Capstone II: Final Project Report

Steven Harpham

08/23/2021

Problem Identification

The problem to solve, is to determine whether, based on conditions that existed before a vehicular accident(driver, road, weather, time of day etc.), if it is possible to predict the resulting severity-level of each crash. Crashes in this dataset had been attributed one of three different severity scores, "Slight", "Severe", or "Fatal". The potential use cases for solving such a problem could be substantial, from car manufacturers, to insurance companies, and governmental emergency response. The dataset included accidents from the years 2004 until 2016, and was obtained from Kaggle, where its original data source had been the UK government.

Data Wrangling

My first steps with this project were to create the project's organizational structure. I started by creating a virtual environment in which to work in. I then used the python package "Cookie Cutter" (https://cookiecutter.readthedocs.io/en/1.7.2/)to create an organizational structure for the project, with folders for Data, Notebooks Models, Reports,*.yaml/spec files so that is someone wanted to reproduce the results they would have everything they needed.

With the organizational structure created, two .csv files were downloaded from Kaggle and merged on their index(a unique alphanumeric accident identifier) to create an initial raw dataset of 57 features and ~2 million rows. Summary statistics on the features were computed and reviewed at this step to get an idea of the parameters of the dataset.

Columns containing null values were reviewed and appropriate actions were taken, depending on the context. Columns containing less than 2000(.1% of rows)null values, had their entries/rows, removed entirely. Other columns, containing many more nulls values, null values were imputed with the most common value(mode), or dropped all together if they were deemed to have no relevance to the problem under investigation.

This wrangled dataset was written out to .csv and passed to the next set of the process.

Exploratory Data Analysis

In this step of the DSM, features were explored further and eventually pared down to smaller feature-set than had been present in in the Data Wrangling notebook. Under the conditions of the problem or goal of this project, the features of interest are those that exist *before or during* the accident. Therefore, those that were not relevant to the problem were dropped. For example, one variable which was in the dataset, "*Did_Police_Officer_Attend_Scene_of_Accident*', would not be helpful, as it is something that happened post-accident, and would not be useful as a predictive feature for future accidents. Location-based features were also removed, as the goal of the final model is to generalize to an accident situation independent of location.

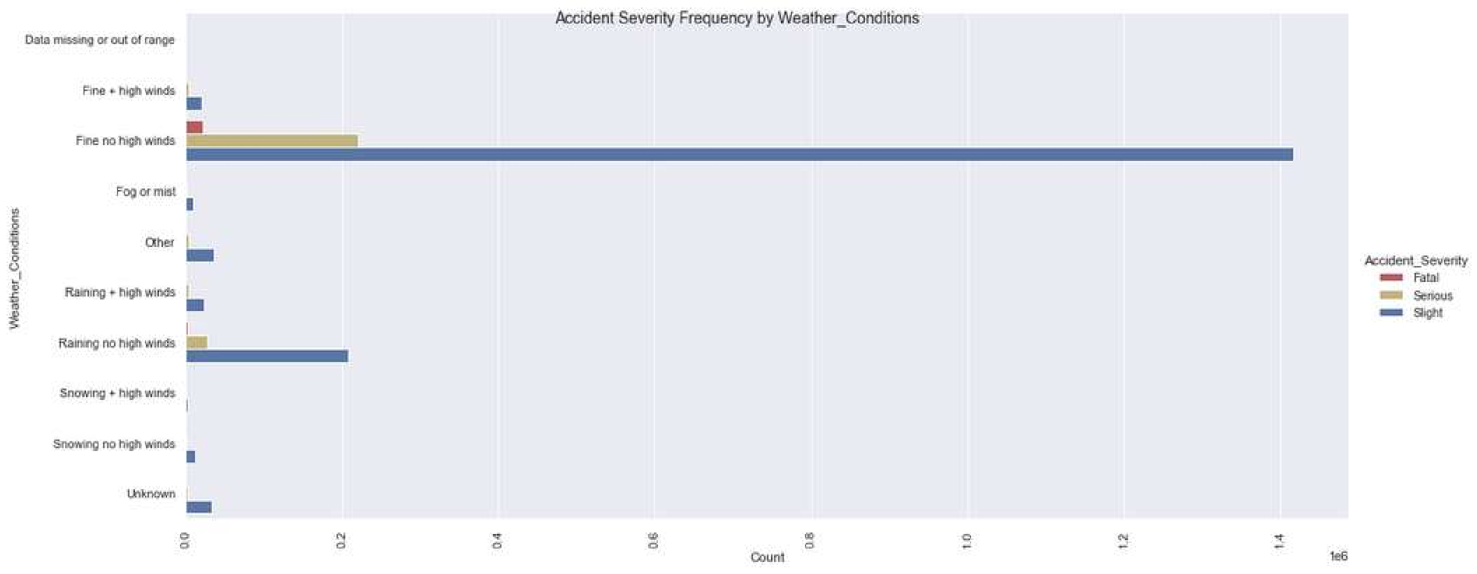
Below are some features that were dropped and their reasoning:

