



SAN JOSÉ STATE UNIVERSITY

Final Project Report Prediction of Accident Severity

Group 4

Indira Priyadarshini Bobburi (012445927)

Jainul Patel (012463737)

Sai Teja Desu (012455287)

Section 1: Introduction

Motivation

Road accidents are a serious concern for the majority of nations around the world because accidents can cause severe injuries and fatalities. According to the World Health Organization's Global Status Report, approximately 1.25 million people deaths happened per year are because of road accident injuries, and most fatality rates were in lower income countries [1]. Our motivation is to predict the accident severity of any road, which will play a crucial factor for traffic control authorities to take proactive precautionary measures. In addition, the dataset we chose was rarely solved in prediction point of view, so we took this opportunity to predict the severity of the accident. Also, we chose this highly imbalanced dataset as we wanted to apply acquired concepts as part of CMPE 255 course.

Objective

The purpose of this project is to predict the severity of an accident by training an efficient machine learning model with the help of existing accidents data from 2005-2015. This project is majorly focussed on predicting rarer classes accurately such as Serious and Fatal.

Section 2: System Design & Implementation details

Algorithms, technologies, and tools

Seven classification algorithms were used and evaluated to predict the accident severity. Algorithms considered for classification were K-nearest neighbors, Naive Bayes classifier, Random Forest classifier, Logistic Regression, Gradient Boosting classifier, XGBoost classifier, SVM. The first thing that came to our mind is that severity is based on different decisions like road, weather conditions. So, we tried different decision tree classifiers. We also tried ensemble methods as the data is very imbalanced.

K-Nearest Neighbors Classifier: This model is a non-parametric method used for classification. We tried using different values of neighbors and got the best result for considering three neighbors for each point and weights parameter was set to 'distance' which weights the points by an inverse of their distances. Closer neighbors of a query point will have a greater influence than neighbors which are farther away. This model didn't perform well on this dataset probably because the data was multidimensional, the points were very close to many data points which were classified in different classes. F1- score of Accident Severity prediction from this model was 0.57.

Naive Bayes Classifier: This model is a probabilistic framework for solving classification problems. Since our features are independent of each other, this model works well to predict the values for a given set of features. As the features selected are discrete, we used multinomial Naive Bayes classifier instead of Gaussian. We tried this algorithm after data

cleaning on the entire dataset which gave an F1-score for 'serious' severity prediction of 0.04. The same on an undersampled data with an equal ratio of different accident severity gave an 'serious' severity prediction F1-score of 0.63

Random Forest classifier: As our data is very imbalanced, we tried ensemble methods, of which random forest classifier is one. A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. We tried this algorithm with different class weights such as incremental, multiplicative, exponential weights. All of these gave very bad F1-score for 'severe' output class. The inbuilt class weight - `balanced_subsample`, gave us the best results. The F1 scores are .60 for 'severe' class and 0.62 overall for the undersampled data

Logistic Regression: This model is the basic and popular for solving classification problems. Unlike Linear Regression, Logistic regression model uses a sigmoid function to deal with outliers. Class weights parameter sets the weights for imbalanced classes by adjusting weights inversely proportional to class frequency. As the dataset was highly imbalanced, even after using 'balanced' class weight, the model did not perform well. Using the undersampling method, the majority class which was the 'Slight' accident type was undersampled. On this undersampled data, logistic regression model performed better than using 'balanced' class weights. As both the classes were not exactly separable, this model gave the f1-score equals 0.62.

Support Vector Machine: SVMs are based on the idea of finding a hyperplane that best divides a data set into two classes. As the features in this dataset were very sparse and non-separable by any hyperplane, SVM did not work at all for this dataset. For larger dataset, SVM training time is high. When we tried to train the dataset using SVM it ran forever and did not fit the model.

