BACKGROUND

Every year, there are hundreds thousands of road accidents in Canada, with tens of thousands injured and thousands of fatalities. Nowadays there are big datasets available that contain details about road accidents. Therefore we are hopeful that we can use this data to find more insights about accidents which can help us reduce the accident numbers.

DATASET

Because of the above, we choose to work on the data set of accidents in France from 2005 to 2016 from Kaggle in order to learn new information from the dataset to reduce accident numbers.

Source: https://www.kaggle.com/ahmedlahlou/accidents-in-france-from-2005-to-2016/home

All data consists of four main tables: **Characteristics, Places, Users and Vehicles**. We have decided to join tables based on a common variable that has been in all tables – Accident ID, and then also join it with the Vehicles table based on Vehicle ID. Thanks to that, we ended up with a DataFrame with 1,875,983 rows and 51 columns. Each row is corresponding to one user that was in an accident.

There also was a table available with information about which days during these years were Holidays. We will later use this information during feature engineering.

A new variable was created – **Severity**, which is equal to 1 when a user that took part in an accident has died and 0 when they did not die. This will be our target variable that we will be trying to build a model for. We want to understand how different variables influence the lethality of an accident.

TOOLS USED

Python

a. Pandas, numpy – for data exploration and manipulation
b. Matplotlib, seaborn – for visualisation
c. Sklearn – for modeling

DATA PREPARATION

MISSING VALUES

	Number of missing values	% of missing Values
RouteName	1796682	95.77%
RouteNumber	1127003	60.08%
Longitude	1043737	55.64%
Latitude	1043727	55.64%
GpsCoding	1035269	55.19%
PRDistance	916802	48.87%
HomePRNumber	914010	48.72%
PostalAddress	337264	17.98%
RoadNumber	124484	6.64%
Place	100366	5.35%
SafetyMeasures	43458	2.32%
CentralLaneWidth	21272	1.13%
OuterLaneWidth	18297	0.98%
NumberOfLanes	4063	0.22%
ReservedLane	3769	0.20%
SchoolProximity	2914	0.16%
Infrastructure	2899	0.15%
LaneShape	2443	0.13%
prof	2399	0.13%
DateOfBirth	2351	0.13%
SurfaceCondition	2310	0.12%
AccidentSituation	2278	0.12%
TrafficRegime	1781	0.09%
PedestrianAction	1773	0.09%
PedestrianGroup	1706	0.09%
PedestrianLocation	1664	0.09%
obs	790	0.04%
obsm	677	0.04%
trajet	369	0.02%
manv	234	0.01%
choc	133	0.01%
AtmosphericConditions	116	0.01%

The table above shows us number of missing values and % of missing values for variables that have missing values. Depending on the situation we had different approaches to solve this situation.

1. For variables where % of missing values is larger than 40%, we decided to drop these columns entirely. (For example: RouteName, RouteNumber)

2. Look closely at why some variables may have missing values and if there is a way to infer its proper values then fill them accordingly:

a. For Place variable, all missing values were for users whose UserCategory was Pedestrian. For these users we created a separate Place category equal to 0, which means not applicable.
b. Similar situation for SafetyMeasures, where missing values were for Pedestrians. This makes sense, as there aren't any common safety measures for pedestrians. Like before we grouped these results in another group.

3. Rest of the missing values had low percentages of missing values (under 1%). That's why we decided to fill these values depending on whether a variable is categorical or continuous in two ways:
    a. Categorical – fill missing values with median values of a variable.
    b. Continuous – fill missing values with mean values of a variable.

VEHICLES VARIABLES

Unfortunately, we are lacking descriptions of variables for all Vehicle data. We only have some explanation of Vehicle Category Variable (catv), however it's incomplete. Due to a lack of information regarding these variables we decided to not take these variables into account during our analysis.

FEATURE ENGINEERING

We decided to create a few new features based on existing ones. We feel like these new variables may describe the problem better and thanks to that we may end up with a better model:

a. AgeRange – categorical variable that says to which age range a user belongs. Calculated based on a date of an accident and a date of birth of a user that had an accident.
b. SafetyUsed – categorical variable that says whether safety measures were used or not. Calculated based on the SafetyMeasures variable.
c. Rush Hour – categorical variable that says whether an accident took place during rush hour. Calculated based on exact time of an accident. Rush hours assumed as between 7:30 – 9:30 and 16:00 – 18:00.
d. Holidays – categorical variable that says whether an accident took place during a holiday. Calculated based on Holiday table which was available on Kaggle.

DROPPING UNNECESSARY FEATURES

After dealing with missing values and creating new features we decided to drop features that will not be useful during the process of creating a model (for example: AccidentID, VehicleID, PostalAddress, etc.)

ANALYSIS

PREDICTIVE DATA ANALYSIS – BUILDING A MODEL

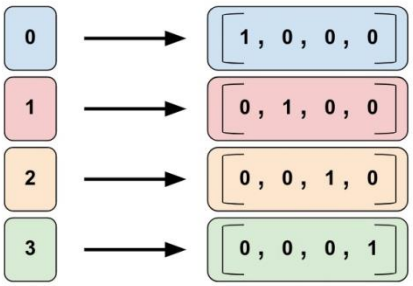
In this part we will use sklearn library to build models that predict whether an accident is going to be fatal or not. This is a problem of supervised learning known as a classification problem. The target variable is Severity, which is equal to 1 if the user died during an accident and 0 if the user did not die. The analysis consists of multiple steps.

