Springboard Data Science Project - By Murali Sankaran

(Mentor: Ryan Rosario)



United Kingdom – Fatal Accidents Prediction

Using LogiSTIC Regression

**Table of Contents**

[1 Introduction 3](#_Toc472354623)

[2 Dataset 3](#_Toc472354624)

[3 Data Facts 4](#_Toc472354625)

[3.1 Data Volume 4](#_Toc472354626)

[3.2 Data Dictionary 4](#_Toc472354627)

[4 Exploratory Data Analysis 6](#_Toc472354628)

[4.1 Accidents over the Days of the Week 6](#_Toc472354629)

[4.2 Accident Severity 7](#_Toc472354630)

[4.3 Number of Vehicles involved in the Accidents 8](#_Toc472354631)

[4.4 Number of Casualties 9](#_Toc472354632)

[4.5 Speed Limit 10](#_Toc472354633)

[4.6 Light Conditions 11](#_Toc472354634)

[4.7 Weather Conditions 12](#_Toc472354635)

[4.8 Urban or Rural Area 13](#_Toc472354636)

[5 Fatal Accidents across the United Kingdom 14](#_Toc472354637)

[6 Logistic Regression 15](#_Toc472354638)

[6.1 Building the Prediction Model – Choosing the Attributes 15](#_Toc472354639)

[6.2 Performing the Regression 16](#_Toc472354640)

[6.3 Interpreting the results 17](#_Toc472354641)

[6.4 Odds ratio 18](#_Toc472354642)

[6.5 Area Under the Curve (AUC) 19](#_Toc472354643)

[7 Conclusion 20](#_Toc472354644)

[7.1 Analysis 20](#_Toc472354645)

[7.2 Client Recommendation 20](#_Toc472354646)

[7.3 Limitations 21](#_Toc472354647)

[8 References 22](#_Toc472354648)

[9 Python Code used in the project 23](#_Toc472354649)

[9.1 Initialize libraries, get datasets, consolidate and cleanse 23](#_Toc472354650)

[9.2 Getting a feeling of the dataset 23](#_Toc472354651)

[9.3 Perform operations to get data distribution and NULL/NAN values 24](#_Toc472354652)

[9.4 Exploratory Data Analysis 27](#_Toc472354653)

[9.5 Plotting all Fatal Accidents location on a map using Latitude and Longitude 29](#_Toc472354654)

[9.6 Perform Logistic Regression using StatsModels 30](#_Toc472354655)

[9.7 ROC Curve 32](#_Toc472354656)

# Introduction

In Great Britain, the Road Accident Data are collected by the Department for Transport and comprise road accident statistics collected from information about personal injury road accidents and their consequent casualties reported, to a common national standard.

The aim of collecting and publishing national road accident statistics is to inform public debate and to provide the basis for determining and monitoring effective road safety policies to reduce the road accident casualty toll. The UK Data Archive holds annual Road Accident Data from 1985 onwards. A similar series, the Northern Ireland Police Recorded Injury Road Traffic Collision Data, are available from 2004 onwards.

Most of the statistics are based on road accidents reported to the police (STATS19 system). These provide detailed statistics about the circumstances of personal injury road accidents, including the types of vehicles involved and the consequent casualties.

Other sources directly related to road safety are also used, including hospital admissions, death registrations, coroner’s reports and national travel survey, crime survey from England and Wales and statistics on breath tests and motoring offences from the Home Office and Ministry of Justice.

# Dataset

The dataset files provide detailed data about the circumstances of personal injury road accidents in Great Britain from 2005 onwards, the types of vehicles involved and the consequential casualties. The statistics relate only to personal injury accidents on public roads that are reported to the police, and subsequently recorded, using the STATS19 accident reporting form. Information on damage-only accidents, with no human casualties or accidents on private roads or car parks are not included in this data.

The data for analysis (2005 to 2014) will be used from

 https://data.gov.uk/dataset/road-accidents-safety-data.

# Data Facts

## Data Volume

The data file has 1,640,597 observations and has 17 variables.

## Data Dictionary

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Dataset Column** | **Definition** | | |
| Accident Index | Unique identification number of the incident | | |
| Location (East) | Eastward-measured distance of geographic Cartesian coordinates (x-coordinate) | | |
| Location (North) | Northward-measured distance of geographic Cartesian coordinates (y-coordinate) | | |
| Longitude | Latitude is a geographic coordinate that specifies the north–south position of a point on the Earth's surface | | |
| Latitude | Longitude is a geographic coordinate that specifies the east-west position of a point on the Earth's surface | | |
| Accident Severity | Fatal or serious/slight injury. | | |
| Number of Vehicles | Number of vehicles involved in the accident. | | |
| Number of Casualties | Number of casualties in the accident. | | |
| Date | Date on which accident occurred (DD/MM/YYYY format). | | |
| Day of Week | Sunday to Saturday | | |
| Time | Time in 24 hour format | | |
| Speed Limit | Speed limit in the area | | |
| Light Conditions | Daylight or dark (without or with light) | | |
| Weather Conditions | Normal, Rain, wind or fog. | | |
| Urban or Rural Area | Town/City or Countryside | | |
| Accident Location Code | In the United Kingdom, the Office for National Statistics maintains a series of codes to represent a wide range of geographical areas of the UK, for use in tabulating census and other statistical data | | |

# Exploratory Data Analysis

Exploratory data analysis is an approach to analyze datasets to summarize their main characteristics, often with visual methods. A statistical model may be used, but primarily EDA is for seeing what the data can tell us prior to the formal modeling or hypothesis testing task.