Xtreme Gradient Boosting(XGBoost): As part of trying out different ensemble methods, we have selected one of the most popular distributed gradient boosting technique. This algorithm, in general, performs faster compared with other algorithms due to the parallel tree boosting method [2]. We have tried this algorithm on under-sampled data (80k records) and randomly sampled data (100k records). The positive of this algorithm is the precision values for "Serious" class is very good (70%) however, xgboost gave worst recall values for "Serious" class. Tried a different set of parameters for improving the recall but the algorithm was not able to improve the results. Moreover, we had a challenge in applying this algorithm to the full dataset using HPC but unable to import the library due to permission issues.

Gradient Boosting Machine (GBM): Gradient Boosting is another type in ensemble method which is very popular in decision tree algorithm. GBM performs well in reducing mean squared error (MSE) [3]. This algorithm performed well on undersampled data in terms of both precision and recall values, however, the algorithm is not able to predict even one rarer class when applied on full data set.

Technologies & Tools used

For developing this project, below tools and technologies have been used.

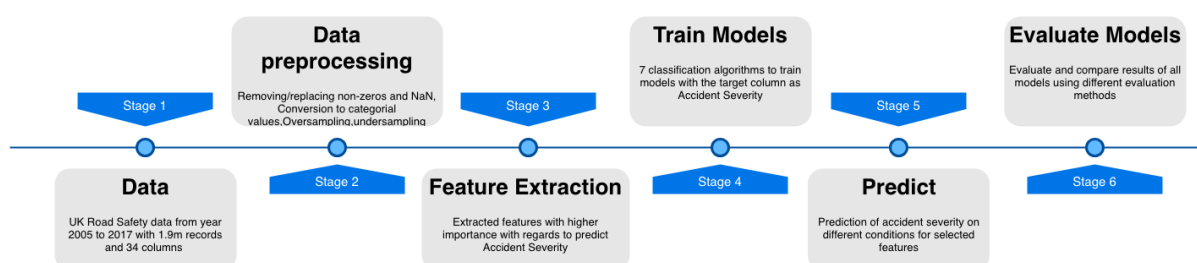
Python: Python is easy to understand language and has a rich set of libraries to use for data pre-processing, modeling, and evaluating the algorithms. Moreover, python has very good community support which is very useful for debugging the code.

Jupyter Notebook: Jupyter notebook is a simple and interactive tool for running python code. Also, it has many different sets of features such as downloadable to .py, .ipynb, and .html files.

