FEATURE ENGINEERING

The feature engineering done here was to create a month feature from the case number feature. Features with an enormous amount of missing values like species and features like time which cannot be imputed were dropped. Then the activities were binned and only the activities which had a count of greater than 15 were taken in the final train dataset.Finally one-hot encoding was done to all the categorical variables.

MACHINE LEARNING

The next step was to build a classifier which classifies the fatalities to non-fatallities.

FEATURES USED

1) Attack Type
2) Gender
3) Month
4) Age
5) Country
6) Activity

MODELS USED

1) Logistic Regression
2) Decision Tree
3) Random Forest
4) Ada-Boost
5) XG-Boost
6) Gaussian Naive bayes
7) Multinomial Naive bayes

Since this dataset was imbalanced(i.e it had a 79-21 class percentage) SMOTE had to be applied on it to make it balanced.

MODELS PERFORMANCE

	Logistic Regression	Decision Tree	Random Forest	Ada Boost	XGBOOST
TP	164.000000	198.000000	155.000000	132.000000	134.000000
TN	648.000000	466.000000	665.000000	731.000000	749.000000
FP	81.000000	47.000000	90.000000	113.000000	121.000000
FN	235.000000	417.000000	218.000000	152.000000	134.000000
SENSITIVITY	0.411028	0.321951	0.415550	0.464789	0.500000
SPECIFICITY	0.888889	0.908382	0.880795	0.866114	0.860920
Precision	0.669388	0.808163	0.632653	0.538776	0.525490
f1_score	0.509317	0.460465	0.501618	0.499055	0.512428

Since this was an imbalanced class problem the accuracy metric becomes the f1-score.

From the above table we can say that XG-Boost performs best but it will be very obvious that it performs best due to its mathematical complexity. But for a small dataset like this it would be better to go with Logistic Regression or Random Forest.

HYPERPARAMETER TUNING

Tuning both logistic regression and random forest did not yield better results The best c parameter for Logistic regression was 1.
The best parameters for Random Forest was
{'bootstrap': True,
'max_depth': 30,
'max_features': 3,
'min_samples_leaf': 3,
'min_samples_split': 8,
'n_estimators': 300}

CLASSIFYING USING TEXT FEATURE

There is a feature named injury which describes the type of injuries that person has got. So we will try to classify fatalities with that text column.

CLEANING THE INJURY COLUMN

1) Converting all text to lowercase
2) Removing stopwords
3) Applying Stemming

The next step was to create a Document Term Matrix to that text column.This is a highly sparse matrix. With a sparsity of 99.5%

Both Gaussian Naive bayes and Multinomial Naive bayes were used in this classification.

The multinomial naive bayes performs better than the Gaussian Naive Bayes with a very good sensitivity of 98.300.

CONCLUSION

We can conclude that classifying fatalities with just the single text column gives far more superior results than conventional Machine learning algorithms. This is due to the fact that multinomial naive bayes is essentially built to perform on text data.