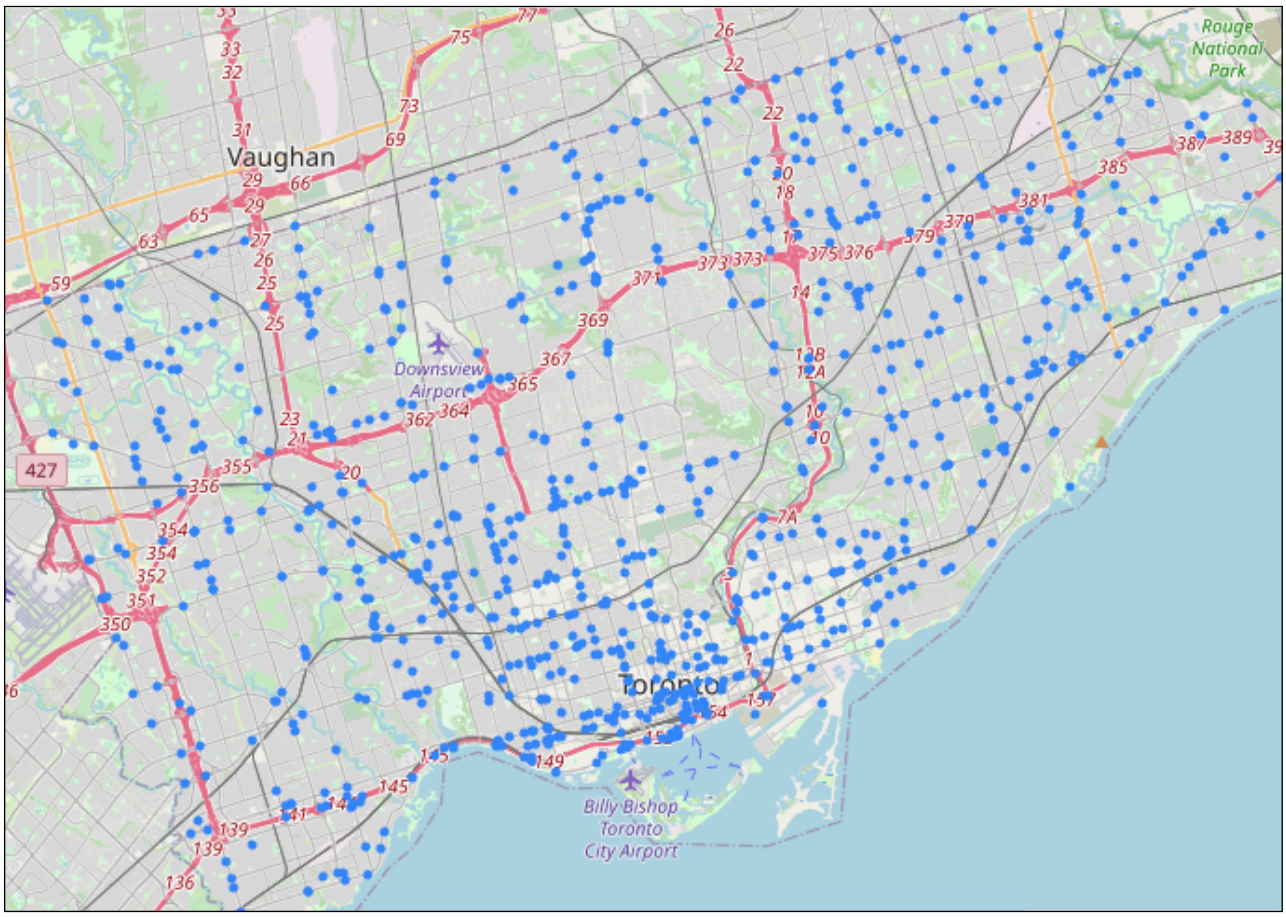
Predicting fatality of a serious accidents in TORONTO

Killed or Seriously Injured Dataset from Toronto Police (2006-2019)



By *Minh Ngoc Pham*

Capstone Project #1

Final Report

Introduction

Toronto's commute has often been considered one of the worst in the world. The city's public transport system faces various criticisms as it has a much smaller subway network system compared to cities with similar size. Most of people in Toronto commute using cars as a result leading to frequent traffics and potential accidents. Toronto Police compiled data from 2006-2019 on Killed or Seriously Injured accidents in the city of Toronto over this time period. This project aims to identify the potential variables leading to a serious/fatal accident in Toronto.

The Dataset

Toronto Police Service
PUBLIC SAFETY DATA PORTAL

The dataset for this project was published publicly by the Toronto Police Service.

The dataset has 14,457 rows including detailed description of 5,690 accidents over the period of 14 years (2006-2019). There are more rows than the number of accidents because each accident is recorded in the number of rows equivalent to the number of people involved (passengers, drivers, cyclists, pedestrians and etc.).

A snapshot of data.head():

	INJURY	ACCNUM	HOUR	ROAD_CLASS	District	LATITUDE	LONGITUDE	LOCCOORD	TRAFFCTL	VISIBILITY	...	SPEEDING	AG_DRIV	REDLIGHT
2006-01-01 02:36:00	Major	893184	2	Major Arterial	Toronto and East York	43.699595	-79.318797	Intersection	No Control	Clear	...	Yes	Yes	No
2006-01-01 02:36:00	Minor	893184	2	Major Arterial	Toronto and East York	43.699595	-79.318797	Intersection	No Control	Clear	...	Yes	Yes	No
2006-01-01 02:36:00	Minor	893184	2	Major Arterial	Toronto and East York	43.699595	-79.318797	Intersection	No Control	Clear	...	Yes	Yes	No
2006-01-01 02:36:00	Minor	893184	2	Major Arterial	Toronto and East York	43.699595	-79.318797	Intersection	No Control	Clear	...	Yes	Yes	No
2006-01-01 02:36:00	Minor	893184	2	Major Arterial	Toronto and East York	43.699595	-79.318797	Intersection	No Control	Clear	...	Yes	Yes	No

5 rows × 45 columns

Independent variable: The independent variable of interest is INJURY - which tells us how injured is the person

Feature variables: The columns of interest include: 'ACCNUM' -accident code, 'ROAD_CLASS', 'LOCCOORD' - where the accident is located, 'TRAFFCTL' - whether there is traffic signal, 'VISIBILITY', 'LIGHT', 'RDSFCOND' - road conditions, 'ACCLASS' - whether it was fatal or non-fatal, 'IMPACTYPE' - nature of impact, 'INVTYPE' - who is involved, 'INVAGE' - age of involved person, 'INJURY' - nature of injury, 'VEHTYPE' -

vehicle, 'MANOEUVER', 'DRIVACT' - driver's action, 'DRIVCOND', 'PEDTYPE', 'PEDACT', 'PEDCOND', 'CYCLISTYPE', 'CYCACT', 'CYCCOND', 'PEDESTRIAN', 'CYCLIST', 'AUTOMOBILE', 'MOTORCYCLE', 'TRUCK', 'EMERG_VEH', 'PASSENGER', 'SPEEDING', 'AG_DRIV' - aggressive driving, 'REDLIGHT', 'ALCOHOL', 'DISABILITY'

All of the columns are categorical.