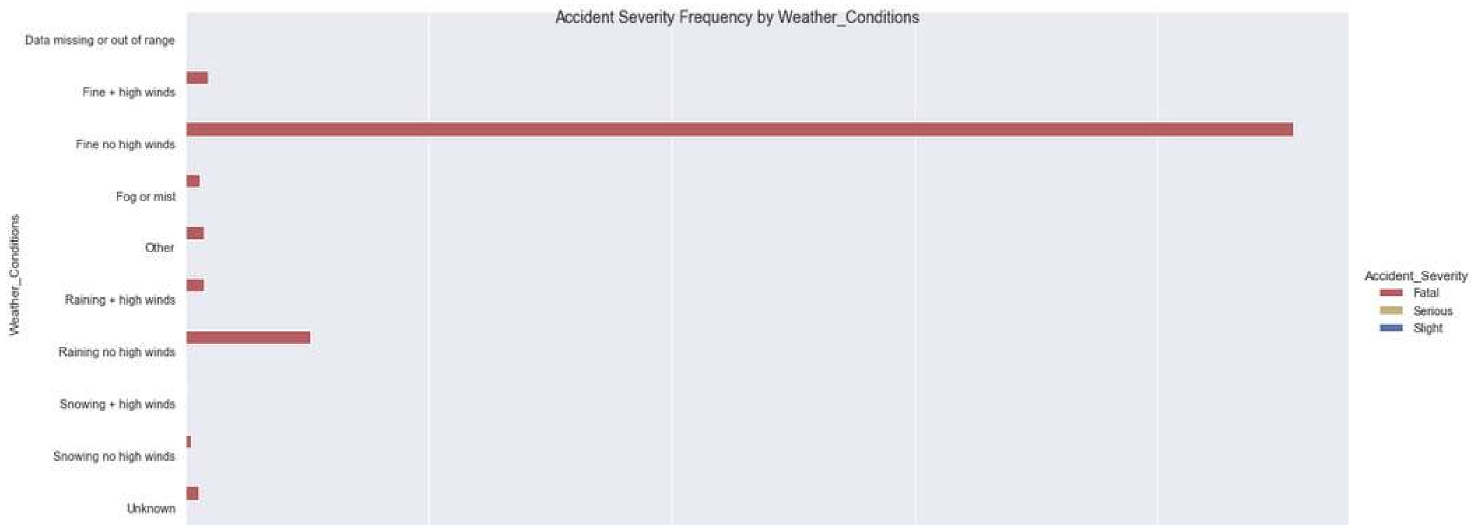
Column Dropped	**Reasoning**
'Did_Police_Officer_Attend_Scene_of_Accident',	Post incident
'Number_of_Casualties',	Post-incident
'Hit_Object_in_Carriageway',	Post incident
'Hit_Object_off_Carriageway',	Post incident
'Skidding_and_Overturning',	Post incident
'X1st_Point_of_Impact',	Post incident
'Carriageway_Hazards',	Post incident
'Police_Force',	Location based
InScotland,	Location based
'Local_Authority_(District)',	Location based
'Local_Authority_(Highway)',	Location based
'Driver_Home_Area_Type',	rural or urban etc. Location based
Driver_IMD_Decile,	location based - a calculated score
Number_of_Vehicles',	Post incident
'Year_x',	duplicate found in date
'1st_Road_Number',	irrelevant - location based
'2nd_Road_Number',	irrelevant - location based
'Latitude',	location based
'Longitude',	location based
Year_y,	duplicate found in date
'Urban_or_Rural_Area'	location based

At this point, summary histograms were viewed to give an idea of the initial distributions of numerical columns.

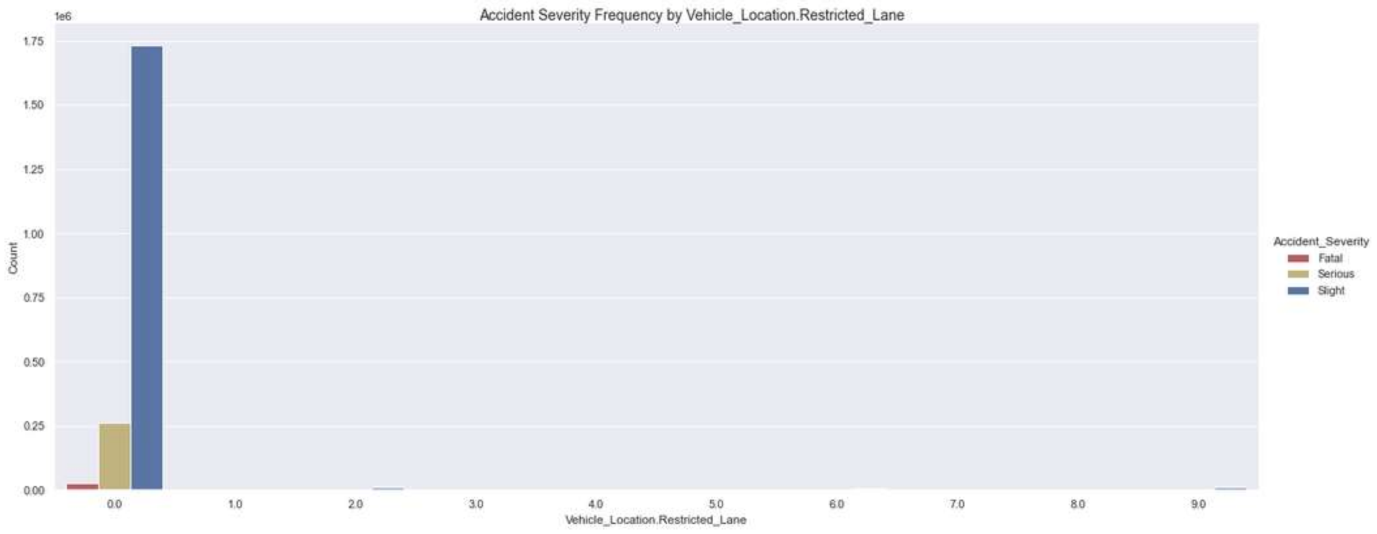
The python package "seaborn" was used to graph categorical values initially in side-by-side histograms showing distributions of every feature by resulting severity, Slight, Serious, or Fatal. For each visualization I produced I also produced a graph highlighting just the 'Fatal' cases.