## Accidents over the Days of the Week

We start by summarizing all the accidents spread across the days of the week. Saturday and Sunday are days with fewer accidents which could be explained by less traffic on the road on the weekends. While other weekdays have almost closer number of accidents while Friday is the highest.

The Nationwide Insurance report found Friday was the most dangerous day to commute to work by car. Nationwide concluded that drivers' concentration drops as the week wears on. "Everybody is anxious to start their weekends, so they're all thinking about something other than focusing on their driving," Bill Windsor, associate vice president of safety at Nationwide, told AOL Autos.

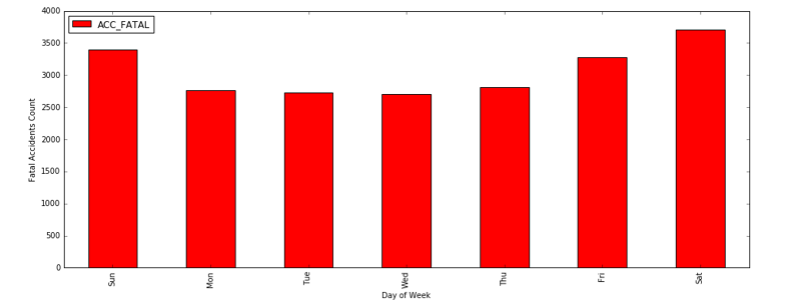
|  |  |
| --- | --- |
| **Day of Week** | **Number of Accidents** |
| Sunday | 180,068 |
| Monday | 233,238 |
| Tuesday | 245,275 |
| Wednesday | 247,022 |
| Thursday | 246,015 |
| Friday | 268,985 |
| Saturday | 219,994 |

## Accident Severity

Here we check the proportion of accidents and find that minor injuries are the most common with 85% of accidents and it is comforting to see that the fatal accidents are less than 1.5% on the whole.

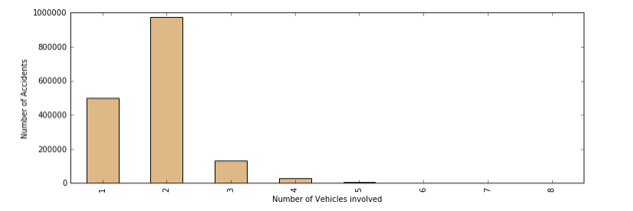
|  |  |  |
| --- | --- | --- |
| Description | Injuries | Percentage |
| Fatal | 21382 | 1.30 |
| Serious | 222042 | 13.53 |
| Slight | 1397173 | 85.16 |

We continue our analysis to plot the fatal accidents across the days of the week. Once again we see that the weekends have more fatal accidents compared to the weekdays.



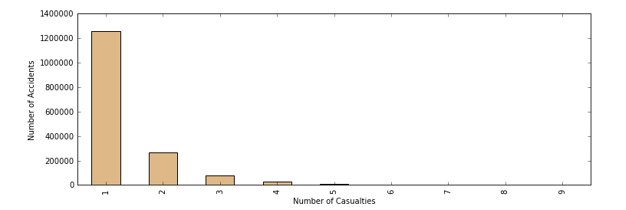
## Number of Vehicles involved in the Accidents

The number of vehicles involved in each accident show that two vehicles accidents are the most common. However, one vehicle accidents may be pedestrian related accidents, hit by a single vehicle or the driver did not hit anyone else. There also cases with more vehicles involved which can also indicate a vehicle related pile up during rain, fog or snow. The maximum number of vehicles involved in accidents are 34 and 67, which could have been pile up in extreme weather conditions.



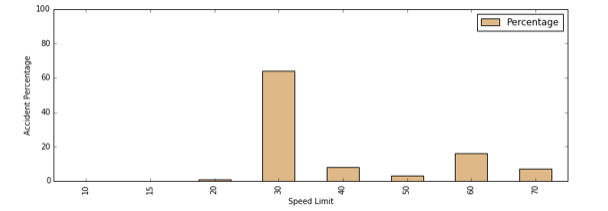
## Number of Casualties

Most of the accidents had mostly one person injured. There are also cases where as much as 93 persons were involved which could be related to more than one bus being involved.