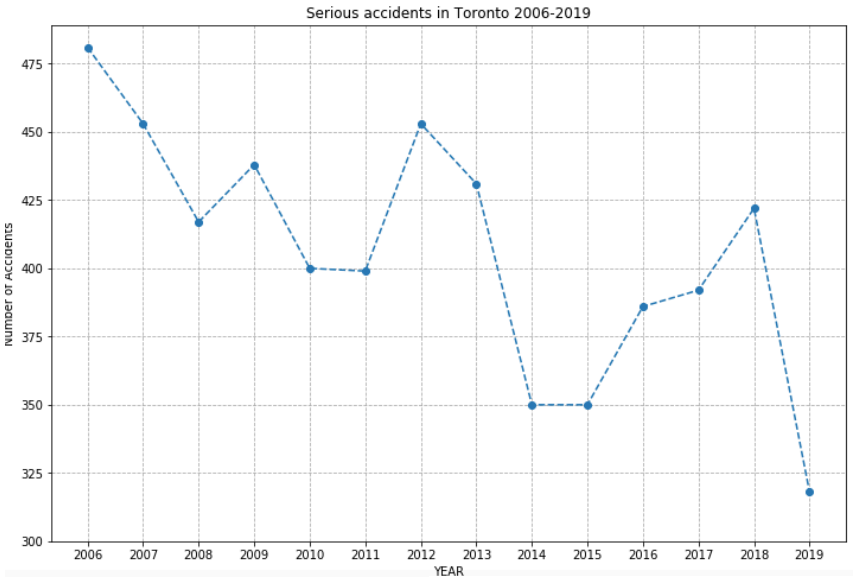
Source: https://data.torontopolice.on.ca/datasets/ksi/data?geometry=-80.371%2C43.550%2C-78.393%2C43.897&page=2

I. Data Wrangling:

- Combine DATE and TIME columns together to create a date time column
- Replace all the empty columns with NaN
- Dropping columns that are not of interest: 'OFFSET', 'Division','ACCLOC', 'Hood_ID', 'FATAL_NO', 'Index_', 'YEAR', 'DATE', 'TIME', 'HOUR', 'X', 'Y', 'WardNum'
- Cleaning and filling District/Ward/Neighbourhood columns
- Replacing all NaN values with 'No' (for columns with Yes/No values) and 'Unknown/Other' (for other categorical columns).

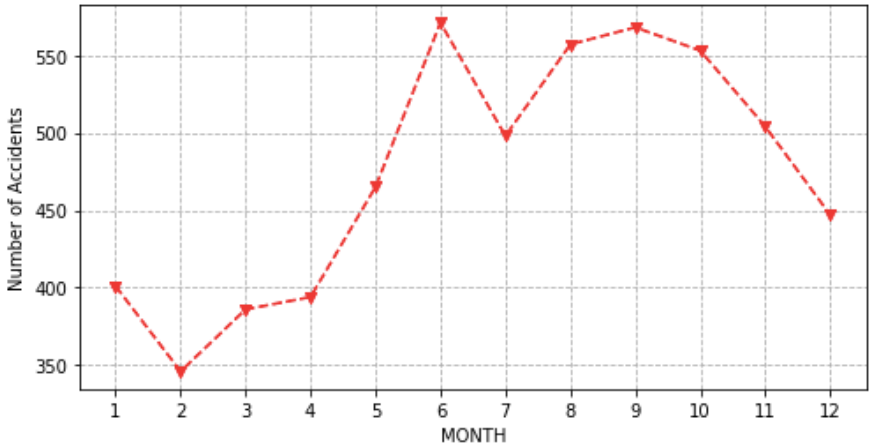
II. Findings from exploratory analysis:

Looking at the trend over time:

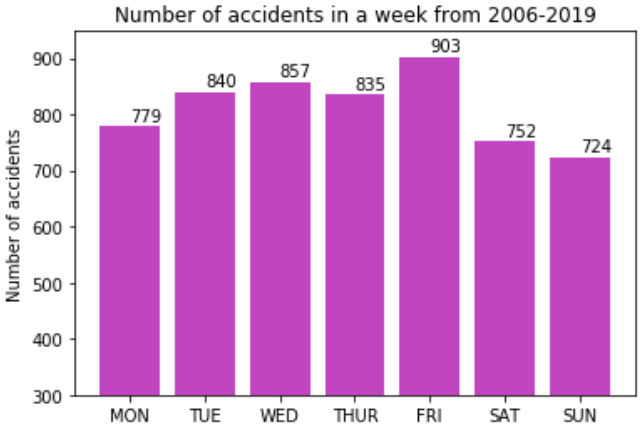


OVER THE YEAR: From the graph above, we can see there is a general decline in the number of serious accidents happening in Toronto in the past 7 years (from 2013 to 2019) compared to from 2006 to 2013. The biggest drops are from the period from 2013 and 2014 and the period from 2018 to 2019 where we can see a big decline in the number of serious accidents.

OVER THE MONTH: The graph shows that there is a very low number of traffic accidents happening from December until April in Toronto, which



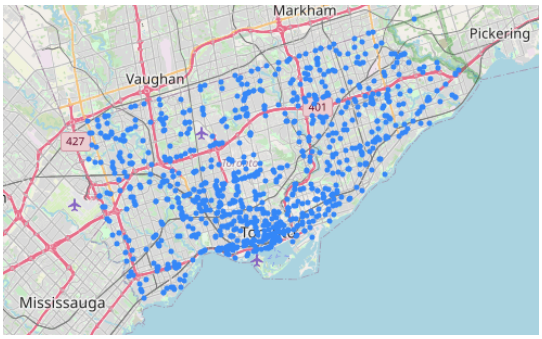
are also winter months in Toronto where the temperature is lowest with a lot of snow days. This seems to correspond to the fact that people stay in more during winter, hence there are less number of accidents.



OVER THE WEEK: From the bar chart above, we can see that there is a higher number of serious accidents on Friday. Assuming that it's the last day of the week so people tend to go out more at night, we can have a look at the data of those accidents happening on Friday only and see what time it usually occurred.

LOCATION

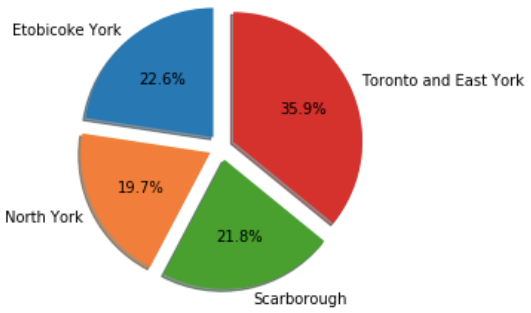
Having a closer look at the accidents happening on Friday (based on coordinates), we notice a large number of accidents happening in the downtown area on Friday, which corresponds to party nights for many people.



By **Neighbourhood**:

Neighbourhood	ACCNUM
Waterfront Communities-The Island (77)	205
West Humber-Clairville (1)	166
Bay Street Corridor (76)	134
Rouge (131)	129
Woburn (137)	116

The highest number of accidents happening in Toronto and East York seems to coincide with the neighbourhood of Waterfront Communities-The Island (77) as this neighbourhood is located in this district.

