |   |--- feature_19
<= 0.07|   |   |   |   |   |   |--- feature_19 <= 0.01|   |   |   |   |   |   |   |--- class:
2|   |   |   |   |   |   |--- feature_19 >  0.01|   |   |   |   |   |   |   |--- class: 3|   |
|   |   |   |   |--- feature_19 >  0.07|   |   |   |   |   |   |--- feature_21 <= 70.50|   |   |
|   |   |   |   |   |--- class: 4|   |   |   |   |   |   |--- feature_21 >  70.50|   |   |   |   |
|   |   |   |--- class: 4|   |   |   |   |--- feature_13 >  0.50|   |   |   |   |   |---
feature_19 <= 0.72|   |   |   |   |   |   |--- feature_22 <= 29.40|   |   |   |   |   |   |   |
|--- class: 2|   |   |   |   |   |   |   |--- feature_22 >  29.40|   |   |   |   |   |   |   |---
class: 2|   |   |   |   |   |   |--- feature_19 >  0.72|   |   |   |   |   |   |--- feature_16 <=
9.50|   |   |   |   |   |   |   |   |--- class: 2|   |   |   |   |   |   |--- feature_16 >  9.50|
|   |   |   |   |   |   |   |--- class: 1|   |   |--- feature_15 >  8.50|   |   |   |--- feature_17 <=
63.50|   |   |   |   |--- class: 2|   |   |   |--- feature_17 >  63.50|   |   |   |   |--- feature_11
<= 0.50|   |   |   |   |   |--- class: 2|   |   |   |   |--- feature_11 >  0.50|   |   |   |   |--
- feature_16 <= 19.00|   |   |   |   |   |   |--- class: 2|   |   |   |   |   |--- feature_16 >
19.00|   |   |   |   |   |   |--- class: 2|--- feature_17 >  64.50|   |--- feature_15 <= 8.50|   |
|--- feature_17 <= 117.00|   |   |   |--- feature_22 <= 29.26|   |   |   |   |--- feature_15 <= 3.50|
|   |   |   |   |--- feature_18 <= 0.50|   |   |   |   |   |   |--- feature_20 <= 48.00|   |   |   |
|   |   |   |--- class: 4|   |   |   |   |   |   |--- feature_20 >  48.00|   |   |   |   |   |   |
|--- class: 3|   |   |   |   |   |--- feature_18 >  0.50|   |   |   |   |   |   |--- class: 3|   |   |
|   |--- feature_15 >  3.50|   |   |   |   |   |--- feature_19 <= 0.47|   |   |   |   |   |   |---
feature_21 <= 78.50|   |   |   |   |   |   |   |--- class: 2|   |   |   |   |   |   |--- feature_21 >
78.50|   |   |   |   |   |   |   |--- class: 3|   |   |   |   |   |--- feature_19 >  0.47|   |   |   |
|   |   |--- feature_18 <= 0.50|   |   |   |   |   |   |   |--- feature_21 <= 65.00|   |   |   |   |
|   |   |   |--- class: 3|   |   |   |   |   |   |   |--- feature_21 >  65.00|   |   |   |   |   |   |
|   |--- class: 2|   |   |   |   |   |   |--- feature_18 >  0.50|   |   |   |   |   |   |   |---
class: 3|   |   |   |--- feature_22 >  29.26|   |   |   |   |--- feature_13 <= 0.50|   |   |   |   |
|--- feature_19 <= 0.02|   |   |   |   |   |   |--- class: 2|   |   |   |   |   |--- feature_19 >
0.02|   |   |   |   |   |   |--- feature_22 <= 29.44|   |   |   |   |   |   |   |--- class: 3|   |   |
|   |   |   |--- feature_22 >  29.44|   |   |   |   |   |   |   |--- feature_20 <= 78.00|   |   |   |
|   |   |   |   |--- class: 4|   |   |   |   |   |   |   |--- feature_20 >  78.00|   |   |   |   |   |
|   |   |--- class: 3|   |   |   |   |--- feature_13 >  0.50|   |   |   |   |   |--- feature_21 <=
35.50|   |   |   |   |   |   |--- feature_19 <= 0.45|   |   |   |   |   |   |   |--- feature_19 <=
0.11|   |   |   |   |   |   |   |   |--- class: 2|   |   |   |   |   |   |   |--- feature_19 >  0.11|
|   |   |   |   |   |   |   |   |--- class: 2|   |   |   |   |   |   |--- feature_19 >  0.45|   |   |   |
|   |   |   |--- feature_23 <= 9.50|   |   |   |   |   |   |   |   |--- class: 3|   |   |   |   |   |
|   |--- feature_23 >  9.50|   |   |   |   |   |   |   |   |--- class: 2|   |   |   |   |   |---
feature_21 >  35.50|   |   |   |   |   |   |--- feature_19 <= 0.07|   |   |   |   |   |   |   |---
feature_20 <= 82.50|   |   |   |   |   |   |   |   |--- class: 2|   |   |   |   |   |   |   |---
feature_20 >  82.50|   |   |   |   |   |   |   |   |--- class: 4|   |   |   |   |   |   |---
feature_19 >  0.07|   |   |   |   |   |   |   |--- feature_19 <= 0.45|   |   |   |   |   |   |   |
|--- class: 2|   |   |   |   |   |   |   |--- feature_19 >  0.45|   |   |   |   |   |   |   |   |---
class: 2|   |   |--- feature_17 >  117.00|   |   |   |--- feature_19 <= 0.23|   |   |   |   |   |---
feature_13 <= 0.50|   |   |   |   |   |--- feature_19 <= 0.13|   |   |   |   |   |   |--- class: 4|
|   |   |   |   |--- feature_19 >  0.13|   |   |   |   |   |   |--- feature_23 <= 9.50|   |   |   |
|   |   |   |--- feature_15 <= 4.50|   |   |   |   |   |   |   |   |--- class: 4|   |   |   |   |   |
|   |--- feature_15 >  4.50|   |   |   |   |   |   |   |   |--- class: 3|   |   |   |   |   |   |---
feature_23 >  9.50|   |   |   |   |   |   |   |--- class: 4|   |   |   |   |--- feature_13 >  0.50|
|   |   |   |   |--- feature_20 <= 46.50|   |   |   |   |   |   |--- feature_23 <= 18.50|   |   |   |
|   |   |   |--- class: 2|   |   |   |   |   |   |--- feature_23 >  18.50|   |   |   |   |   |   |
```