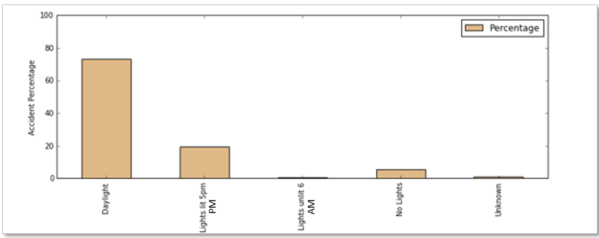
## Speed Limit

While analyzing how the speed limits where the accidents have occurred, we find that 30 mph seem to have the highest percentage of accidents. This could be an indication cities might be the highest place where accidents occur, followed by 60 mph which could be the highways.



## Light Conditions

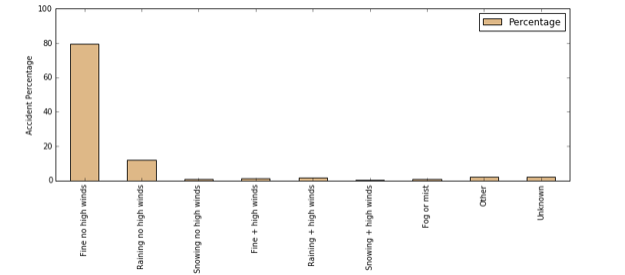
Accidents happen throughout the day and the below graph shows most accidents have happened during the day and also in early evening when dark.



’

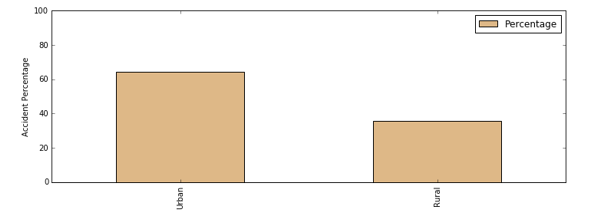
## Weather Conditions

The below chart shows that most accidents took place under normal weather conditions, while rainy condition takes second place.



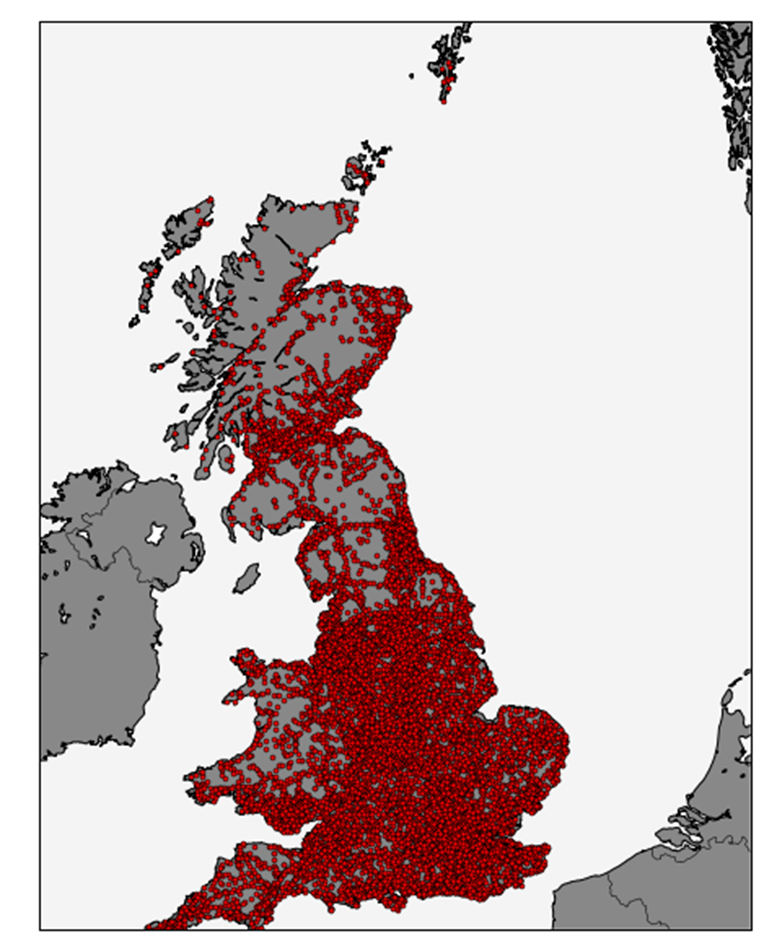
## Urban or Rural Area

Based on the chart below, we can safely conclude that most of the accidents have occurred in an urban area. This could also mean that the number of vehicles as well as traffic conditions is fewer in rural areas.



# Fatal Accidents across the United Kingdom

We attempt to isolate only the fatal accidents from 2005 to 2014 and plot it in United Kingdom map. Most of the fatal accidents are concentrated in the southern part of the map because population density is higher there.



# Logistic Regression

## Building the Prediction Model – Choosing the Attributes

It is time to go beyond the exploratory analysis and build a model for prediction. We need to look at all the available attributes in the dataset and pick the variables that are potential candidates for the model.