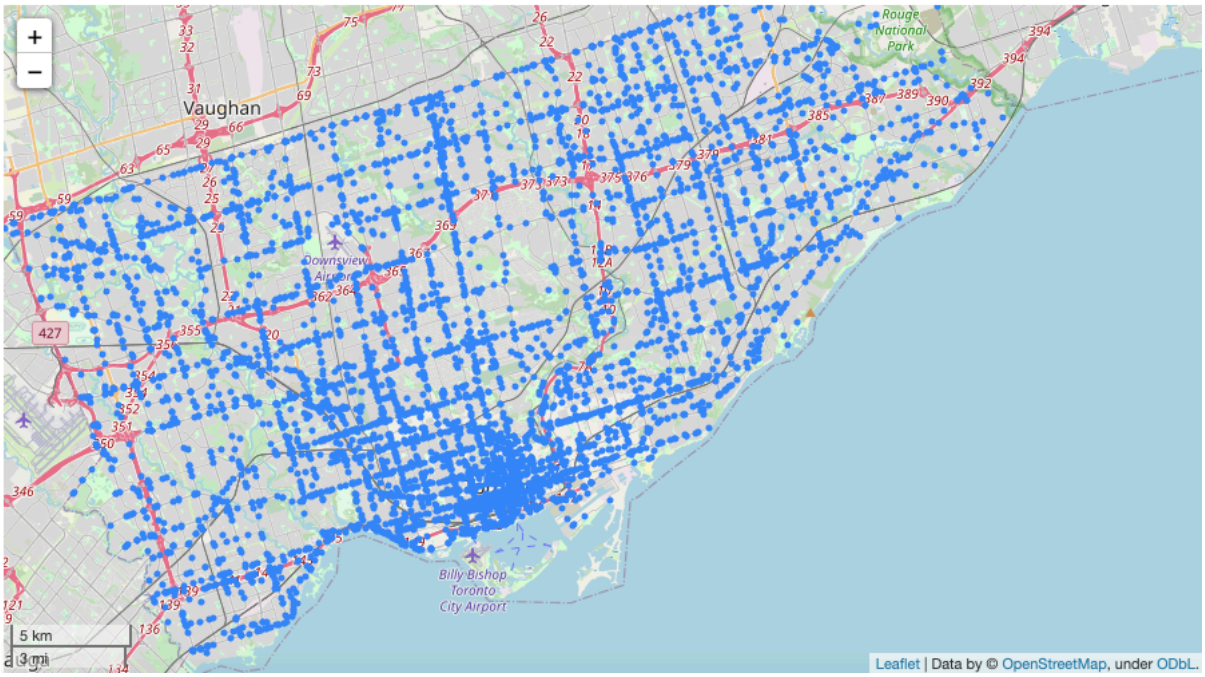


The map above shows that our finding is quite accurate in terms of the location of the accidents, a lot of them occurred in the waterfront area of Toronto (strong cluster of accidents over the year).

Looking at the ***road class*** of the accident location:

ROAD_CLASS	Total accidents	Fatal	Rate of Fatal Accidents
Collector	354	49.0	0.138418
Expressway	6	1.0	0.166667
Laneway	4	3.0	0.750000
Local	289	44.0	0.152249
Major Arterial	3981	538.0	0.135142
Major Arterial Ramp	1	0.0	0.000000
Minor Arterial	931	107.0	0.114930
Unknown/Other	124	24.0	0.193548

Without looking at the rate of Fatal accidents for Unknown and Laneway, we can see that accidents happening on Expressway tend to be more fatal (0.166666) than major arterial (0.135) even though there are more accident happening on major arterial roads. A possible explanation for this could be the fact that people tend to drive faster on Expressway making it more fatal when collision occurs.



Map of all the serious accidents occurring in Toronto over the period from 2006-2019

Looking at the location with ***high frequency of accidents*** (those intersections with more than 4 accidents over the years):

We can see from the table here that the almost half (205 out of 451) of the location with high frequency of accidents are with pedestrian collisions.

IMPACTYPE	ACCNUM
Angle	41
Approaching	10
Cyclist Collisions	28
Other	7
Pedestrian Collisions	205
Rear End	32
SMV Other	37
SMV Unattended Vehicle	1
Sideswipe	7
Turning Movement	82

INJURY INVTYPE	Fatal	Non-Fatal
Cyclist	37.0	632.0
Cyclist Passenger	0.0	2.0
Driver	145.0	2578.0
Driver - Not Hit	0.0	1.0
In-Line Skater	0.0	4.0
Moped Driver	0.0	26.0
Motorcycle Driver	65.0	467.0
Motorcycle Passenger	1.0	28.0
Passenger	87.0	1581.0
Pedestrian	441.0	2251.0
Truck Driver	3.0	49.0
Wheelchair	2.0	11.0

```
fatal=compare1['Fatal']
non_fatal=compare1['Non-Fatal']
table = pd.crosstab(fatal, non_fatal)
from scipy.stats import chi2_contingency
chi2, p, dof, expected = chi2_contingency(table.values)
print (chi2, p)
```

```
95.99999999999999 0.262528152279769
```

whether their injury will be fatal or not.

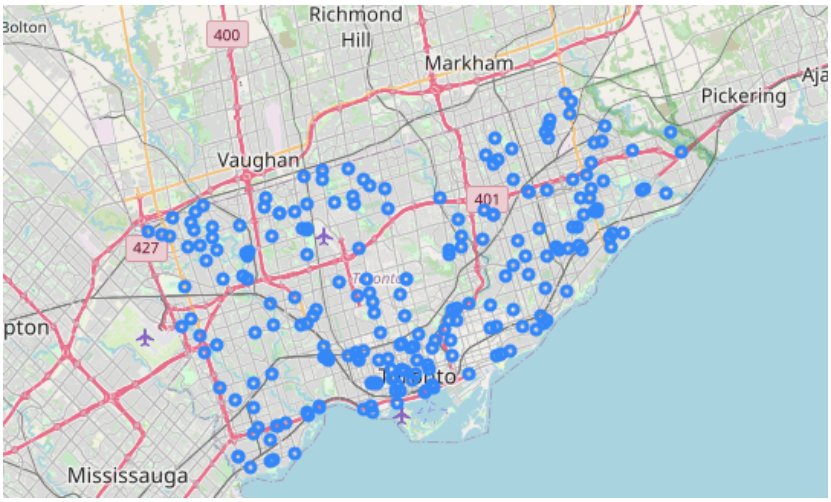
III.Chi-square test for two variables

Most of the variables used in the analysis are categorical variables so we will use Chi-square to determine if there is a relationship between two variables.
We will carry out chi-square test for a couple of variable-pairs to determine if there is some correlation between them (they are not independent if p value from the chi-square test is low)
•INVOLVED PERSON AND INJURY: low P-value, that means who the involved person is does not necessarily determined

•Is there a relationship

ALCOHOL Neighbourhood	No	Yes
Agincourt North (129)	145.0	5.0
Agincourt South-Malvern West (128)	142.0	0.0
Alderwood (20)	57.0	7.0
Annex (95)	185.0	9.0
Banbury-Don Mills (42)	143.0	19.0
...	...	...
Wychwood (94)	92.0	0.0
Yonge-Eglinton (100)	45.0	4.0
Yonge-St.Clair (97)	46.0	0.0
York University Heights (27)	205.0	9.0
Yorkdale-Glen Park (31)	102.0	3.0

140 rows × 2 columns



between

```
not_drunk=compare3['No']
drunk=compare3['Yes']
table3 = pd.crosstab(not_drunk, drunk)
from scipy.stats import chi2_contingency
chi2_3, p_3, dof_3, expected_3 = chi2_contingency(table3.values)
print (chi2_3, p_3)
```

```
2328.8207547169814 9.225194703874935e-05
```

Neighbourhood and whether alcohol is involved?

The P-value is actually very low here, suggesting a relationship between location of the accident and whether there is alcohol involved.
Further look into the relationship shows that most of the alcohol-involved accidents occur in about 87 out of 140 neighbourhoods.

IV. Machine learning process:

The KSI dataset consists mostly of categorical data, and hence we will use the method of supervised learning to make prediction on whether a certain person will survive or not from an accident (each row represents a single person involved in the accident).
The independent variable in question is 'INJURY', which we will classify as Fatal/Major or Minimal as impact in a person from an accident.