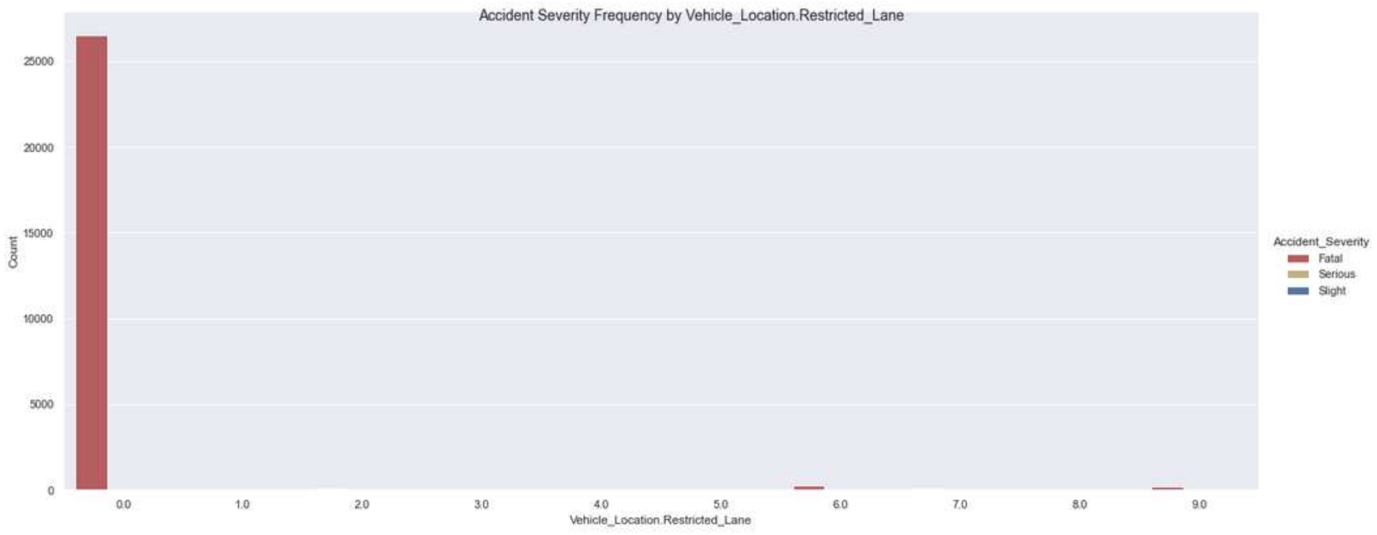
I wrote a quick function to streamline this as I had many categorical explanatory variables to review. The figure below highlights one such feature(Weather Conditions). The rest are available within the notebook.





After some exploratory data analysis, the visual representations of some features vs. the target highlighted that some features were not useful and were removed. For example, “*Vehicle_Location.Restricted_Lane*”, as all values were nearly identical, and it was removed. Figure below:





From this point the dataset is written out to a more cleaned csv file and I moved on the next notebook, “preprocessing”.

Preprocessing

In the preprocessing step, data was read into the notebook from the previous step. As I had dropped both the year columns, the date feature was converted to a DateTime feature, and columns relating to time series were created. Individual columns were created for the ‘Month’ and ‘Hour’(“Day of Week” was already present in the dataset). As the these Month and Hour are cyclical (repeating patterns), they can be used to generalize, and were set to ‘object’ data type as the will be treated categorically. These next steps sequentially were: The index was reset, dummy variables for categorical features were created, and the numeric features was scaled with StandardScaler, these two were merged together for the final explanatory(X) dataset. The use of the dummy variables expanded of the categorical features and resulted in a final feature count of 214. Data was split and written out to respective ‘X’ & ‘y’.csv files for explanatory and target variables.

Modeling

Data for X and y features were read into the notebook, and split into train and test sets for modeling. I chose a test size of 30%, holding that out for model validation. The training set had 1.4million rows, and 214 columns. The test set had approx. 600,000 rows with 3 columns/classes. I took a supervised-learning approach to solving this data-problem.

Initially, for the sake of simplicity, I was most curious how possible it was to predict just the “Fatal” response. I first started my modeling with the a logistic model, on a binary target, “Fatal”(1), or “Not ‘Fatal’(0). Within this model(and all models), I used RandomSearchCV to search the hyperparameters space and to perform cross validation. Modeling, I found, can be a computationally time-consuming activity, which is the reason I elected for RandomSearchCV over GridSearchCV.

With each models fit-time being substantial, I learned how to save and recall models when needed with the use of the ‘pickle’ python package. It allows a model to be written to file after it is fit, and recalled very quickly to produce metrics for comparison(or prediction).

I tried three different methods for predicting just the “Fatal” category, the other two being Random Forest and XGBoost Classifier, writing these models to .pkl files based on the result of RandomSearchCV.