After carefully checking the attributes, we pick the following:

* Day of Week
* Number of Vehicles
* Number of Casualties
* Light Conditions
* Weather Conditions
* Urban or Rural Area
* Speed Limit (will be utilized as a Dummy Variable)

Dummy variables are independent variables which take the value of either 0 or 1. Just as a "dummy" is a stand-in for a real person, in quantitative analysis, a dummy variable is a numeric stand-in for a qualitative fact or a logical proposition.

However, social scientists often need to work with categorical variables in which the different values have no real numerical relationship with each other. Examples include variables for race, political affiliation, or marital status. Here the model performed well by using Speed as a dummy variable.

The solution is to use dummy variables - variables with only two values, zero and one.

## Performing the Regression

Actually doing the Logistic Regression is quite simple. Specify the column containing the variable we're trying to predict followed by the columns that the model should use to make the prediction.

|  |  |  |  |
| --- | --- | --- | --- |
| **Logit Regression Results** | | | |
| Dependent Variable | Observations | DF | Pseudo R-square |
| Fatal Accident\* | 1,640,597 | 12 | 0.07134 |

\*Whether or not accident was fatal.

|  |  |  |
| --- | --- | --- |
|  | Coefficient | p-value |
| intercept | -3.1182 | 0.000 |
| Number of Vehicles | -0.2187 | 0.000 |
| Number of Casualties | 0.3193 | 0.000 |
| Day of Week | -0.0013 | 0.704 |
| Light Conditions | 0.1458 | 0.000 |
| Weather Conditions | -0.0808 | 0.000 |
| Urban or Rural Area | 0.3778 | 0.000 |
| Speed Limit 20 mph | -2.6164 | 0.000 |
| Speed Limit 30 mph | -2.5370 | 0.000 |
| Speed Limit 40 mph | -1.9265 | 0.002 |
| Speed Limit 50 mph | -1.6528 | 0.007 |
| Speed Limit 60 mph | -1.4479 | 0.018 |
| Speed Limit 70 mph | -1.6748 | 0.006 |

|  |
| --- |
| **Area under the ROC Curve: 0.740256** |

## Interpreting the results

To isolate and evaluate coefficients, we prepare a confidence interval table.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **95% Confidence Interval** | | **Interpretation** |
| intercept | -4.318008 | -1.918320 |  |
| Number of Vehicles | -0.238077 | -0.199307 | More fatalities when the number of vehicles involved are less. |
| Number of Casualties | 0.308910 | 0.329621 |  |
| Day of Week | -0.008192 | 0.005532 | We have seen before number of fatalities progressively increase through the week and more accidents occurred in the weekends. |
| Light Conditions | 0.139080 | 0.152453 | More accidents when occurred in fine light conditions. |
| Weather Conditions | -0.090468 | -0.071078 | Accidents increase in improved weather conditions |
| Urban or Rural Area | 0.335821 | 0.419692 | Urban area had more accidents than rural areas. |
| Speed limit 20 mph | -3.829254 | -1.403597 | Accidents seem to be less at higher speeds. The reason could be more congested city areas where speed limits are restricted and the vehicles are more. There seems to be an inverse relationship between accidents and increase in Speed Limits. |
| Speed limit 30 mph | -3.735102 | -1.338971 |
| Speed limit 40 mph | -3.125220 | -0.727783 |
| Speed limit 50 mph | -2.852276 | -0.453370 |
| Speed limit 60 mph | -2.646206 | -0.249512 |
| Speed limit 70 mph | -2.873504 | -0.476034 |

## Odds ratio

An odds ratio (OR) is a measure of association between an exposure and an outcome. The OR represents the odds that an outcome will occur given a particular exposure, compared to the odds of the outcome occurring in the absence of that exposure.

|  |  |
| --- | --- |
|  | Odds Ratio |
| intercept | 0.044238 |
| Number of Vehicles | 0.803569 |
| Number of Casualties | 1.376116 |
| Day of Week | 0.998671 |
| Light Conditions | 1.156926 |
| Weather Conditions | 0.922403 |
| Urban or Rural Area | 1.459007 |
| Speed Limit 20 | 0.073064 |
| Speed Limit 30 | 0.079100 |
| Speed Limit 40 | 0.145657 |
| Speed Limit 50 | 0.191509 |
| Speed Limit 60 | 0.235073 |
| Speed Limit 70 | 0.187352 |

We can expect odds of accidents to increase in better lighting conditions (day light), urban area (city limits with more vehicles) as also the increase in casualties.

## Area Under the Curve (AUC)

One cannot really give an overview of ROC curves without mentioning AUC. The good news is it is exactly what it sounds like--the amount of space underneath the ROC curve. You can think of the AUC as sort of a holistic number that represents how well the model performs across thresholds.