1. Preprocessing:

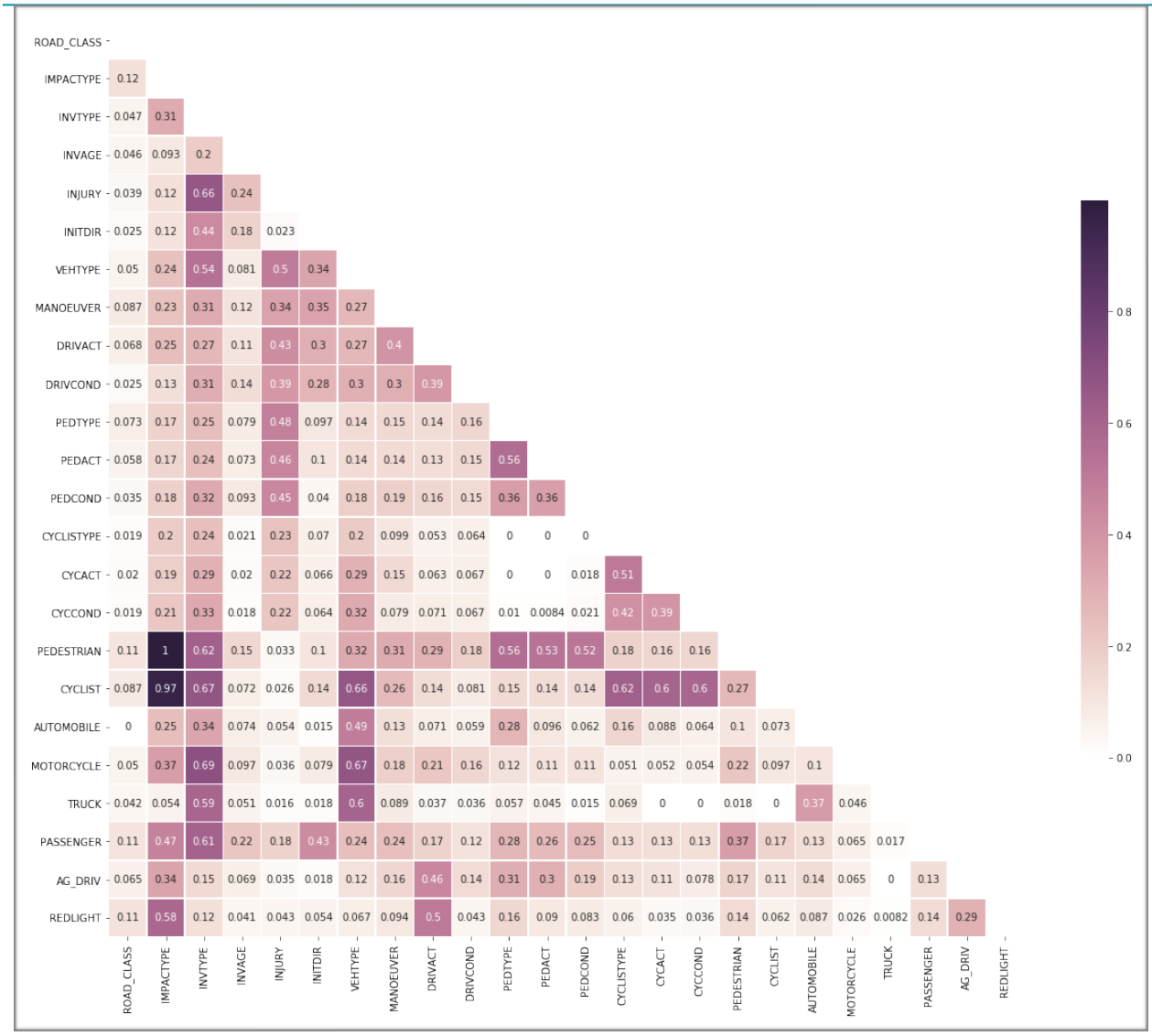
- Dropping unnecessary columns and engineer the date columns to get hour and day of the week
- Our dataset looks quite balanced in terms of the two classes: Minimal and Fatal/Major with slightly more number of people falling into the Minimal Injury (57.4% versus 42.6%)
- The majority of our data includes categorical variables so we will perform Chi-square to determine independence with independent variables (in this case 'INJURY')
- From the Chi-square we can see the following variables: HOUR, District, LOCCOORD, TRAFFCTL, VISIBILITY, LIGHT, RDSFCOND, EMERG_VEH, SPEEDING, ALCOHOL, DISABILITY, Hood_ID, TRSN_CITY_, WEEKDAY show high P-value corresponding to a higher dependency to the target variable 'INJURY'.
- Another way to determine relationship between data is by looking at the **Cramer's V**. Chi-square test can be sensitive to sample size (here our sample size is big) and Cramer's V actually takes in consideration of the sample size. Cramer's V is a measure of association between features variables. The value is between 0 and 1, with a number closer to 1 denoting a strong association. We will have a look at the correlation between feature variables.

Cramer's V's formula:

$$V = \sqrt{\frac{(\chi^2/n)}{min(c-1, r-1)}}$$

where r corresponds to the number of rows, and c corresponds to the number of columns

- We notice that dummy variables 'PEDESTRIAN', 'CYCLIST', 'PASSENGER', 'TRUCK' also show up in INVTYPE and that explains the high correlation. We can drop these variables because later one when we do OneHotEncoding, we will have duplicate variables representing the same thing.



- There is strong correlation between Hood_ID and District as different neighbourhoods are located in certain District. We will drop District and keep Hood_ID
- ***Converting data into numerical values:***

Machine learning requires that the input and the output variables being numerical values. As a result, the first step in getting the data ready is by making sure all the variables are in numerical form.

```
from sklearn.preprocessing import OneHotEncoder, LabelBinarizer
# Transform all the X's categorical values into numerical values

onehotencoder = OneHotEncoder()
X_coded = onehotencoder.fit_transform(X)
X_number=X_coded.toarray()
# y= column 'INJURY' is of binary value and hence we will convert it into binary label
lb = LabelBinarizer()
y_number = lb.fit_transform(y)

X_number
```

```
array([[0., 0., 1., ..., 0., 0., 1.],
       [0., 0., 1., ..., 0., 0., 1.],
       [0., 0., 1., ..., 0., 0., 1.],
       ...,
       [0., 0., 0., ..., 0., 1., 0.],
       [0., 0., 0., ..., 0., 1., 0.],
       [0., 0., 0., ..., 0., 1., 0.]])
```

For the X value, we will use the method called OneHotEncoder to transform the data into different columns since all the data we have are unordered categorical variables, we do not want our model to run as if the numbered ordering matters. OneHotEncoder will split data in the number of columns corresponding to the number of options for each categorical data.
For the Y value, we use LabelBinarizer to encode the result into 0 and 1.
A snapshot of the feature names after conversion:

```
feature_names = onehotencoder.get_feature_names(input_features=X.columns)
feature_names
```

```
Out[12]: array(['HOUR_0', 'HOUR_1', 'HOUR_2', 'HOUR_3', 'HOUR_4', 'HOUR_5',
                'HOUR_6', 'HOUR_7', 'HOUR_8', 'HOUR_9', 'HOUR_10', 'HOUR_11',
                'HOUR_12', 'HOUR_13', 'HOUR_14', 'HOUR_15', 'HOUR_16', 'HOUR_17',
                'HOUR_18', 'HOUR_19', 'HOUR_20', 'HOUR_21', 'HOUR_22', 'HOUR_23',
                'ROAD_CLASS_Collector', 'ROAD_CLASS_Expressway',
                'ROAD_CLASS_Laneway', 'ROAD_CLASS_Local',
                'ROAD_CLASS_Major Arterial', 'ROAD_CLASS_Major Arterial Ramp',
                'ROAD_CLASS_Minor Arterial', 'ROAD_CLASS_Unknown/Other',
                'LOCCOORD_Entrance Ramp Westbound',
                'LOCCOORD_Exit Ramp Southbound', 'LOCCOORD_Exit Ramp Westbound',
                'LOCCOORD_Intersection', 'LOCCOORD_Mid-Block',
                'LOCCOORD_Mid-Block (Abnormal)',
```

After encoding the data, we will have a total of 436 features, corresponding to the number of columns in the dataset times the number of possible values for each column.