```
|--- class: 4|   |   |   |   |--- feature_20 >  46.50|   |   |   |   |   |--- feature_20 <=
82.50|   |   |   |   |   |   |--- feature_21 <= 57.50|   |   |   |   |   |   |   |--- class:
4|   |   |   |   |   |   |--- feature_21 >  57.50|   |   |   |   |   |   |   |--- class: 3|
|   |   |   |   |   |--- feature_20 >  82.50|   |   |   |   |   |   |--- class: 2|   |   |   |---
feature_19 >  0.23|   |   |   |   |--- feature_17 <= 119.50|   |   |   |   |--- class: 3|   |   |
|   |--- feature_17 >  119.50|   |   |   |   |   |--- class: 4|   |--- feature_15 >  8.50|   |   |---
feature_17 <= 249.50|   |   |   |--- feature_23 <= 16.50|   |   |   |   |--- feature_21 <= 66.50|   |
|   |   |   |   |--- feature_3 <= 0.50|   |   |   |   |   |   |--- feature_19 <= 0.01|   |   |   |   |   |
|   |--- feature_19 <= 0.01|   |   |   |   |   |   |   |   |--- class: 2|   |   |   |   |   |   |   |-
-- feature_19 >  0.01|   |   |   |   |   |   |   |   |--- class: 2|   |   |   |   |   |   |---
feature_19 >  0.01|   |   |   |   |   |   |   |--- feature_21 <= 48.50|   |   |   |   |   |   |   |
|--- class: 2|   |   |   |   |   |   |   |--- feature_21 >  48.50|   |   |   |   |   |   |   |   |---
class: 2|   |   |   |   |   |--- feature_3 >  0.50|   |   |   |   |   |--- feature_17 <= 116.50|
|   |   |   |   |   |   |--- class: 2|   |   |   |   |   |   |--- feature_17 >  116.50|   |   |   |
|   |   |   |--- class: 4|   |   |   |   |--- feature_21 >  66.50|   |   |   |   |   |--- feature_15
<= 11.50|   |   |   |   |   |   |--- feature_20 <= 59.50|   |   |   |   |   |   |   |--- feature_21 <=
70.50|   |   |   |   |   |   |   |   |--- class: 2|   |   |   |   |   |   |   |--- feature_21 >
70.50|   |   |   |   |   |   |   |   |--- class: 2|   |   |   |   |   |   |--- feature_20 >  59.50|
|   |   |   |   |   |   |   |--- feature_20 <= 74.50|   |   |   |   |   |   |   |   |--- class: 2|   |   |
|   |   |   |   |   |--- feature_20 >  74.50|   |   |   |   |   |   |   |   |--- class: 2|   |   |   |   |
|--- feature_15 >  11.50|   |   |   |   |   |   |--- feature_17 <= 119.50|   |   |   |   |   |   |
|--- feature_16 <= 16.50|   |   |   |   |   |   |   |   |--- class: 2|   |   |   |   |   |   |   |---
feature_16 >  16.50|   |   |   |   |   |   |   |   |--- class: 2|   |   |   |   |   |   |---
feature_17 >  119.50|   |   |   |   |   |   |   |--- class: 2|   |   |   |--- feature_23 >  16.50|   |
|   |   |--- feature_21 <= 51.50|   |   |   |   |   |--- feature_22 <= 29.50|   |   |   |   |   |   |-
-- feature_22 <= 29.46|   |   |   |   |   |   |   |--- feature_16 <= 0.50|   |   |   |   |   |   |   |
|--- class: 2|   |   |   |   |   |   |   |--- feature_16 >  0.50|   |   |   |   |   |   |   |   |---