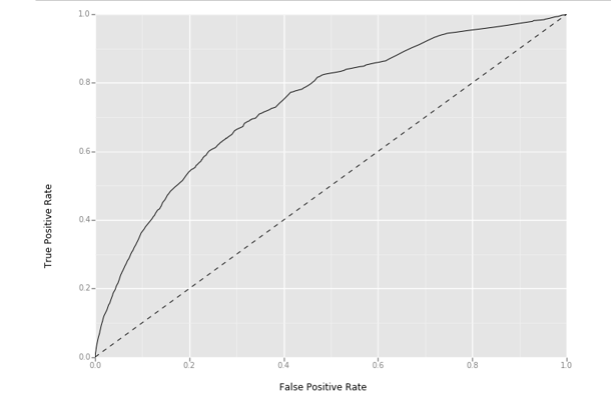
To make it super simple:

AUC=0 -> BAD

AUC=1 -> GOOD

So in the context of a ROC curve, the more "up and left" the curve approaches, the larger the AUC will be and thus, the better the classifier is. Comparing AUC values is also really useful when comparing different models, as we can select the model with the highest AUC value, rather than just look at the curves.

Now we build the ROC curve using the results from the Logistic regression model:



# Conclusion

## Analysis

We get a great overview of the coefficients of the model, how well those coefficients fit, the overall fit quality, and several other statistical measures.

We are very confident that there is an inverse relationship between the probability of a fatal accident and number of vehicles, weather conditions and speed limit.

We also observe:

* There are more fatalities when the number of vehicles involved are less.
* We have seen before number of fatalities progressively increase through the week and more accidents occurred in the weekends.
* More accidents when occurred in fine light conditions.
* Accidents increase in improved weather conditions
* Urban area had more accidents than rural areas.
* Accidents seem to be less at higher speeds. The reason could be more congested city areas where speed limits are restricted and the vehicles are more.

## Client Recommendation

More fatalities have occurred in the Southern part of United Kingdom. Drivers should exercise more caution while driving during the weekends, within city limits as well as driving fine weather conditions.

Here are a few driving tips for drivers in United Kingdom:

*Observe the speed limit* Most speed limits are indicated by black numerals on a circular white sign with a red border.  The exception is the National Speed Limit, a kind of "default" speed limit, indicated by a plain white circular sign with a black diagonal stripe. Speed limits signs and distance signs in the UK are always indicated in miles.

*Right on Red? Never! Left on Red? No!* There is no UK equivalent of "Right on Red" (this would be "Left on Red"). Always stop and wait for the light.  At some traffic lights there may be a green filter arrow. This indicates a filter lane only.

*Roundabouts*Known as "traffic circles" in those few places in the US where they are seen, roundabouts actually make some junctions a lot easier by keeping traffic moving. The two key things to remember are: Get in lane and stay there and give way to the right (traffic on the roundabout).

*Communicating with Other Drivers* It is often necessary to stray into the opposing lane to get around obstacles such as parked cars, vans unloading, or roadworks.  British drivers use hand signals and flash their lights to allow others into gaps, make room for each other and to thank other drivers for their courtesy. Confusingly, they also flash their lights as a rebuke too!  If someone helps you out, it is a good idea to wave to say "thanks".

## Limitations

* Logistic regression is an excellent algorithm for classification. Logistic regression attempts to predict outcomes based on a set of independent variables, but if we include the wrong independent variables, the model will have little to no predictive value.
* Logistic regression works well for predicting categorical outcomes like admission or rejection at a particular college, but logistic regression cannot predict continuous outcomes.
* Even though black box classification algorithms like SVM and RandomForest can perform better in some cases, it is hard to deny the value in knowing exactly what your model is doing.
* However we can get by using RandomForest to select the features of the model and then rebuild the model with Logistic Regression using the best features.

# References

* Road Safety Datasets and other UK Accidents related facts

<https://data.gov.uk/dataset/road-accidents-safety-data>

* Predicting Malaysian Road Fatalities

Sarani R, Allyana S, Voon WS. [J. Australas. Coll. Road Saf.](http://www.safetylit.org/week/journalpage.php?jid=7621) 2016; 27(2): 18-22

<http://www.academia.edu/14474529/Predicting_Malaysian_Road_Fatalities>

* Study on fatal accidents in Toyota city aimed at zero traffic fatality

<http://bast.opus.hbz-nrw.de/volltexte/2013/678/pdf/24_Kiuchi.pdf>

# Python Code used in the project

## Initialize libraries, get datasets, consolidate and cleanse

# Importing libraries and the dataset

from pandas import Series, DataFrame

import pandas as pd

import glob

import numpy as np

import os

import matplotlib.pylab as plt

%matplotlib inline

plt.rcParams['figure.figsize'] = 12, 4 # that's default image size for this interactive session

import scipy

from scipy import stats

from sklearn.cross\_validation import train\_test\_split

from sklearn import linear\_model

from sklearn.metrics import confusion\_matrix, precision\_recall\_fscore\_support, accuracy\_score

from sklearn.preprocessing import Binarizer

from sklearn.ensemble import RandomForestClassifier

from sklearn.preprocessing import StandardScaler

from sklearn import preprocessing

import statsmodels.api as sm

from sklearn.preprocessing import StandardScaler, OneHotEncoder

scaler = StandardScaler()

from sklearn.metrics import roc\_curve, auc

## Getting a feeling of the dataset

We will start exploring 2005-2015 consolidated dataset.