Machine Learning Modelling

I. Decision Tree Classifier

The classification problem is a binary problem and hence we will be performing DecisionTreeClassifier.

Using KFold

```
from sklearn.model_selection import cross_validate, KFold, ShuffleSplit, train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
# Define our data splitting
split = KFold(n_splits=5, shuffle=True, random_state=0)
treemodel = DecisionTreeClassifier(max_depth=15)
treemodel.fit(X_number, y_number.ravel())
scores = cross_validate(treemodel, X_number, y_number.ravel(), cv=split)
print("Test score:                 {}".format(scores["test_score"]))
# Print average across K tests
print("Average test score:       any   {} (+/- {})".format(scores["test_score"].mean(), scores["test_score"].std() * 2)
```

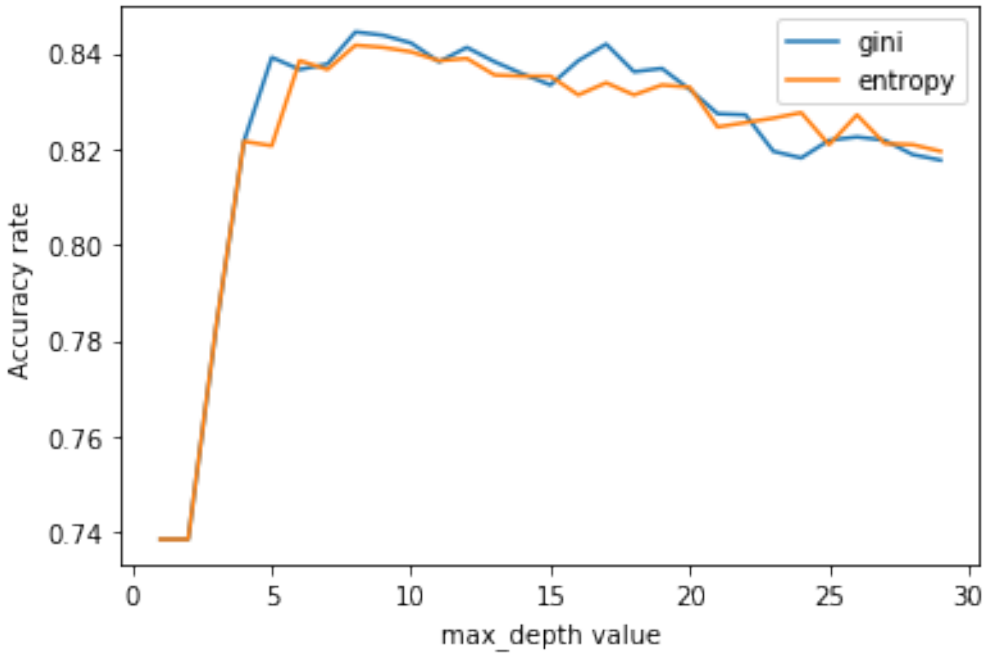
```
Test score:                 [0.83229599 0.82710927 0.84192321 0.8377724  0.84053961]
Average test score:       any   0.8359280937214889 (+/- 0.011011732182319281)
```

Using train_test_split

```
#Splitting the data into Training and Test data
X_train, X_test, y_train, y_test = train_test_split(X_number, y_number.ravel(), test_size = 0.30, random_state = 101)
tree2 = DecisionTreeClassifier(max_depth=15, criterion = 'gini')
tree2.fit(X_train, y_train)
from sklearn.metrics import classification_report, confusion_matrix, roc_curve,auc, accuracy_score
tree_predictions = tree2.predict(X_test)
print(accuracy_score(y_test, tree_predictions))
```

```
0.8358690640848317
```

Notice the similar result between KFold and train_test_split (accuracy score of around 0.835), for a big dataset like this, it might be more computational appropriate to go with train_test_split.



•Parameter tuning with max_depth and criterion

We will check the model accuracy using different max_depth level from 1-30 and compare that between criterion as 'entropy' or ' gini'.

The graph above shows that max_depth value of around 9 or 10 gives us the best accuracy rate for 'gini' Decision Tree Classifier. The model is as followed:

```
#Splitting the data into Training and Test data
finaltree = DecisionTreeClassifier(max_depth=10, criterion = 'gini')
finaltree.fit(X_train, y_train)
finaltree_predictions = finaltree.predict(X_test)
print(accuracy_score(y_test, finaltree_predictions))
```

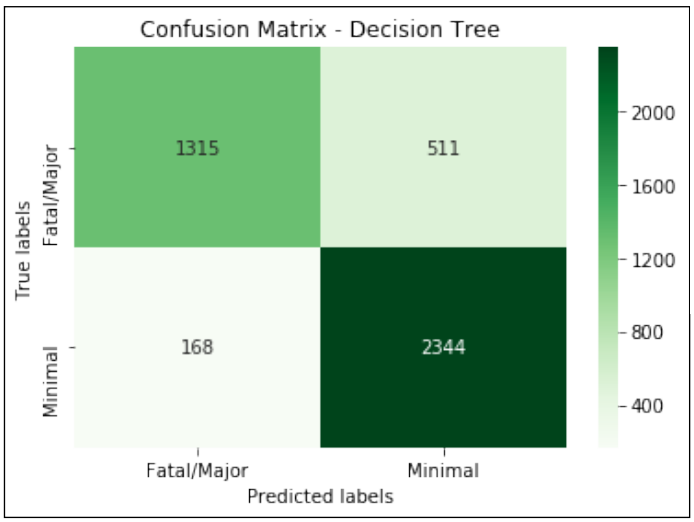
```
0.8416320885200553
```

We will then scale the model using StandardScaler and Pipeline:

```
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
std_slc = StandardScaler()
pipe = Pipeline(steps=[('std_slc', std_slc),
                       ('decision tree', DecisionTreeClassifier(max_depth=10))])
final_scaled = pipe.fit(X_train, y_train)
final_scaled_pred = final_scaled.predict(X_test)
print(accuracy_score(y_test, final_scaled_pred))
```

```
0.8434762563393269
```

Tuning the tree model using pipeline slightly improve the model's accuracy. Our final Decision Tree model gives the following result:
The decision tree modelling gave us a pretty good result of 89% precision for prediction of Fatal/Major impact and 82% accuracy of predicting a person suffering from Minimal impact due to an accident.



	precision	recall	f1-score
0	0.89	0.72	0.79
1	0.82	0.93	0.87
accuracy			0.84
macro avg	0.85	0.83	0.83
weighted avg	0.85	0.84	0.84

The confusion matrix above also show us a better prediction of Minimal impact of (2344 times accurate and 511 inaccurate prediction) while with Fatal/Major impact the model is accurate 1315 times and the number of inaccurate prediction is low at 168 times.