class: 2|   |   |   |   |   |   |--- feature_22 >  29.46|   |   |   |   |   |   |   |--- feature_17 <=
90.00|   |   |   |   |   |   |   |   |--- class: 2|   |   |   |   |   |   |   |--- feature_17 >
90.00|   |   |   |   |   |   |   |   |--- class: 4|   |   |   |   |   |--- feature_22 >  29.50|   |
|   |   |   |   |   |--- feature_22 <= 30.11|   |   |   |   |   |   |   |--- class: 2|   |   |   |   |   |
|--- feature_22 >  30.11|   |   |   |   |   |   |   |--- feature_19 <= 0.58|   |   |   |   |   |   |   |
|   |--- class: 2|   |   |   |   |   |   |   |--- feature_19 >  0.58|   |   |   |   |   |   |   |   |-
-- class: 2|   |   |   |   |--- feature_21 >  51.50|   |   |   |   |--- feature_17 <= 206.50|   |
|   |   |   |   |--- class: 2|   |   |   |   |   |--- feature_17 >  206.50|   |   |   |   |   |   |---
feature_17 <= 213.50|   |   |   |   |   |   |   |--- class: 2|   |   |   |   |   |   |--- feature_17 >
213.50|   |   |   |   |   |   |   |--- class: 2|   |   |--- feature_17 >  249.50|   |   |   |---
feature_14 <= 2019.50|   |   |   |   |--- feature_17 <= 306.50|   |   |   |   |   |--- class: 4|   |
|   |   |--- feature_17 >  306.50|   |   |   |   |   |--- class: 2|   |   |   |--- feature_14 >
2019.50|   |   |   |   |--- feature_16 <= 10.50|   |   |   |   |   |--- feature_22 <= 29.39|   |   |
|   |   |   |--- class: 2|   |   |   |   |   |--- feature_22 >  29.39|   |   |   |   |   |   |---
feature_20 <= 35.50|   |   |   |   |   |   |   |--- feature_19 <= 0.06|   |   |   |   |   |   |   |
|--- class: 2|   |   |   |   |   |   |   |--- feature_19 >  0.06|   |   |   |   |   |   |   |   |---
class: 2|   |   |   |   |   |   |--- feature_20 >  35.50|   |   |   |   |   |   |   |--- feature_15 <=
10.50|   |   |   |   |   |   |   |   |--- class: 2|   |   |   |   |   |   |   |--- feature_15 >
10.50|   |   |   |   |   |   |   |   |--- class: 2|   |   |   |   |--- feature_16 >  10.50|   |   |
|   |   |--- feature_19 <= 0.68|   |   |   |   |   |--- feature_19 <= 0.65|   |   |   |   |   |
|   |--- feature_16 <= 15.50|   |   |   |   |   |   |   |   |--- class: 2|   |   |   |   |   |   |
|--- feature_16 >  15.50|   |   |   |   |   |   |   |   |--- class: 2|   |   |   |   |   |   |---
feature_19 >  0.65|   |   |   |   |   |   |   |--- feature_17 <= 287.50|   |   |   |   |   |   |   |
|--- class: 4|   |   |   |   |   |   |   |--- feature_17 >  287.50|   |   |   |   |   |   |   |   |---
class: 2|   |   |   |   |--- feature_19 >  0.68|   |   |   |   |   |--- class: 2
```

Appendix D

Random Forest and Limited Random Forest model

Random forest

Random forest with default 100 trees

In [70]:
```
# Random Forest algorithm