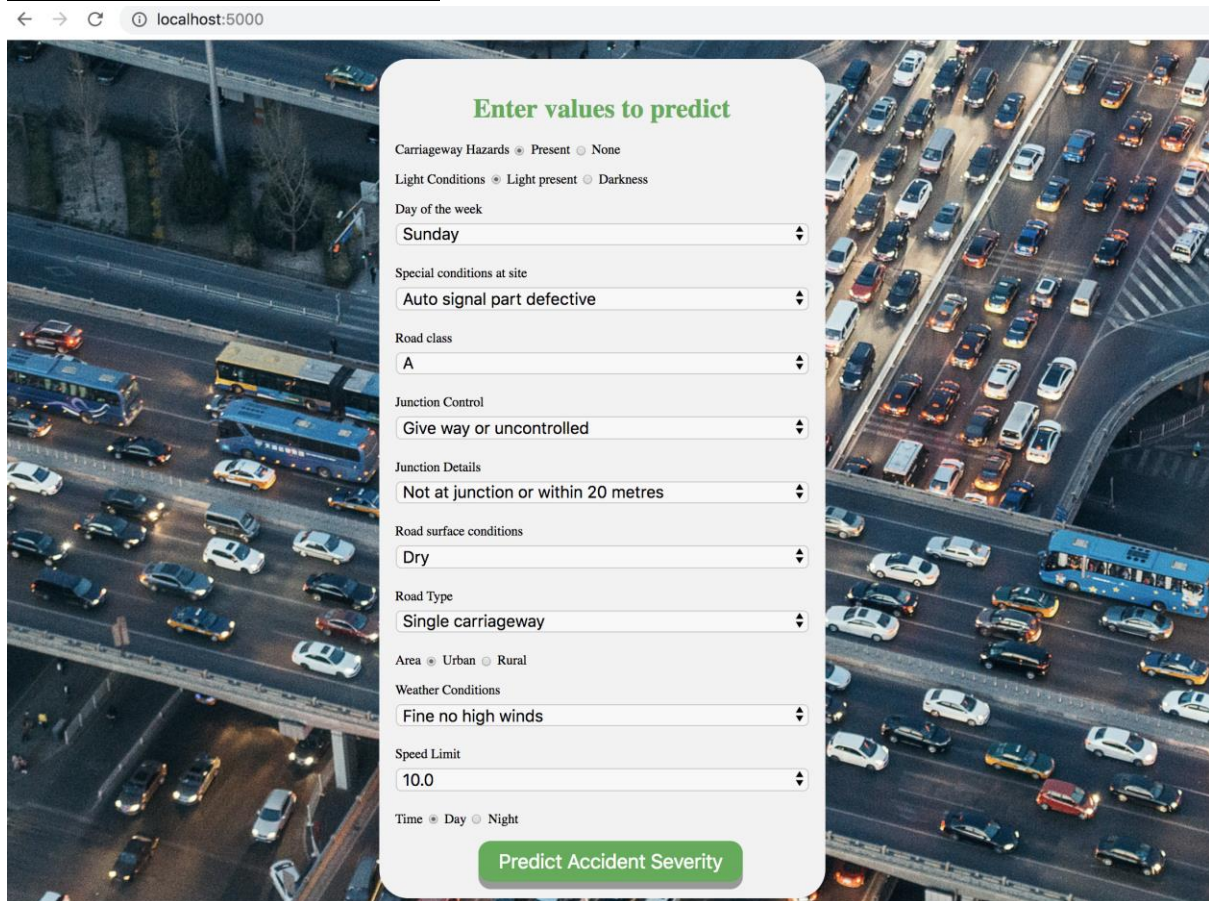
Flask: Flask is a web framework which generally used for developing web apps with python as backend service. Flask also has rich extension support which can be useful for installing different libraries.

HTML/CSS/JavaScript: HTML and CSS are the web design framework languages used for designing the front end of the web apps.

Model Flow Diagram



Graphical User Interface (GUI)



← → ↻ 📄 localhost:5000

Enter values to predict

Carriageway Hazards ☒ Present ☐ None

Light Conditions ☒ Light present ☐ Darkness

Day of the week
Sunday

Special conditions at site
Auto signal part defective

Road class
A

Junction Control
Give way or uncontrolled

Junction Details
Not at junction or within 20 metres

Road surface conditions
Dry

Road Type
Single carriageway

Area ☒ Urban ☐ Rural

Weather Conditions
Fine no high winds

Speed Limit
10.0

Time ☒ Day ☐ Night

Predict Accident Severity

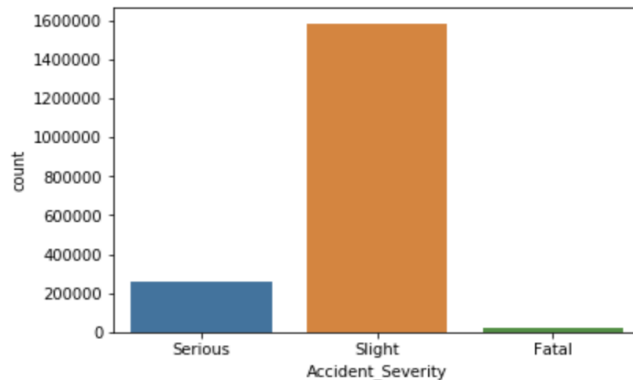
The outcome of the best model is integrated with a web app which gives the prediction of severity by passing the different parameters of the road. The above figure is the reference of the web app developed as part of this project.