II. Logistic Regression

```
#Creating the prediction model
from sklearn.linear_model import LogisticRegression
log_model = LogisticRegression(solver='lbfgs')
log_model.fit(X_train, y_train)
#Performance Check
predictions = log_model.predict(X_test)
print(accuracy_score(y_test, predictions))
```

```
0.8497003227293684
```

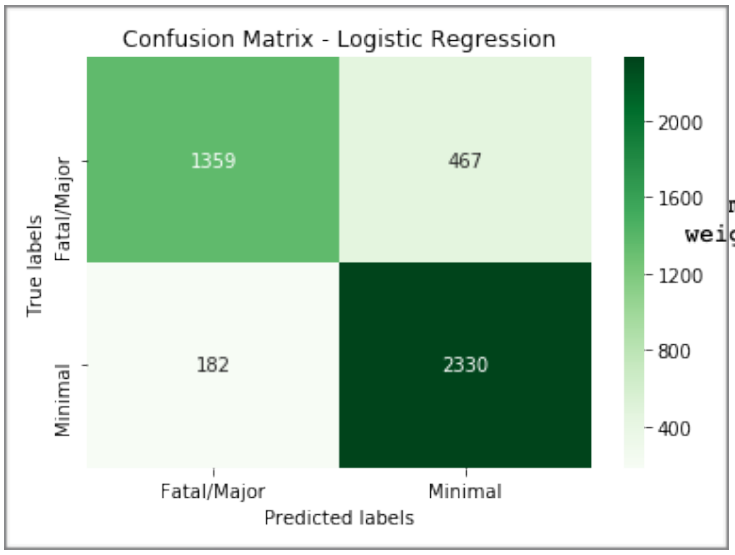
Logistic regression seems to have a better accuracy score compared with decision tree. We will use grid search to Searching for best performing hyper-parameters for the models

The result of the grid search is : Based on the search above, we can see that the best performing logistic regression model is the one with C=0.1 and solver = 'liblinear' We then have our final Logistic Regression model as:

```
Best: 0.843331 using {'C': 0.1, 'solver': 'liblinear'}
0.841256 (0.009537) with: {'C': 100, 'solver': 'newton-cg'}
0.841025 (0.009995) with: {'C': 100, 'solver': 'lbfgs'}
0.841256 (0.009537) with: {'C': 100, 'solver': 'liblinear'}
0.842376 (0.010090) with: {'C': 10, 'solver': 'newton-cg'}
0.842145 (0.010011) with: {'C': 10, 'solver': 'lbfgs'}
0.842376 (0.010103) with: {'C': 10, 'solver': 'liblinear'}
0.843067 (0.010044) with: {'C': 1.0, 'solver': 'newton-cg'}
0.843100 (0.010095) with: {'C': 1.0, 'solver': 'lbfgs'}
0.843067 (0.010092) with: {'C': 1.0, 'solver': 'liblinear'}
0.843298 (0.009397) with: {'C': 0.1, 'solver': 'newton-cg'}
0.843232 (0.009365) with: {'C': 0.1, 'solver': 'lbfgs'}
0.843331 (0.009623) with: {'C': 0.1, 'solver': 'liblinear'}
0.834700 (0.008576) with: {'C': 0.01, 'solver': 'newton-cg'}
0.834700 (0.008576) with: {'C': 0.01, 'solver': 'lbfgs'}
0.834799 (0.008522) with: {'C': 0.01, 'solver': 'liblinear'}
```

```
final_log = LogisticRegression(solver='liblinear', C=0.1)
final_log.fit(X_train, y_train)
log_predictions = final_log.predict(X_test)
print(accuracy_score(y_test, log_predictions))
```

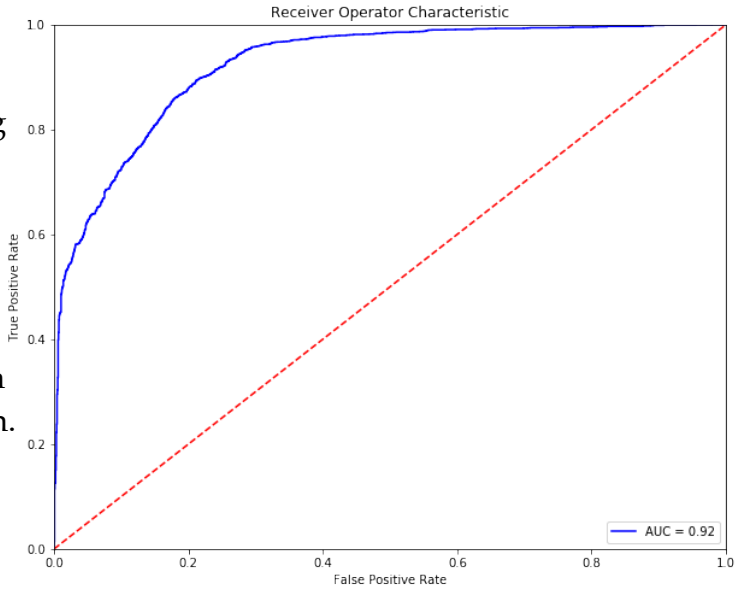
```
0.8503918856615952
```



	precision	recall	f1-score
0	0.88	0.74	0.81
1	0.83	0.93	0.88
accuracy			0.85
macro avg	0.86	0.84	0.84
weighted avg	0.85	0.85	0.85

Interestingly enough, by looking at the confusion matrix here for Logistic Regression and comparing it to the confusion matrix for Decision Tree, we can see that they both have similar true prediction for Fatal/Major accident but logistic regression is slightly better in predicting which person suffers from Minimal injury.

The AUC is very high here (0.92). Combined with the confusion matrix and classification report, logistic regression seems to be doing better than Decision Tree in modelling this situation.



III. Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier

forest=RandomForestClassifier(n_estimators = 100, max_depth=15)
forest.fit(X_train, y_train)

# Predict and view stats
forest_prediction = forest.predict(X_test)
print(accuracy_score(y_test, forest_prediction))
```

```
0.8432457353619179
```

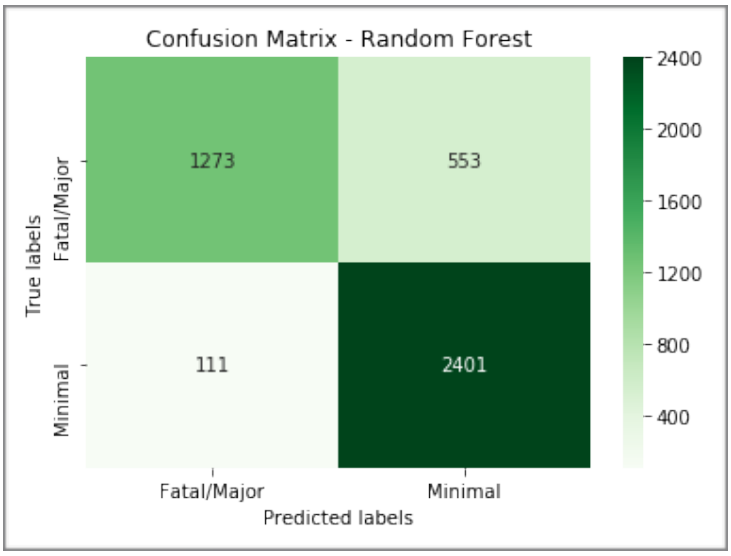
Using GridSearch to find the best parameters for Random Forest