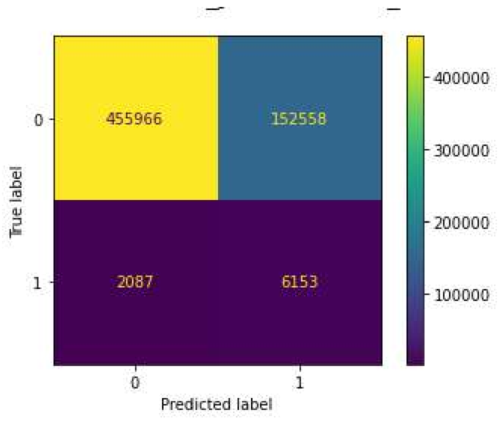
After looking at the binary outcome(Fatal or Not) I moved on to look at multiclass classification, starting with a logistic multiclass model, then moving to a Logistic with One Vs. Rest, Random Forest, and a model where I attempted the use of SMOTE(Synthetic Minority Oversampling Technique), to produce synthetic datapoints for my minority class, to attempt to balance out the imbalanced classes, and then fit an logistic model around this. Due to the computational time needed, I used RandomSearchCV with all of these methods to search/tune the hyper parameters of each model. See next page for Model Specifics.

Model Specifics:

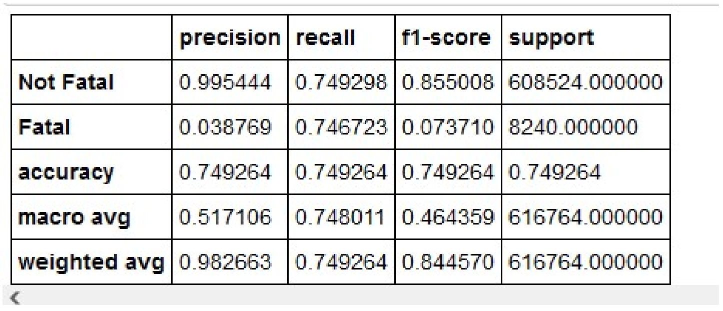
As these were classification problems, the most appropriate metrics for evaluation, across models, were the confusion matrix and classification reports. I produced ROC-AUC metrics as well in the notebook for the binary response, but did not find them particularly useful. The plot_confusion_matrix tool within Sklearn creates a clean plot showing the predicted vs actual in the test set.

Logistic Model 1 – Binary Target- Fatal(1) or Non-Fatal(0)

This first model shows a fairly acceptable recall score on both categories, but very low precision. There are ~8000 fatalities in the test set. The model predicts ~6100 of them correctly as Fatal, but does this at a cost of predicting many accidents Fatal, that are not, and suffers quite badly in precision. Overpredicting the Fatal accidents might not be the worst thing in the world in this context. Any application of a model in this area, someone would rather err on the side of caution for predicting severity.



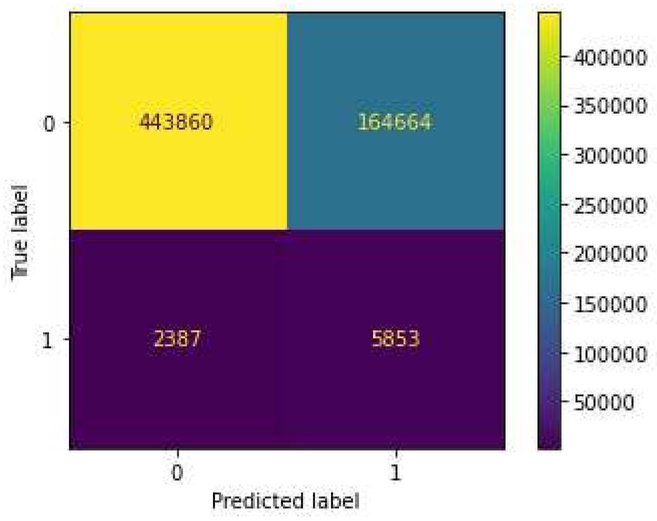
Here we have the Confusion Matrix for logistic Binary Model, Fatal is "1", Non-Fatal is '0'



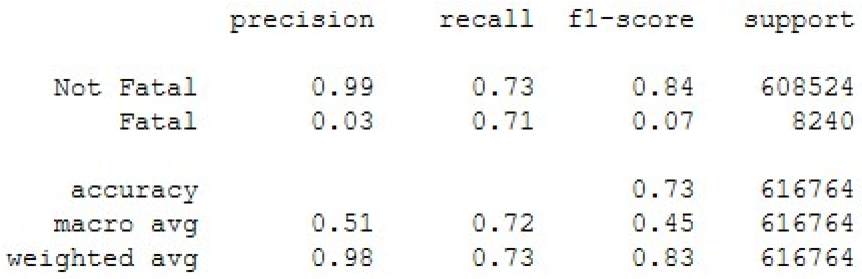
	precision	recall	f1-score	support
Not Fatal	0.995444	0.749298	0.855008	608524.000000
Fatal	0.038769	0.746723	0.073710	8240.000000
accuracy	0.749264	0.749264	0.749264	0.749264
macro avg	0.517106	0.748011	0.464359	616764.000000
weighted avg	0.982663	0.749264	0.844570	616764.000000

Random Forest – Binary Target- Fatal(1) or Non-Fatal(0)

The tuned Random Forest model I performed on the binary test set performed worse than the logistic model. Results of the confusion matrix and classification report can be seen below:



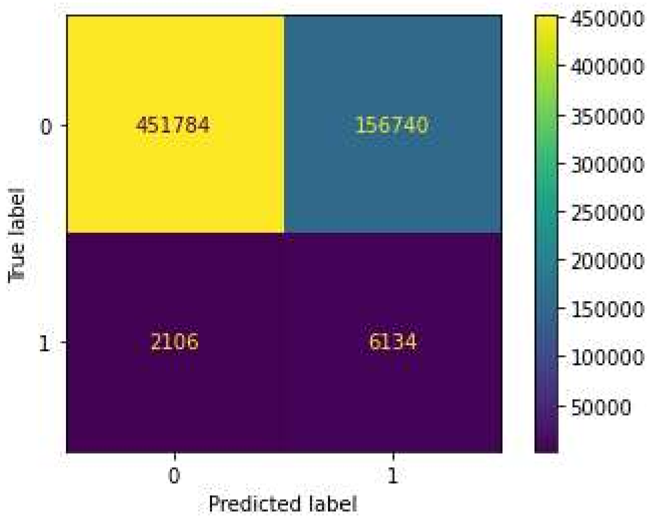
Confusion matrix and Classification report for this model



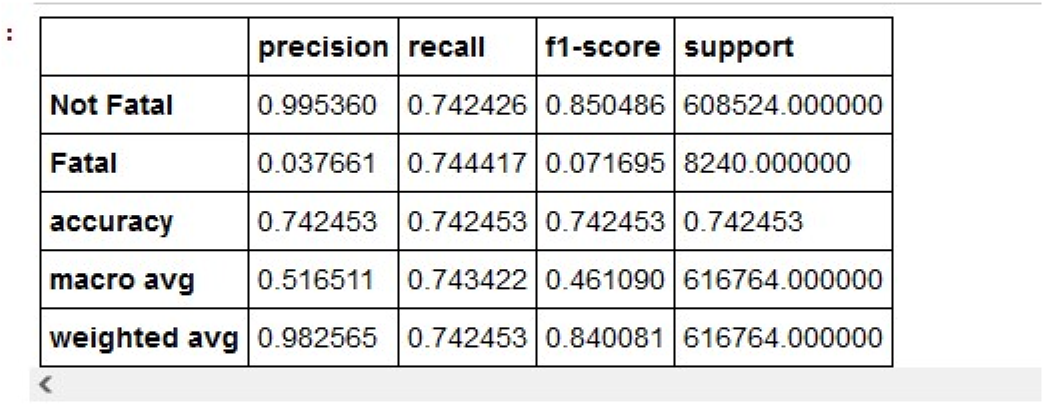
```
              precision    recall  f1-score   support

   Not Fatal       0.99      0.73      0.84    608524
       Fatal       0.03      0.71      0.07      8240

    accuracy                           0.73    616764
   macro avg       0.51      0.72      0.45    616764
weighted avg       0.98      0.73      0.83    616764
```

XGBoost Model - Binary Target- Fatal(1) or Non-Fatal(0)

The XGBboost model performed nearly as well as the logistic model, but still not quite as well.



Confusion matrix and Classification report for this model



	precision	recall	f1-score	support
Not Fatal	0.995360	0.742426	0.850486	608524.000000
Fatal	0.037661	0.744417	0.071695	8240.000000
accuracy	0.742453	0.742453	0.742453	0.742453
macro avg	0.516511	0.743422	0.461090	616764.000000
weighted avg	0.982565	0.742453	0.840081	616764.000000

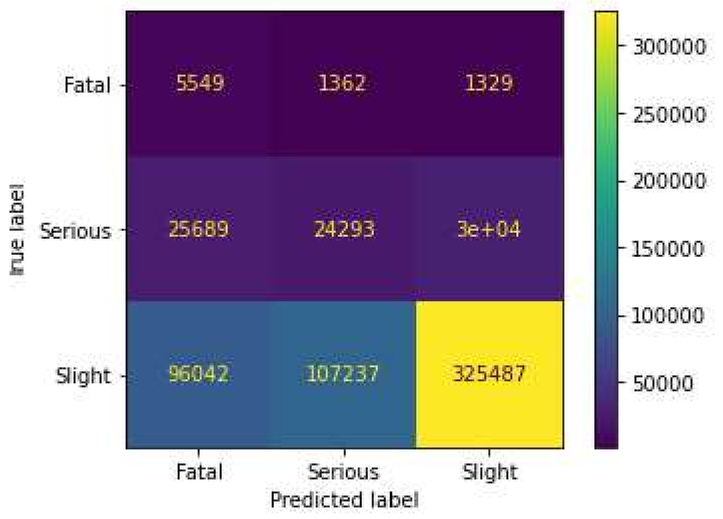
.

Multiclass Modeling

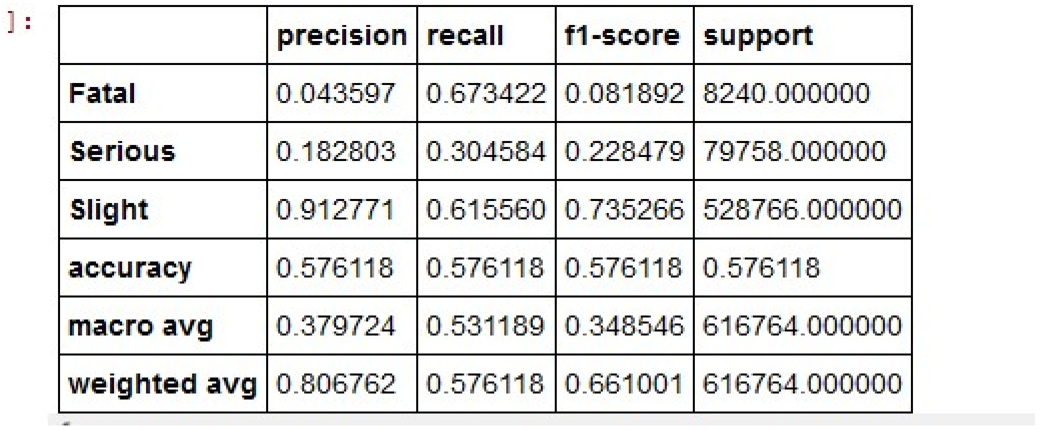
Logistic Multiclass

Moving on from binary classification, I went on to model multiclass outcomes(Fatal, Severe, Slight). I once again started with a logistic model, this time set up for multiclass classification. Results of this model are below:

Logistic Multiclass



Confusion matrix and Classification report for this model



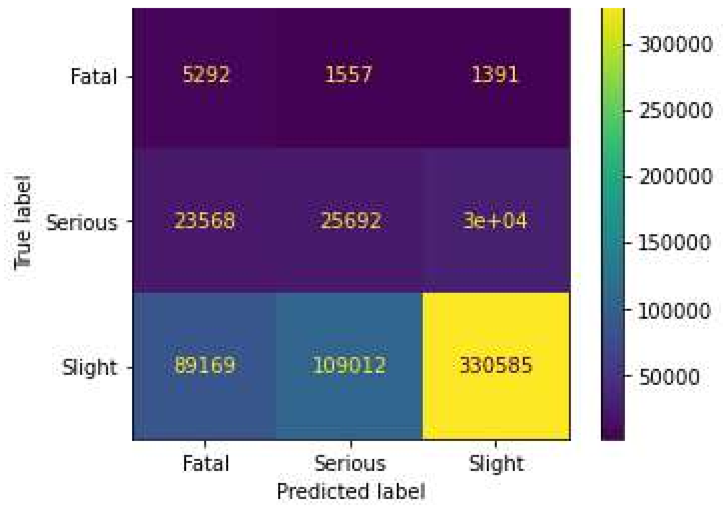
]:

	precision	recall	f1-score	support
Fatal	0.043597	0.673422	0.081892	8240.000000
Serious	0.182803	0.304584	0.228479	79758.000000
Slight	0.912771	0.615560	0.735266	528766.000000
accuracy	0.576118	0.576118	0.576118	0.576118
macro avg	0.379724	0.531189	0.348546	616764.000000
weighted avg	0.806762	0.576118	0.661001	616764.000000

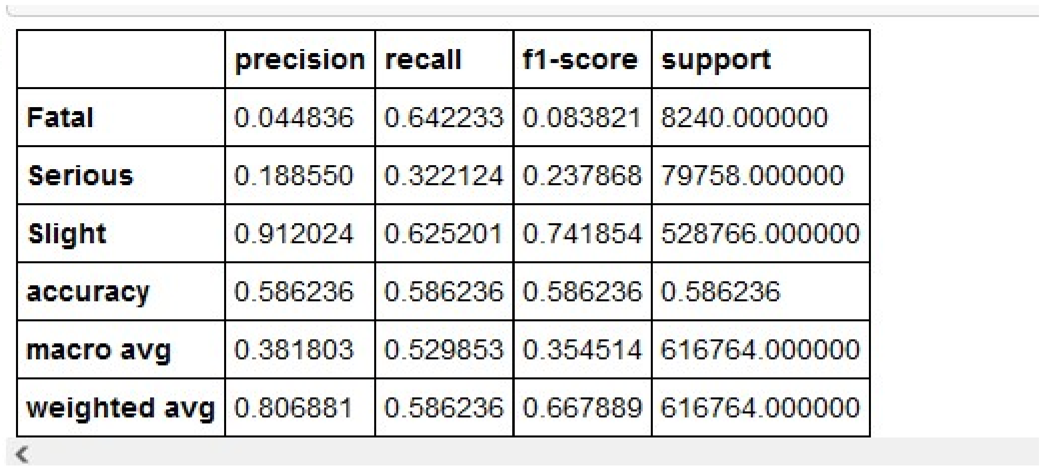
Results here show what I consider decent recall scores of classification for the fatal and slight classes, but fairly poor for the Serious Class.

Logistic model with One Vs. Rest

I fit this model as an attempt to get tweak/tune and better results from the logistic modeling method. It incrementally improved the accuracy of the model.(~1% improvement in Accuracy)



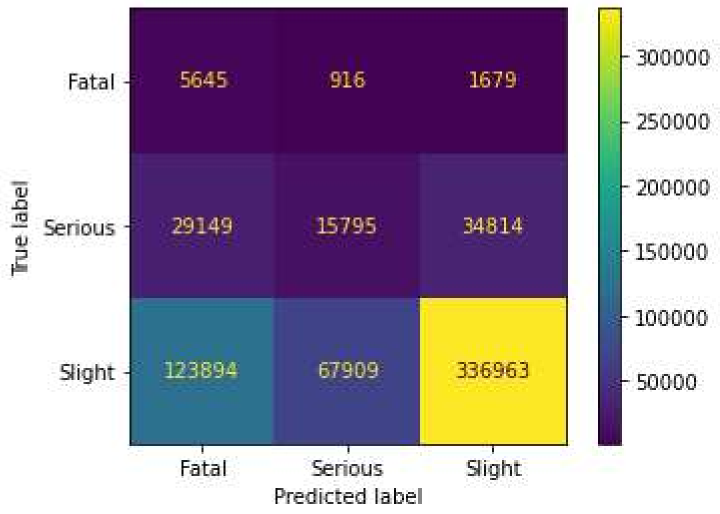
Confusion matrix and Classification report for this model