RoadSafetyData = pd.read\_csv("Accidents2005to2014.csv")

RoadSafetyData.shape

(1640597, 17)

RoadSafetyData.dtypes

﻿Accident\_Index object

Location\_Easting\_OSGR float64

Location\_Northing\_OSGR float64

Longitude float64

Latitude float64

Accident\_Severity int64

Number\_of\_Vehicles int64

Number\_of\_Casualties int64

Date object

Day\_of\_Week int64

Time object

Speed\_limit int64

Light\_Conditions int64

Weather\_Conditions int64

Urban\_or\_Rural\_Area int64

LSOA\_of\_Accident\_Location object

dtype: object

## Perform operations to get data distribution and NULL/NAN values

### Add data columns to isolate Accident Types and compute Accident Count

RoadSafetyData['ACC\_FATAL'] = np.where(RoadSafetyData.Accident\_Severity == 1, 1 , 0)

RoadSafetyData['ACC\_SEVERE'] = np.where(RoadSafetyData.Accident\_Severity == 2, 1 , 0)

RoadSafetyData['ACC\_SLIGHT'] = np.where(RoadSafetyData.Accident\_Severity == 3, 1 , 0)

### Compute the summary of Accidents grouped by Severity

bySeverity = RoadSafetyData.groupby('Accident\_Severity')

bySeverity ['Accident\_Severity'].count()

Accident\_Severity

1 21382

2 222042

3 1397173

Name: Accident\_Severity, dtype: int64

### Compute the summary of Accidents grouped by Number of Vehicles

bySeverity = RoadSafetyData.groupby('Number\_of\_Vehicles')

bySeverity ['Number\_of\_Vehicles'].count()

Number\_of\_Vehicles

1 497873

2 974317

3 131155

4 27874

5 6157

6 1902

7 683

8 333

9 141

10 69

11 29

12 17

13 12

14 11

15 3

16 6

17 1

18 3

19 2

20 2

21 1

22 1

28 1

29 1

32 1

34 1

67 1

Name: Number\_of\_Vehicles, dtype: int64

### Compute the summary of Accidents grouped by Number of Casualties

byDayOfWeek = RoadSafetyData.groupby('Number\_of\_Casualties')

byDayOfWeek['Number\_of\_Casualties'].count()

Number\_of\_Casualties

1 1258011

2 263773

3 75197

4 27178

5 10030

6 3760

7 1294

8 553

9 271

10 147

11 95

12 51

13 46

14 29

15 21

16 16

17 17

18 11

19 13

20 4

21 9

22 11

23 4

24 4

25 4

26 7

27 3

28 2

29 6

32 1

33 1

35 1

36 2

38 1

40 2

41 2

42 3

43 2

45 2

46 1

47 1

48 1

51 2

54 1

62 1

63 1

68 1

70 1

87 2

93 1

Name: Number\_of\_Casualties, dtype: int64

### Compute the summary of Accidents grouped by Day of Week

byDayOfWeek = RoadSafetyData.groupby('Day\_of\_Week')

byDayOfWeek['Day\_of\_Week'].count()

Day\_of\_Week

1 180068

2 233238

3 245275

4 247022

5 246015

6 268985

7 219994

Name: Day\_of\_Week, dtype: int64

### Compute the summary of Accidents grouped by Speed Limit

bySpeed\_limit = RoadSafetyData.groupby('Speed\_limit')

bySpeed\_limit['Speed\_limit'].count()

Speed\_limit

10 17

15 16

20 17727

30 1051982

40 134488

50 52880

60 263282

70 120205

Name: Speed\_limit, dtype: int64

### Compute the summary of Accidents grouped by Light Conditions

byLight\_Conditions = RoadSafetyData.groupby('Light\_Conditions')

byLight\_Conditions['Light\_Conditions'].count()

Light\_Conditions

1 1201866

4 322177

5 7466

6 91689

7 17399

Name: Light\_Conditions, dtype: int64

### Compute the summary of Accidents grouped by Weather Conditions

byWeather\_Conditions = RoadSafetyData.groupby('Weather\_Conditions')

byWeather\_Conditions['Weather\_Conditions'].count()

Weather\_Conditions

-1 161

1 1309195

2 194930

3 11860

4 20960

5 23490

6 2134

7 9037

8 37179

9 31651

Name: Weather\_Conditions, dtype: int64

### Compute the summary of Accidents grouped by Urban or Rural Area

bySex\_of\_Casualty = RoadSafetyData.groupby('Urban\_or\_Rural\_Area')

bySex\_of\_Casualty ['Urban\_or\_Rural\_Area'].count()

Urban\_or\_Rural\_Area

1 1054341

2 586113

3 143

Name: Urban\_or\_Rural\_Area, dtype: int64

## Exploratory Data Analysis

### Comparison of Accident Severity (Fatal, Severe or Slight Injury)

RoadSafetyData.pivot\_table(columns='Day\_of\_Week')

delays\_list = ['ACC\_FATAL','ACC\_SEVERE','ACC\_SLIGHT']