```
Best: 0.841091 using {'max_depth': 20, 'max_features': 'sqrt', 'n_estimators': 1000}
0.827058 (0.000955) with: {'max_depth': 10, 'max_features': 'sqrt', 'n_estimators': 10}
0.828639 (0.003744) with: {'max_depth': 10, 'max_features': 'sqrt', 'n_estimators': 100}
0.828540 (0.003039) with: {'max_depth': 10, 'max_features': 'sqrt', 'n_estimators': 1000}
0.820437 (0.003107) with: {'max_depth': 10, 'max_features': 'log2', 'n_estimators': 10}
0.816780 (0.001436) with: {'max_depth': 10, 'max_features': 'log2', 'n_estimators': 100}
0.815298 (0.001430) with: {'max_depth': 10, 'max_features': 'log2', 'n_estimators': 1000}
0.834667 (0.002451) with: {'max_depth': 15, 'max_features': 'sqrt', 'n_estimators': 10}
0.837435 (0.001387) with: {'max_depth': 15, 'max_features': 'sqrt', 'n_estimators': 100}
0.837237 (0.002341) with: {'max_depth': 15, 'max_features': 'sqrt', 'n_estimators': 1000}
0.823698 (0.001828) with: {'max_depth': 15, 'max_features': 'log2', 'n_estimators': 10}
0.822710 (0.002479) with: {'max_depth': 15, 'max_features': 'log2', 'n_estimators': 100}
0.823006 (0.001915) with: {'max_depth': 15, 'max_features': 'log2', 'n_estimators': 1000}
0.837138 (0.001942) with: {'max_depth': 20, 'max_features': 'sqrt', 'n_estimators': 10}
0.839905 (0.001087) with: {'max_depth': 20, 'max_features': 'sqrt', 'n_estimators': 100}
0.841091 (0.002820) with: {'max_depth': 20, 'max_features': 'sqrt', 'n_estimators': 1000}
0.824686 (0.002547) with: {'max_depth': 20, 'max_features': 'log2', 'n_estimators': 10}
0.830122 (0.001894) with: {'max_depth': 20, 'max_features': 'log2', 'n_estimators': 100}
0.829430 (0.003032) with: {'max_depth': 20, 'max_features': 'log2', 'n_estimators': 1000}
```

Based on the Gridsearch above, the best combination for the parameter is max_depth= 20, max_features = sqrt and n_estimators = 1000.

```
final_forest=RandomForestClassifier(max_depth= 20, max_features = 'sqrt' ,n_estimators = 1000)
final_forest.fit(X_train, y_train)

# Predict and view stats
final_forest_pred = final_forest.predict(X_test)
print(accuracy_score(y_test, final_forest_pred))
```

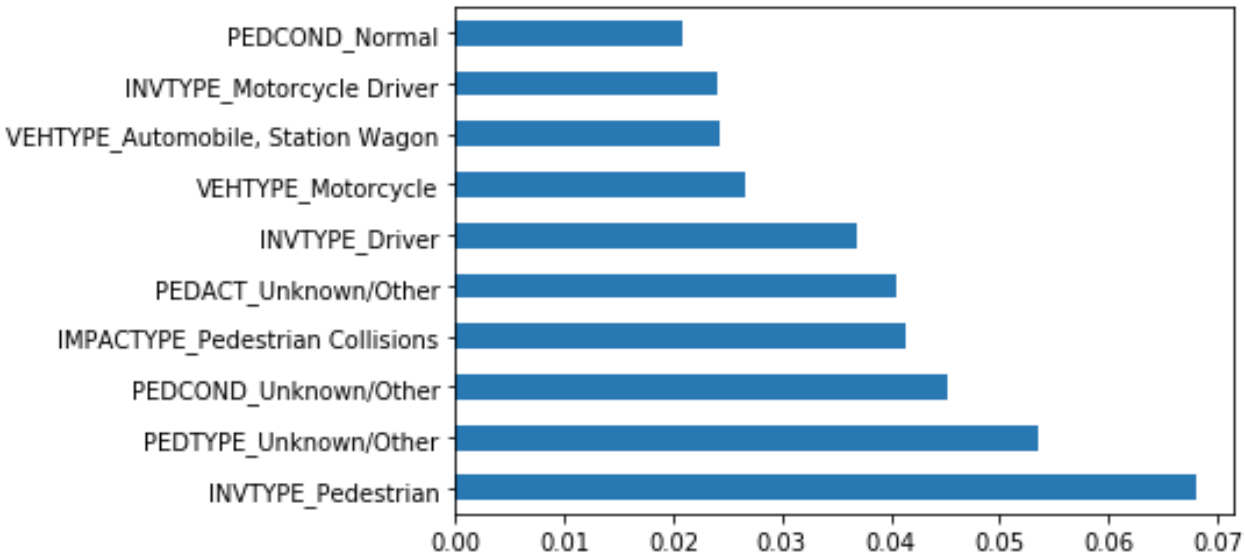
```
0.846934071000461
```



Comparing with the other models, Random Forest Classifier seems to be doing better at prediction those suffering from Minimal impact but seems to make a lot of wrong prediction on those who might suffer from Fatal/Major accident.
Final result from Random Forest:

	precision	recall	f1-score
0	0.92	0.70	0.79
1	0.81	0.96	0.88
accuracy			0.85
macro avg	0.87	0.83	0.84
weighted avg	0.86	0.85	0.84

Feature Importance in Random Forest Classifier:



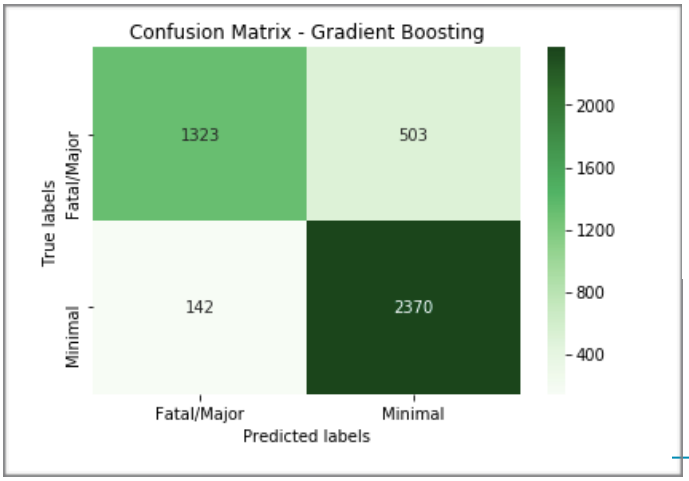
Based on the above we can see that some features are more important in others in predicting the severity of an accident in the Random Forest Modelling. The top 10 indicates that INVTYPE (involvement type, whether you are a pedestrian, driver or motorcycle driver), VEHTYPE (the type of vehicle) and PEDCOND (pedestrian conditions) are important features in the random forest model.

IV. Gradient Boosting Classifier

We will use Gridsearch to find the best learning_rate and n_estimators for our Gradient Boosting Classifier

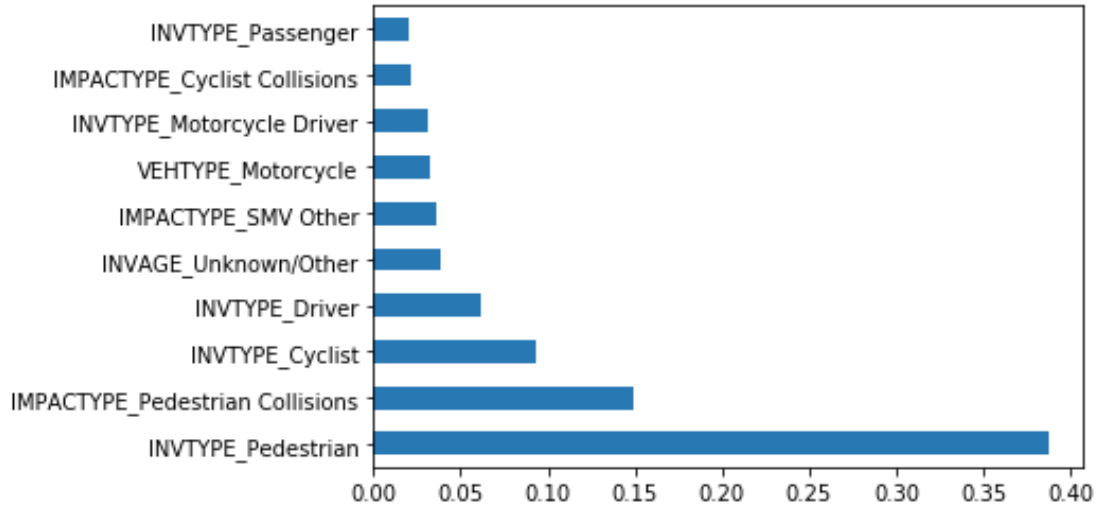
```
final_gradient=GradientBoostingClassifier(random_state=0, learning_rate = 0.01, n_estimators=1000)
final_gradient.fit(X_train, y_train)