1. **Splitting our dataset into train and test set.**

We decide do split our dataset in a 75% vs 25% ratio, into train and test set respectively. Also, we used a stratified split to make sure that the ratio of positive samples to negative samples of our target variable is the same in both train and test sets.

2. **Preparing the Dataset for the Machine Learning algorithms**

Most of our variables are categorical. In order for machine learning algorithms to work well on such data, we need to perform One-Hot-Encoding. Below is a picture presenting an idea behind the process.

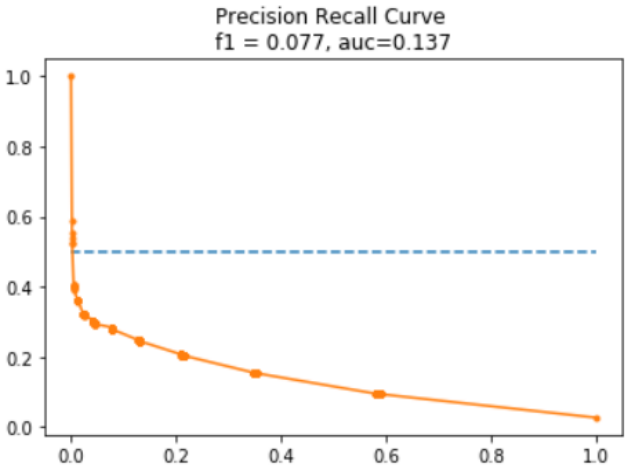


One-Hot-Encoding is a representation of categorical values as binary vectors. For each categorical column, data will be split into the amount of columns that is equal to the amount of categories within that variable. This is why the size of our dataset has changed after One-Hot-Encoding, before we had 30 columns, after we have 201 columns.

3. **Training a Random Forest Classifier algorithm with basic parameters.**

After the data is prepared we decided to train a basic Random Forest Classifier on our Data so we could have a point of reference and see if there are any issues that we need to fix.

The model that we build had 97% accuracy on Test Set, which seems very high. However we decided to look at the Precision-Recall curve to make sure that it works correctly.



After seeing the Precision-Recall curve we understand that the model had high accuracy because it almost always predicted 0 for our dataset. This is because we have a highly imbalanced dataset. Fewer than 3% of values for our target variables are equal to 1.

4. Undersampling the dataset.

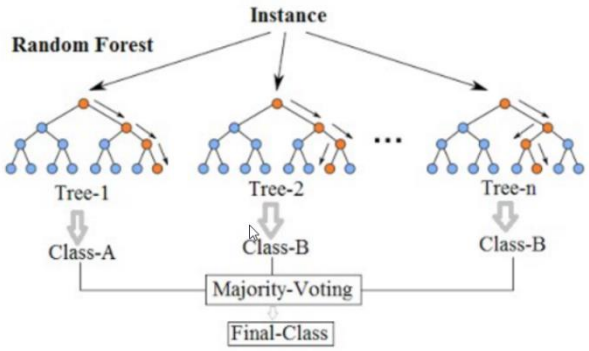
Because our dataset is highly imbalanced, we decide to undersample our train and test sets. This means that we will keep reducing the amount of rows of the majority class (Severity=0) for our target variable until it is close to the amount of rows of the minority class (Severity=1). The rows that we are removing are chosen randomly to reduce possible bias.

5. Training two models on undersampled dataset

We decide to train two models on undersampled dataset.

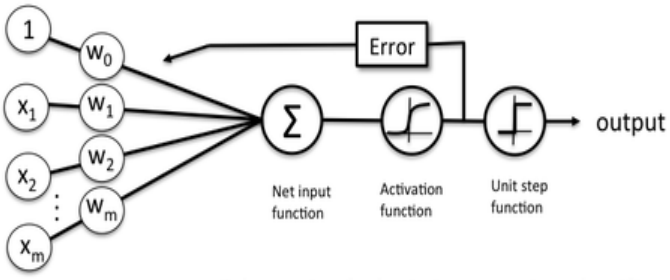
a. Random Forest Classifier

The Random Forest Classifier is a supervised learning algorithm which is an ensemble-type of algorithms, meaning that it consists of multiple different models that work together to find the best prediction. In this case the models that are being used are Decision Trees.



b. Logistic Regression Classifier

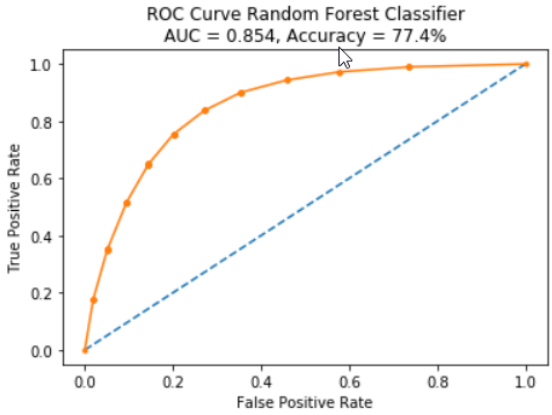
The Logistic Regression Classifier is used to estimate the probability that an instance belongs to a particular class, If the estimated probability is greater than 50%, then the model predicts that the instance belongs to that class (called the positive class, labeled “1”), or else it predicts that it does not (i.e., it belongs to the negative class, labeled “0”).