Section 3: Experiments / Proof of concept evaluation

Dataset used:

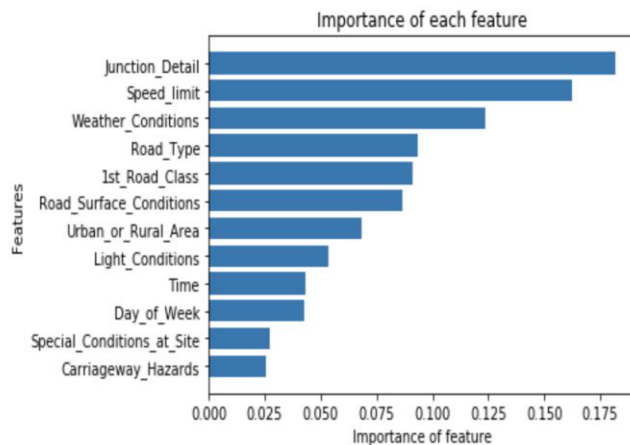
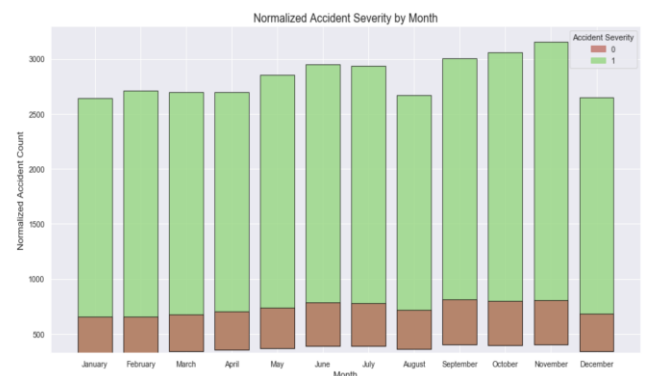
We have used UK road traffic accidents dataset for our project. The dataset contains 1,920,000 records and 34 columns including weather conditions, Road class, road type, junction details, road surface conditions, light conditions, etc. Our dataset is very imbalanced with data corresponding to *slight* severity is 84.84%, *serious* severity is 13.86% and for *fatal* severity is 1.30%. For validation of data, we have separated the dataset based on the accident severity and choose data for train and test dataset in equal ratios. We have used k-fold validation with 5 folds from each subset.

Data Visualization



UK Road Safety data: Total accident counts with accident severity as Slight, Serious and Fatal

Normalized accident counts each month for slight and (Serious and Fatal clubbed)



Plotting importance of each feature for considered features

Methodology followed

Data Pre-processing techniques: The dataset is imputed by replacing NaN and missing values with most frequent values of the corresponding column. All the categorical values have been labeled by integers from 0 to n for each column. Time has been converted to categorical feature with 2 values i.e., daytime and night time.

The data is visualized for correlation. Negatively correlated features are selected to be dropped. Feature importance is plotted to visualize and only features with high importance are taken into consideration for predicting accident severity.

The multi class label is converted to binary class by merging “Serious” and “Fatal” to Serious class.

