1 Introduction

Road traffic accidents are a major source of death and injury worldwide, with more than 1.2 million people killed each year, resulting in an estimated cost of $518 billion USD [1]. In the UK, there were a total of 174,510 road accident casualties of all severities with 27,010 of those being fatal or serious in the year 2017 [2]. A literature review conducted by Weijermars et al. found that the reported proportions of casualties in the Netherlands experiencing long term disabilities as of a result of road accidents varied between 11% and 80%, depending on the types of road users, the duration of the follow-up and the type of disabilities that were taken into account [3]. The total value of prevention of reported road accidents due to loss of property and healthcare in the UK for 2012 was estimated to be £15.1 billion [4]. Research into the origins of road accidents may help to decrease the incidence rates, thereby reducing: loss of property, disability and casualties.

Big Data (BD) generates several problems for conventional computing, namely: extreme hardware requirements and time-consuming data processing. In order to circumvent this, BD technologies typically use distributed computing. This gives BD technologies the ability to store, manipulate and analyse large amounts of traffic accident data, while avoiding the need for extreme hardware requirements and time-consuming data processing [5]. This report aims to provide further insight into the origin of road traffic accidents by analysing Road Safety Data in the UK from the year 2017 using BD technologies in order to answer the following problem statements:

1. The top 3 Police Force that have most the Fatal Accidents.
2. The top 5 Local Authorities that have most Serious Accidents.
3. The Vehicle Type that have most Serious and Fatal Accidents.
4. The Age Band of Casualty that have most Fatal Accidents.
5. Group and display the Slight Accidents by Day of Week in West Yorkshire Police Force Area.
6. The peak hour that have most Fatal Accidents in Dual carriageway.
7. The area that have most Fatal Motorcycle Accidents

2 Data

The data used showed detailed road safety data about the circumstances of personal injury road accidents in GB for the year 2017. 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. As well as giving details of date, time and location, the accident file gives a summary of all reported vehicles and pedestrians involved in road accidents and the total number of casualties, by severity. Figures for deaths refer to persons killed immediately or who died within 30 days of the accident. The data consists of three main tables:

1. **Accidents**: 32 variables, detailing the location, time, date, lighting, weather, and road surface conditions, number of causalities, road type and other variables.

2. **Casualties**: linked via "Accident Index" to the Accidents table, this table has 16 columns, giving further detail on the casualties involved. Each row represents a single person injured in a collision.

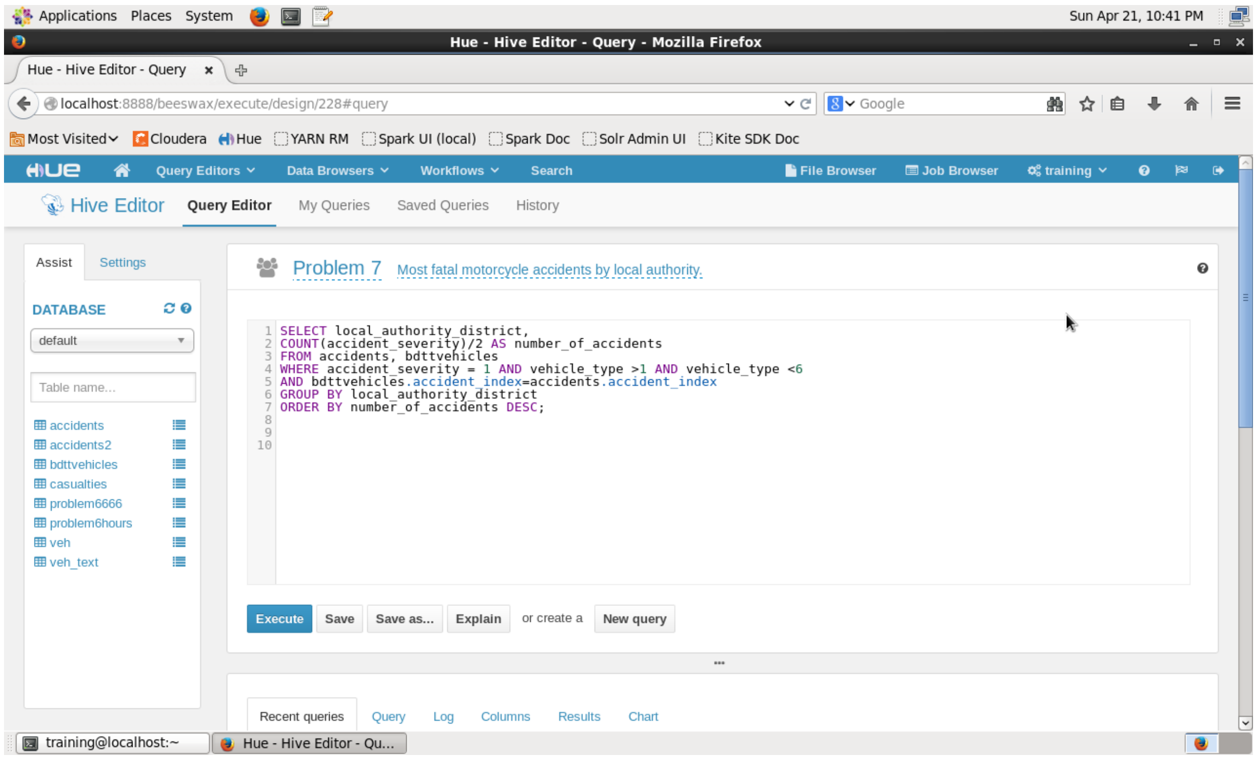
3. **Vehicles**: This table gives details of the vehicles involved in collisions, in addition to the drivers details. the table has 23 columns.