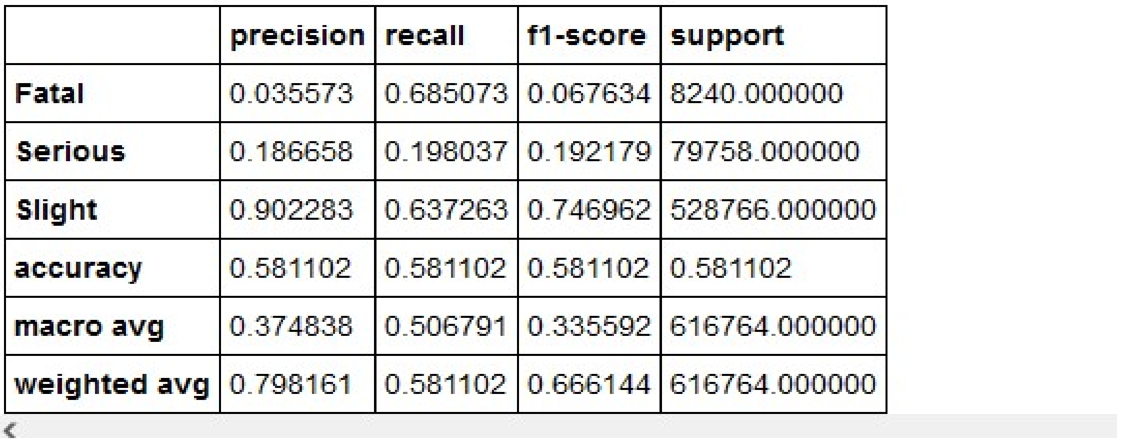
	precision	recall	f1-score	support
Fatal	0.044836	0.642233	0.083821	8240.000000
Serious	0.188550	0.322124	0.237868	79758.000000
Slight	0.912024	0.625201	0.741854	528766.000000
accuracy	0.586236	0.586236	0.586236	0.586236
macro avg	0.381803	0.529853	0.354514	616764.000000
weighted avg	0.806881	0.586236	0.667889	616764.000000

Random Forest - Multiclass

This model had a similar performance to the plain logistic model, while performing slightly better in the recall score for the Fatal category, while other classes suffered.



Confusion matrix and Classification report for this model



	precision	recall	f1-score	support
Fatal	0.035573	0.685073	0.067634	8240.000000
Serious	0.186658	0.198037	0.192179	79758.000000
Slight	0.902283	0.637263	0.746962	528766.000000
accuracy	0.581102	0.581102	0.581102	0.581102
macro avg	0.374838	0.506791	0.335592	616764.000000
weighted avg	0.798161	0.581102	0.666144	616764.000000

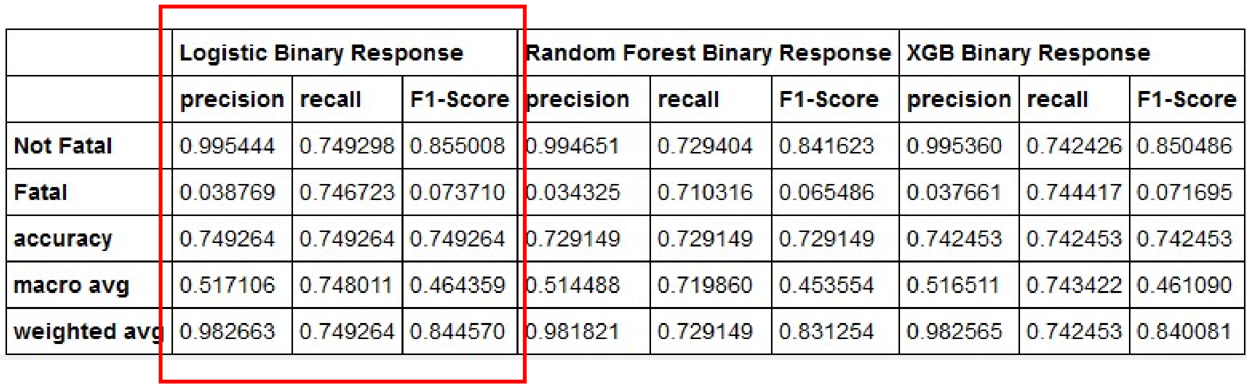
SMOTE – logistic Model

I also attempted the use of a SMOTE(Synthetic Minority oversampling Technique, in conjunction with a Logistic model, but results were so poor they are not worth discussing here.

Findings/Discussion/Best Model

With regard to performance I have two categories of modeling to discuss, binary outcomes models, and Multiclass outcomes models.

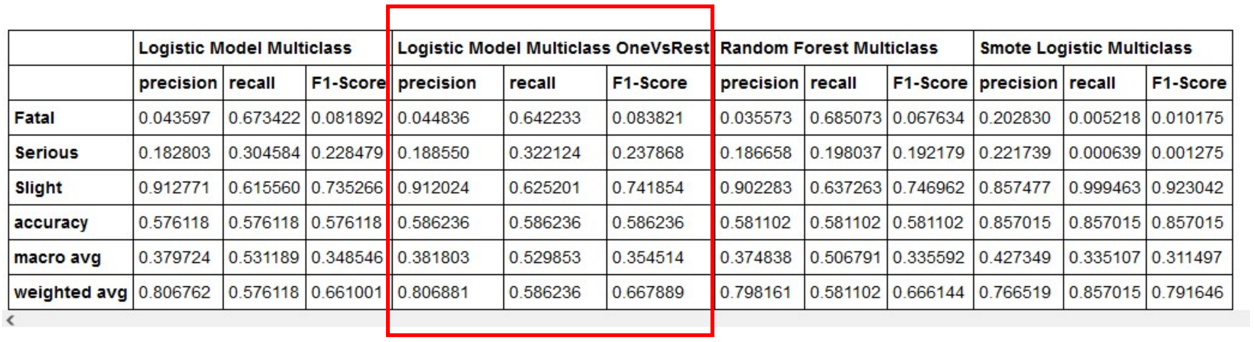
For the binary outcomes, below are summarized classification reports:



	Logistic Binary Response			Random Forest Binary Response			XGB Binary Response		
	precision	recall	F1-Score	precision	recall	F1-Score	precision	recall	F1-Score
Not Fatal	0.995444	0.749298	0.855008	0.994651	0.729404	0.841623	0.995360	0.742426	0.850486
Fatal	0.038769	0.746723	0.073710	0.034325	0.710316	0.065486	0.037661	0.744417	0.071695
accuracy	0.749264	0.749264	0.749264	0.729149	0.729149	0.729149	0.742453	0.742453	0.742453
macro avg	0.517106	0.748011	0.464359	0.514488	0.719860	0.453554	0.516511	0.743422	0.461090
weighted avg	0.982663	0.749264	0.844570	0.981821	0.729149	0.831254	0.982565	0.742453	0.840081

To pick one, the best results are the simple logistic model. It correctly predicts 74.9% of true fatal cases and 74.6 % of true Non-Fatal cases. It does this while producing a lot of false positives(false fatalities).

For the Multiclass outcomes, here are the summarized classification reports:



	Logistic Model Multiclass			Logistic Model Multiclass OneVsRest			Random Forest Multiclass			Smote Logistic Multiclass		
	precision	recall	F1-Score	precision	recall	F1-Score	precision	recall	F1-Score	precision	recall	F1-Score
Fatal	0.043597	0.673422	0.081892	0.044836	0.642233	0.083821	0.035573	0.685073	0.067634	0.202830	0.005218	0.010175
Serious	0.182803	0.304584	0.228479	0.188550	0.322124	0.237868	0.186658	0.198037	0.192179	0.221739	0.000639	0.001275
Slight	0.912771	0.615560	0.735266	0.912024	0.625201	0.741854	0.902283	0.637263	0.746962	0.857477	0.999463	0.923042
accuracy	0.576118	0.576118	0.576118	0.586236	0.586236	0.586236	0.581102	0.581102	0.581102	0.857015	0.857015	0.857015
macro avg	0.379724	0.531189	0.348546	0.381803	0.529853	0.354514	0.374838	0.506791	0.335592	0.427349	0.335107	0.311497
weighted avg	0.806762	0.576118	0.661001	0.806881	0.586236	0.667889	0.798161	0.581102	0.666144	0.766519	0.857015	0.791646

For my choice in this area, I would once again choose the Logistic Model, but the one with the OneVsRest modification for its results. It is a middle ground model, correctly identifying 64% of true fatal accidents, 32% of true serious accidents and 62% of True Slight accidents, while having the highest Accuracy score among the three models (SMOTE model ignored) of 58.6% accuracy.

Use Case Recommendations

The use case for this project was motivated to see if it is possible to predict the level of severity of vehicular accidents. As described above, it is, at least somewhat effective, and I can imagine several use cases for this model.

Within our explanatory variables, all were present before or during the accident, so they should be able to be measured via a computer or sensor system in a "live" context. A live system, working in tandem with this model, could hopefully recognize when the conditions exist that result in a Fatal/Serious/Slight accident.

The use case I imagined for this model, was a for a vehicle manufacturer. A manufacturer could embed this model(or hopefully a better version of this model), within newer vehicles computers systems. A car then, recognizing certain conditions exist(ex. 35 year old driver, conditions raining/during rush hour etc, on a Monday) and give warning to the driver, potentially a message like, "the conditions exist and predict if you were to be involved in an accident it is predicted to be: Serious, Slight, or Fatal, please consider postponing your trip until these conditions change"

Another use case for a model such as this would be for an insurance company. This particular feature-set would have to be tweaked a bit to make it applicable, because the conditions of the road surface conditions, speed limit etc, can't be known to the insurance company in advance. It could be put in production at the level of an the insurance companies call-center line or app, where knowing the insured party's parameters(age, vehicle-type, engine_size, time of day etc…)the insurance company could anticipate severity of the accident, and prioritize certain calls over others.

A final application could be for government emergency services, maybe whereby a caller gives their driver license number to the operator, the system grabs or infers as many feature variables as possible, age, determine make/model, time of day, weather conditions, and predicts the level of severity. An emergency system, i.e. 911, that was working at capacity, could triage or prioritize response to various accidents based on the predicted severity.

Suggested Improvements and Future Research

There are ideas I have for future work with this dataset, which I didn’t perform, either because of a need for expediency, or because I had the idea after the fact, or did not have the technical expertise to perform them at the time.

One such thing to go back and look at is accident location. I removed all location based variables from the dataset, because I wanted to be able to generalize new data, independent of location, however I have no doubt it would be a valid predictor. Certain intersections are more dangerous than others, certain roads can be more dangerous etc, the main problem being that the location based data in this dataset is in the UK, so it could only generalize to new UK accidents.

I would also go back and spend more time with the SMOTE tool. The process seems like it had the opportunity to produce better results, but I was not well equipped enough at the time(in knowledge/time constraints) to use it properly I believe.

If time were not a factor, I would also try many more modelling techniques. I might try some unsupervised methods of classification, like KNN etc.