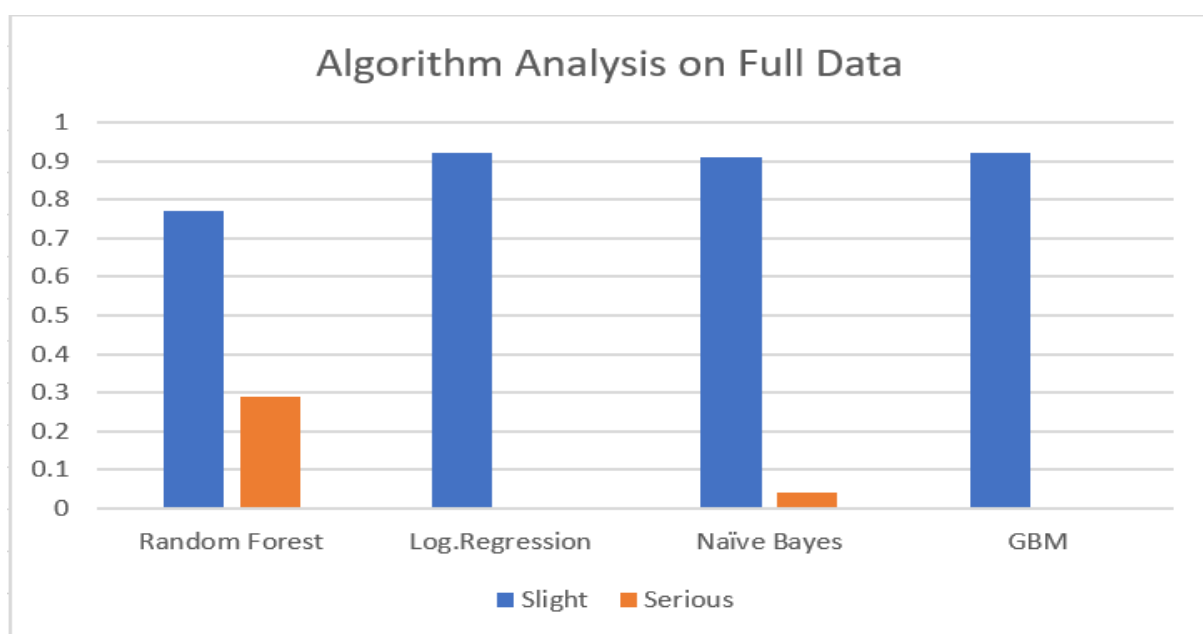
Feature Selection: The dataset has 34 attributes describing the incident of an accident. There are mixed types of data such as continuous and categorical. Manually dropped few columns due to its inconsistency in values such as Accident ID, and Location ID. For selecting the best features, below functions are used from sklearn library [4].

1. SelectKBest: SelectKBest is a sci-kit learn library provides the k best features by performing statistical tests i.e., chi squared computation between two non-negative features. Using chi squared function filters out the features which are independent of target attribute [5].

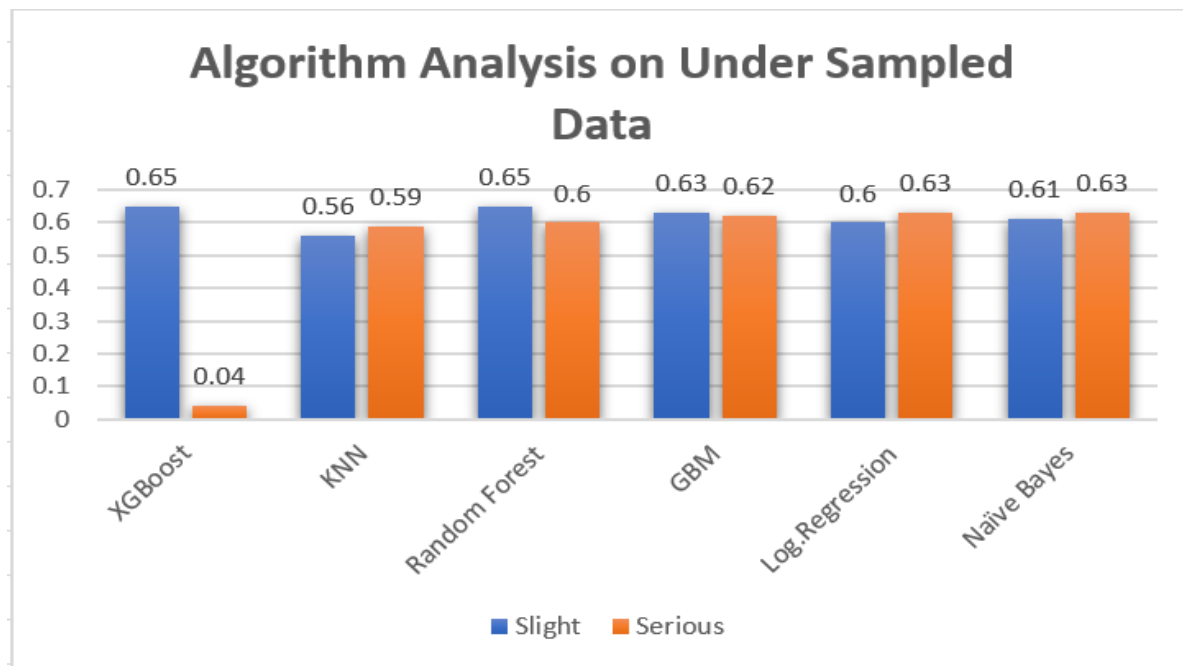
2. Recursive Feature Elimination (RFE): RFE runs the defined model by trying out different possible combinations of features, and it removes the features recursively which are not impacting the class label [4]. Logistic regression algorithm is used as a parameter for RFE to decide on features.