RoadSafetyData\_by\_day = RoadSafetyData.pivot\_table(index='Day\_of\_Week', values=delays\_list, aggfunc='sum')

RoadSafetyData\_by\_day.plot(kind='bar', figsize=[16,6], stacked=True, colormap='autumn') # area plot

ax1 = plt.axes()

ax1.set\_xticklabels(['Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat'])

plt.xlabel('Day of Week')

plt.ylabel('Accident Severity')

plt.title('Comparison of Accident Severity')

RoadSafetyData\_by\_day.columns = ['Fatal Accidents', 'Severe Accidents', 'Slight Accidents']

RoadSafetyData\_by\_day

### Comparison of Number of Accidents by Number of Vehicles

temp1 = RoadSafetyData[ (RoadSafetyData.ACC\_COUNT > 0) & (RoadSafetyData.Number\_of\_Vehicles < 20) ]

temp2 = temp1.groupby('Number\_of\_Vehicles').ACC\_COUNT.count()

temp2.plot(kind='bar', color='#DEB887')

plt.rcParams['figure.figsize'] = 12, 4

plt.xlabel('Number of Vehicles involved')

plt.ylabel('Number of Accidents')

plt.title(' Number of Accidents by Number of Vehicles involved')

### Comparison of Number of Accidents by Number of Casualties

temp1 = RoadSafetyData[(RoadSafetyData.ACC\_COUNT > 0) & (RoadSafetyData.Number\_of\_Casualties < 20)]

temp2 = temp1.groupby('Number\_of\_Casualties').ACC\_COUNT.count()

temp2.plot(kind='bar', color='#DEB887')

plt.xlabel('Number of Casualties')

plt.ylabel('Number of Accidents')

plt.title(' Number of Accidents by Number of Casualties')

### Comparison of Number of Accidents by Day of Week

temp1 = RoadSafetyData[ RoadSafetyData.ACC\_COUNT > 0]

temp2 = temp1.groupby('Day\_of\_Week').ACC\_COUNT.count()

temp2.plot(kind='bar', color='#DEB887')

ax1 = plt.axes()

ax1.set\_xticklabels(['Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat'])

plt.xlabel('Day of Week')

plt.ylabel('Number of Accidents')

plt.title(' Number of Accidents by Day of Week')

### Comparison of Number of Accidents by Speed Limit

temp1 = RoadSafetyData[ RoadSafetyData.ACC\_COUNT > 0]

temp2 = temp1.groupby('Speed\_limit').ACC\_COUNT.count()

temp2.plot(kind='bar', color='#DEB887')

plt.xlabel('Speed Limit')

plt.ylabel('Number of Accidents')

plt.title(' Number of Accidents by Speed Limit')

### Comparison of Number of Accidents by Light Conditions

temp1 = RoadSafetyData[ RoadSafetyData.ACC\_COUNT > 0]

temp2 = temp1.groupby('Light\_Conditions').ACC\_COUNT.count()

temp2.plot(kind='bar', color='#DEB887')

ax1 = plt.axes()

ax1.set\_xticklabels(['Daylight', 'Lights lit 5pm', 'Lights unlit 6', 'No Lights', 'Unknown'])

plt.xlabel('Light Conditions')

plt.ylabel('Number of Accidents')

plt.title(' Number of Accidents based on Light Conditions')

### Comparison of Number of Accidents by Weather Conditions

temp1 = RoadSafetyData[ (RoadSafetyData.ACC\_COUNT > 0) & (RoadSafetyData.Weather\_Conditions <> -1)]

temp2 = temp1.groupby('Weather\_Conditions').ACC\_COUNT.count()

temp2.plot(kind='bar', color='#DEB887')

ax1 = plt.axes()

ax1.set\_xticklabels(['Fine no high winds', 'Raining no high winds', 'Snowing no high winds',

'Fine + high winds','Raining + high winds','Snowing + high winds','Fog or mist','Other','Unknown'])

plt.xlabel('Weather Conditions')

plt.ylabel('Number of Accidents')

plt.title(' Number of Accicents based on Weather Conditions')

### Comparison of Number of Accidents by Urban or Rural Area

temp1 = RoadSafetyData[ RoadSafetyData.ACC\_COUNT > 0]

temp2 = temp1.groupby('Urban\_or\_Rural\_Area').ACC\_COUNT.count()

temp2.plot(kind='bar', color='#DEB887')

ax1 = plt.axes()

ax1.set\_xticklabels(['Urban', 'Rural'])

plt.xlabel('Urban or Rural\_Area')

plt.ylabel('Number of Accidents')

plt.title(' Number of Accidents based on Urban or Rural Area')