3 Methods

To begin implementing the solution using BD technologies, the Apache Hadoop Distributed File System (HDFS) was used. HDFS is a distributed file system which is used for storage and processing of big data using the MapReduce programming framework [6]. Appendix A (Listing 1) shows the script used during data loading. The Accidents and Casualties tables were loaded into the HDFS, then into Apache Hive using Apache Sqoop. The Vehicles table was imported into the HDFS in CSV format, then into Hive. The tables used ranged between 16 and 32 columns, so bulk cleaning measures to remove an instance which contained missing or null values were not taken to prevent unnecessary loss of data; instead, data was filtered on a case-by-case basis to conserve as much data possible.

3.1 Hive Implementation

Hive is an open-source data warehousing system that runs on top of Apache Hadoop. Hive structures data into the well-understood database concepts like tables and partitions. Hive data can be queried by an SQL-like declarative language called HiveQL, which are compiled into map-reduce jobs that are executed using Hadoop [7, 8]. Cloudera Hive User Experience (Hue) was adopted for Hive implementation. Cloudera is one the three major Hadoop distributors, alongside Hortonworks and MapR. Hue is an open source web interface for analysing with Hadoop and its ecosystem components (Figure 1) [9]. Hue provides realtime interactive querying and visualization of results, though Hue visualization functionality was not used in this report. Additionally, Hue provides numerous tools for security, data management and also supports structures as well as unstructured data. The principal drawback of Hue is that join operations are performed in memory that is limited by the smallest memory node present in the cluster, this causes longer processing times for queries which contain joins [10].