Comparative Analysis of Algorithms

1. Below bar plot shows the comparative analysis of algorithms on Full Dataset. This figure shows that most of the algorithms were unable to predict “Serious” class label.



2. Below bar plot shows algorithm analysis after under sampling the data. This figure shows that almost every algorithm performs well after under sampling the most frequent class label.



Section 4: Discussion & Conclusions

Decisions, difficulties and discussions:

Our main aim was to predict the severity of the accident when it is “serious” and “fatal”. It was very difficult to handle this large-sized data. Using HPC we were able to run most of our algorithms. Data is highly imbalanced so even though most of our algorithms were giving > 89% accuracies, it was of no use. It was predicting all the accidents as slight accidents. After checking on all these algorithms, the team even tried dimensionality reduction techniques and but the results were not improved. Then the team decided to use the undersampled dataset as it was giving better results in predicting the severe/fatal accidents. This decision was made on trying out oversampling, undersampling, test and train data with an equal ratio of classification classes.

Conclusion and Future work:

In conclusion, most of the algorithms are biased towards most frequent class. However, efficient pre-processing and corresponding imbalanced data techniques should give optimal results. Based on the current known conditions of weather, light, traffic signal, road surface, speed limit etc., accident severity can be classified. But there is no one feature, that influences the accident severity.

Future work involves considering region from latitude and longitude and this problem can be turned to regression problem. We can then predict the risk of accident in the given region. If the risk is higher then immediate actions can be taken.

Section 5: Project Plan / Task Distribution

Initial data cleaning and feature selection methods were discussed and implemented by all three of us. When we weren't getting good results on our imbalanced dataset, we came up with three different approaches.

Indira has separated the data with different output classes (serious, slight and fatal) and did k-fold validation considering that the train and test data in equal ratios for these classes. Saiteja has done the undersampling of the major class - *slight* and clubbed the *serious* and *fatal* classes. Jainul has done oversampling of minority classes using the SMOTE by imblearn.

Although everyone has worked on all models and preprocessed data together by taking progressive decisions, Indira has tried Naive Bayes Classifier, Random Boost classifier, Gradient Boosting Classifier. SaiTeja has tried for different parameters optimizing it for our dataset on XGboost classifier, Random Forest classifier. Jainul has tried KNN classifier, Logistic Regression.

Indira has worked on creating the front-end of the application and integrated the code with best results from all our Jupiter notebooks to a single backend. Jainul has worked on the data visualization plots for easier understanding of the data. Saiteja also worked on some part of data visualization and continued to work on algorithm tuning to achieve better results.

Project Github Repository: <https://github.com/IndiraBobburi/accident-severity-prediction/>

References:

- [1] Global Status Report on Road Safety 2015
http://www.who.int/violence_injury_prevention/road_safety_status/2015/en/
- [2] <https://github.com/dmlc/xgboost>
- [3] Select How To Explain Gradient Boosting
<https://explained.ai/gradient-boosting/index.html>
- [4] Feature Selection For Machine Learning in Python
<https://machinelearningmastery.com/feature-selection-machine-learning-python/>
- [5] https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.chi2.html#sklearn.feature_selection.chi2