#Create a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100)

#Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train,y_train)

y_pred=clf.predict(X_test)


# Get the accuracy score
acc=accuracy_score(y_test, y_pred)

# Append to the accuracy list
accuracy_lst.append(acc)


# Model Accuracy, how often is the classifier correct?
print("Random forest accuracy : {:.3f}.".format(acc))
```

```
Random forest accuracy : 0.910.
```

In [72]:
```
# List top k important features
k=20
feature_imp.sort_values(ascending=False)[:k]
```

Out[72]:
```
Duration           0.165358
Month              0.157154
Distance           0.114482
Air_Pressure       0.101677
Humidity           0.095058
Temp               0.094801
Hour               0.085254
Wind_Speed         0.070318
Year               0.046931
Daytime            0.018143
Junction           0.011153
POI_nearby         0.010498
Traffic_Signal     0.009428
Crossing           0.005924
Amenity            0.003470
Station            0.003291
Stop               0.003206
Railway            0.002429
Give_Way           0.001403
No_Exit            0.000018
dtype: float64
```

In [73]:
```
# Create a selector object that will use the random forest classifier to identify
# features that have an importance of more than 0.03
sfm = SelectFromModel(clf, threshold=0.03)

# Train the selector
sfm.fit(X_train, y_train)

feat_labels=X.columns

# Print the names of the most important features
for feature_list_index in sfm.get_support(indices=True):
    print(feat_labels[feature_list_index])
```

```
Year
Month
Hour
Duration
Distance
Temp
Humidity
Air_Pressure
Wind_Speed
```

In [74]:
```
# Transform the data to create a new dataset containing only the most important features
# Note: We have to apply the transform to both the training X and test X data.
X_important_train = sfm.transform(X_train)
X_important_test = sfm.transform(X_test)

# Create a new random forest classifier for the most important features
clf_important = RandomForestClassifier(n_estimators=100, random_state=0, n_jobs=-1)

# Train the new classifier on the new dataset containing the most important features
clf_important.fit(X_important_train, y_train)
```

Out[74]:
```
RandomForestClassifier(n_jobs=-1, random_state=0)
```

In [75]:
```
# Apply The Full Featured Classifier To The Test Data
y_pred = clf.predict(X_test)

# View The Accuracy Of Our Full Feature Model
print('[Randon forest algorithm -- Full feature] accuracy_score: {:.3f}.'.format(accuracy_score(y_test, y_pred)))

# Apply The Full Featured Classifier To The Test Data
y_important_pred = clf_important.predict(X_important_test)

# View The Accuracy Of Our Limited Feature Model
print('[Randon forest algorithm -- Limited feature] accuracy_score: {:.3f}.'.format(accuracy_score(y_test, y_important_pr

# Get the accuracy score
acc=accuracy_score(y_test, y_important_pred)

# Append to the accuracy list
accuracy_lst.append(acc)
```

```
[Randon forest algorithm -- Full feature] accuracy_score: 0.910.
[Randon forest algorithm -- Limited feature] accuracy_score: 0.919.
```

References

Dorokhin, S, Ivannikov, V, Yakovlev, K, Shvyriov, A, Shemyakin, A, Borychev, S & Terentyev, A (2020), Reducing the severity of a traffic accident, *E3S Web of Conferences, vol. 164*, pp. http://dx.doi.org/10.1051/e3sconf/202016403012

Green, C, Heywood, J & Navarro, M (2015), Traffic accidents and the London congestion charge, *Journal of Public Economics, vol. 133,* 2016, pp. 11-22, https://doi-org.ezproxy.newcastle.edu.au/10.1016/j.jpubeco.2015.10.005

Mohan Rao, A., & Ramachandra Rao, K. (2012). MEASURING URBAN TRAFFIC CONGESTION – A REVIEW. *International Journal For Traffic And Transport Engineering*, *2*(4), 286-305. https://doi.org/10.7708/ijtte.2012.2(4).01

Moosavi, S. (2021). *US-Accidents: A Countrywide Traffic Accident Dataset*. Moosavi.org. Retrieved 12 November 2021, from https://smoosavi.org/datasets/us_accidents.

Murad, M & Abaza, A (2006). Pavement Friction in a Program to Reduce Wet Weather Traffic Accidents at the Network Level, *Journal of the Transportation Research Board, vol. 1949* (1), pp. 126-136, https://doi-org.ezproxy.newcastle.edu.au/10.1177%2F0361198106194900111

Retallack, A., & Ostendorf, B. (2019). Current Understanding of the Effects of Congestion on Traffic Accidents. *International Journal Of Environmental Research And Public Health*, *16*(18), 3400. https://doi.org/10.3390/ijerph16183400

Shaaban, K., & Ibrahim, M. (2021). Analysis and Identification of Contributing Factors of Traffic Crashes in New York City. *Transportation Research Procedia*, *55*, 1696-1703. https://doi.org/10.1016/j.trpro.2021.07.161

Wu, L., Chu, J., Ci, Y., Feng, S., & Liu, X. (2015). Engineering Solutions to Enhance Traffic Safety Performance on Two-Lane Highways. *Mathematical Problems In Engineering*, *2015*, 1-7. https://doi.org/10.1155/2015/762379