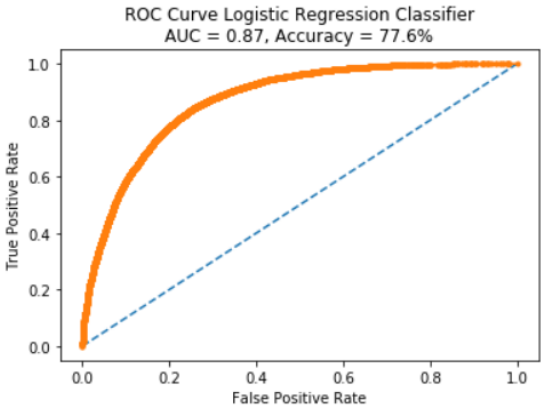


Schematic of a logistic regression classifier.

6. Using GridSearchCV to select best parameters for our models.

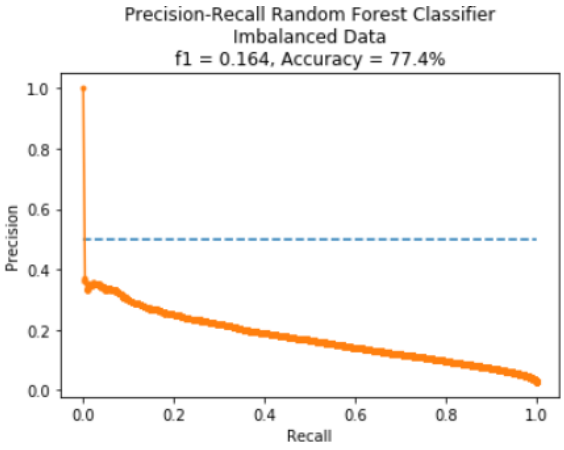
We optimized our models by using Grid Search Cross Validation. We tested different hyper-parameters to see what is the best parameter combination for both Random Forest Classifier and Logistic Regression Classifier. For Random Forest we manipulated parameters such as n_estimators, max_depth, max_leaf_nodes. For Logistic Regression we manipulated parameters such as C (regularization strength), penalty, fit_intercept.

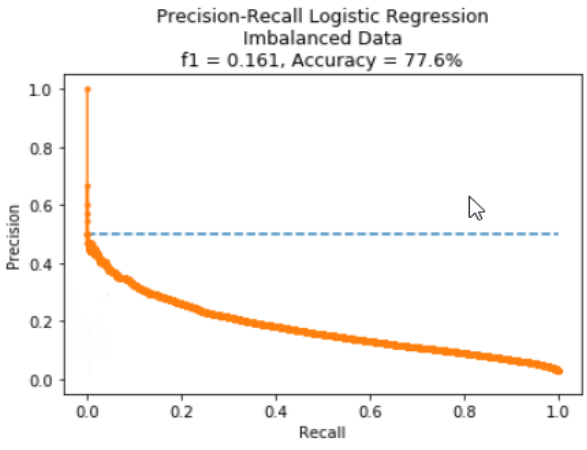




Above are the ROC curves for Random Forest Classifier and Logistic Regression Classifier models based on the balanced dataset. Both models have good test set accuracy, and similar area under the ROC curve. RFC's AUC covers 85.4% of the area with 77.4% accuracy score and LRC's AUC covers 87.0% of the area with the accuracy score of 77.6%. After Undersampling, the performance of the RFC and LRC has improved compared to previous RFC model, where we used an imbalanced dataset.

7. Evaluating model performance on an imbalanced dataset.





As per Precision-Recall curves showed above for the Random Forest Classifier and Logistic Regression Classifier, both models did not perform very well when we reintroduced a highly imbalanced dataset. However, the performance and f1 scores are better than the initial model.

CONCLUSIONS

1. Our Dataset is imperfect. We are missing important features that may influence fatality of an accident, such as, speed, driving under influence etc.
2. After looking at the feature importance of the Random Forest classifier to see what features influence the fatality the most, we noticed that one feature stood out the most – which was a feature that we engineered – Safety Used.
3. Random Forest seems to have a very similar performance to the Logistic Regression on a highly imbalanced and categorical dataset.
4. Possible modeling improvements:
    a. Trying Over-Sampling instead of Under-Sampling the dataset before training. This would create a very large dataset, and training models would take a lot of time.
    b. Trying different algorithms to see if they work better on imbalanced data. We could try algorithms such as Logistic Regression, Neural Networks, and Support Vector Machines.
    c. Trying to use penalized models. Penalized models are trained in such a way that we can vary how big of a penalty the model is going to receive depending on type of misclassification.