# Predict and view stats
final_grad_pred = final_gradient.predict(X_test)
print(accuracy_score(y_test, final_grad_pred))
```

```
0.851313969571231
```



	precision	recall	f1-score
0	0.90	0.72	0.80
1	0.82	0.94	0.88
accuracy			0.85
macro avg	0.86	0.83	0.84
weighted avg	0.86	0.85	0.85

The matrix above shows us that Gradient Boosting seems to also be very good at predicting Minimal impact and the accuracy rate is as high if not slightly higher than logistic regression (at 85%).
Feature importance in Gradient Boosting Classifier:

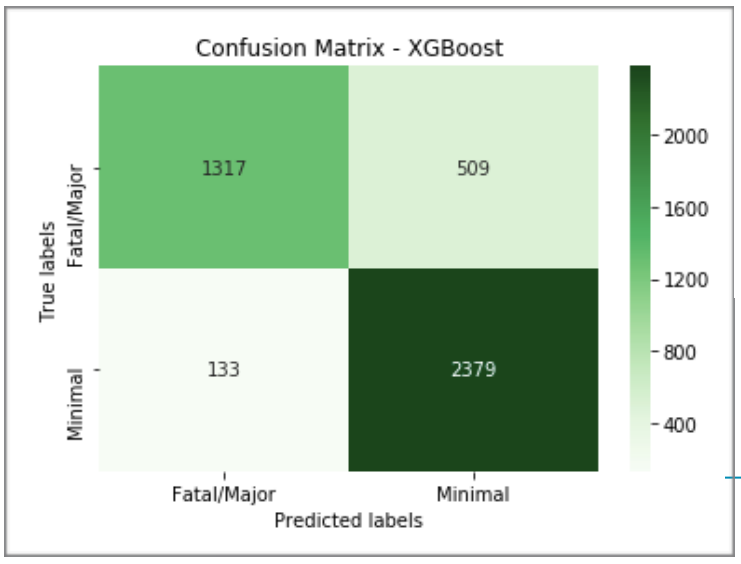


Similarly to the random forest classification model, in gradient boosting, INVTYPE is important in predicting the outcome of an accident but IMPACTYPE is also considered one of the most important features.

V. XGB Classifier

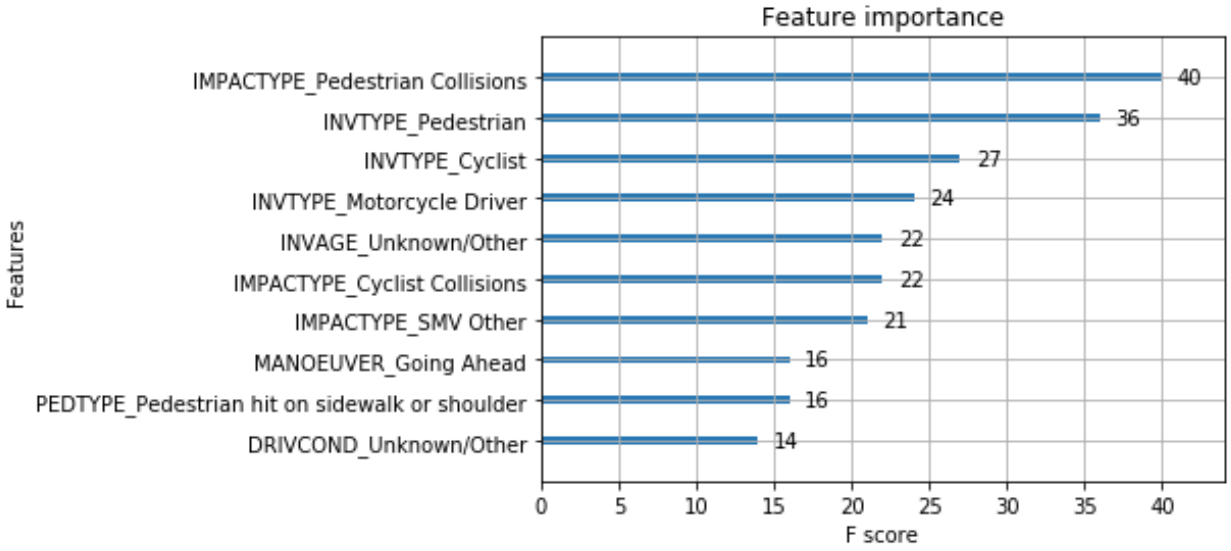
```
from xgboost import XGBClassifier
# fit model
xgboost = XGBClassifier()
xgboost.fit(X_train, y_train)
xgboost_prediction = xgboost.predict(X_test)
print(xgboost)
print(accuracy_score(y_test, xgboost_prediction))
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=3,
              min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
0.8520055325034578
```



	precision	recall	f1-score
0	0.91	0.72	0.80
1	0.82	0.95	0.88
accuracy			0.85
macro avg	0.87	0.83	0.84
weighted avg	0.86	0.85	0.85

Feature importance in XGBClassifier:



Ranking of Feature importance in different models:

Model Name	1st rank	2nd rank	3rd rank	4th rank	5th rank
Random Forest	INVTYPE_ Pedestrian	PEDTYPE_ Unknown	PEDACT_ Unknown	PEDCOND_ Unknown	INVTYPE_ Driver
Gradient	INVTYPE_ Pedestrian	IMPACTYPE_ Pedestrian Collision	INVTYPE_ Cyclist	INVTYPE_ Driver	INVAGE_ Unknown
XGB	IMPACTYPE_ Pedestrian Collision	INVTYPE_ Pedestrian	INVTYPE_ Cyclist	INVTYPE_ Motorcycle Driver	INVAGE_ Unknown

By looking at the table above ranking the feature importance in the 3 models, we can see that a lot of the model rank Pedestrian related features as important features in determining the fatality of an accident.

CONCLUSION:

Based on all the results from the modelling, we find that most of the modelling seem to have similar accuracy rate (from 84%-86%). We can hence conclude the followings:

- The feature variables of each person involved in the accident are good predictor of whether this person will suffer from a major injury or not.
- Better policy can be implemented based on the findings such as: (1) ALCOHOL use can determine whether a person will suffer from major or final injury and hence should be properly controlled by the authority. (2) Certain locations tend to attract accident (waterfront or downtown area) during certain day of the week (Friday with the highest number of accidents) and therefore this areas should be more carefully monitored to avoid serious accidents. (3) Certain road classed such as Major Arterial and Collector attracts a high number of serious accidents as well as high rate of fatal accidents, it is therefore essential to focus on having officers monitoring these areas more effectively.