## Plotting all Fatal Accidents location on a map using Latitude and Longitude

# Import pandas

import pandas as pd

# Import matplotlib and Basemap

import matplotlib.pyplot as plt

from mpl\_toolkits.basemap import Basemap

# Set iPython to display visualization inline

%matplotlib inline

# Create a figure of size (i.e. pretty big)

fig = plt.figure(figsize=(20,10))

fig.suptitle('Fatal Accidents 2005 to 2014', fontsize=20)

# Create a map, using the Gall–Peters projection, gall

map = Basemap(width=120000000,height=90000000,projection='lcc', llcrnrlon=-7.5600,llcrnrlat=49.9600,urcrnrlon=5.7800,

urcrnrlat=60.8400, resolution = 'h', epsg=5520)

# Draw the coastlines on the map

map.drawcoastlines()

# Draw country borders on the map

map.drawcountries()

# Fill the land with grey

map.fillcontinents(color = '#888888')

# Draw the map boundaries

map.drawmapboundary(fill\_color='#f4f4f4')

# Define our longitude and latitude points

# We have to use .values because of a wierd bug when passing pandas data

# to basemap.

fatal\_data = RoadSafetyData[RoadSafetyData['ACC\_FATAL'] == 1]

x,y = map(fatal\_data['Longitude'].values, fatal\_data['Latitude'].values)

# Plot them using round markers of size 6

map.plot(x, y, 'ro', markersize=3)

# Show the map

plt.show()

## Perform Logistic Regression using StatsModels

#Reading the data from 2005 - 2014 Accident Data

df = pd.read\_csv("Accidents2005to2014.csv")

df['ACC\_FATAL'] = np.where(df.Accident\_Severity == 1, 1 , 0)

#df.head()

#drop unnecesary columns

df.drop(['Unnamed: 0', 'Location\_Easting\_OSGR', 'Location\_Northing\_OSGR', 'Longitude', 'Latitude'], axis=1, inplace=True)

df.drop(['Date', 'Time', 'LSOA\_of\_Accident\_Location'], axis=1, inplace=True)

df.columns = ['﻿Accident\_Index','Accident\_Severity','Number\_of\_Vehicles','Number\_of\_Casualties','Day\_of\_Week',

'Speed\_limit','Light\_Conditions','Weather\_Conditions','Urban\_or\_Rural\_Area','ACC\_FATAL']

# making speed limit a dummy column

dummy\_ranks = pd.get\_dummies(df['Speed\_limit'], prefix='Speed\_limit')

# create a clean data frame for the regression

cols\_to\_keep = ['Day\_of\_Week', 'Speed\_limit', 'Light\_Conditions',

'Weather\_Conditions', 'Urban\_or\_Rural\_Area','Number\_of\_Vehicles', 'Number\_of\_Casualties','ACC\_FATAL']

data = df[cols\_to\_keep].join(dummy\_ranks.ix[:, 'Speed\_limit\_2':])

# manually add the intercept to gaurantee that the residuals have a mean of zero

data['intercept'] = 1

train\_cols = data[['intercept','Number\_of\_Vehicles','Number\_of\_Casualties','Day\_of\_Week','Light\_Conditions',

'Weather\_Conditions', 'Urban\_or\_Rural\_Area',

'Speed\_limit\_20' ,'Speed\_limit\_30','Speed\_limit\_40', 'Speed\_limit\_50',

'Speed\_limit\_60', 'Speed\_limit\_70' ]]

#train\_cols.head()

test\_cols = data['ACC\_FATAL']

#test\_cols.head()

result = sm.Logit(test\_cols, train\_cols).fit()

print result.summary()

# Add prediction to dataframe

data['pred'] = result.predict(train\_cols)

fpr, tpr, thresholds =roc\_curve(data['ACC\_FATAL'], data['pred'])

roc\_auc = auc(fpr, tpr)

print("Area under the ROC curve : %f" % roc\_auc)

####################################

# The optimal cut off would be where tpr is high and fpr is low

# tpr - (1-fpr) is zero or near to zero is the optimal cut off point

####################################

i = np.arange(len(tpr)) # index for df

roc = pd.DataFrame({'fpr' : pd.Series(fpr, index=i),'tpr' : pd.Series(tpr, index = i), '1-fpr' : pd.Series(1-fpr, index = i),

'tf' : pd.Series(tpr - (1-fpr), index = i), 'thresholds' : pd.Series(thresholds, index = i)})

roc.ix[(roc.tf-0).abs().argsort()[:1]]

# Plot tpr vs 1-fpr

fig, ax = plt.subplots()

plt.plot(roc['tpr'])

plt.plot(roc['1-fpr'], color = 'red')

plt.xlabel('1-False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver operating characteristic')

ax.set\_xticklabels([])

## ROC Curve

from sklearn import metrics

import pandas as pd

from ggplot import \*

df = pd.DataFrame(dict(fpr=fpr, tpr=tpr))

plt.xlabel('1-False Positive Rate')

plt.ylabel('True Positive Rate')

ggplot(df, aes(x='fpr', y='tpr')) +\

geom\_line() +\

geom\_abline(linetype='dashed') + xlab("False Positive Rate") + ylab("True Positive Rate")