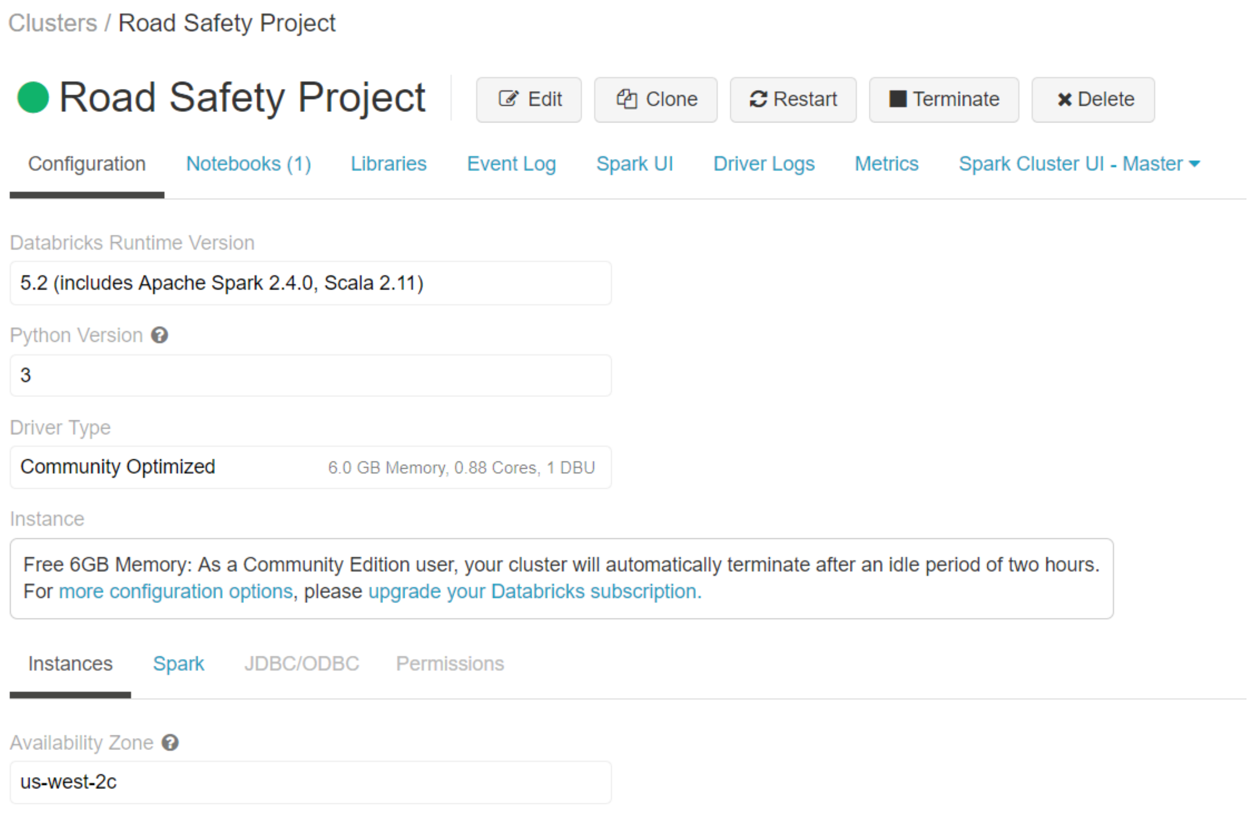
```
SELECT local_authority_district,
COUNT(accident_severity)/2 AS number_of_accidents
FROM accidents, bdttvehicles
WHERE accident_severity = 1 AND vehicle_type >1 AND vehicle_type <6
AND bdttvehicles.accident_index=accidents.accident_index
GROUP BY local_authority_district
ORDER BY number_of_accidents DESC;
```

Figure 1: Screenshot showing Cloudera Hue HiveQL Editor which was used for Hive implementation.

To find the top 3 police forces that have the most fatal accidents the "Police_Force" column was selected and the number of instances where "Accident_Severity" was equal to 1 were counted and grouped by "Police_Force". The same procedure was applied to find the top 5 local authorities that have the most serious accidents but in the case "Local_Authority_District" was selected and the number of instances where "Accident_Severity" was equal to 2 were counted. The vehicle type that had the most serious and fatal accidents was found by selecting "Accident_Severity" from the accidents table and "Vehicle_Type" from the vehicles table. These tables were joined where the accident indices were equivalent. The number of incidents where "Accident_Severity" was equal to 1 were counted. The age band of casualty which had the most fatal accidents was found by first joining the casualties table to the accidents table where the accident indices were equivalent. The number of instances of were counted when "Casualty_Severity" was equal to one. The number of slight accidents by day of week in the West Yorkshire police for area was found by counting the number of instances when "Accident_Severity" $=1$ and "Police_Force" $=13$ from the accidents table. The peak hour that had the most fatal accidents on dual carriageways was found by first using the "substr" function to split the "Time" column on the accidents table into its constituent parts to form two new columns "Hours" and "Minutes" based on the delimiter ":". The number of instances were counted when "Accident_Severity" $=1$ and "Road_Type" $=3$. The counts were grouped by "Accident_Severity", Hours and "Road_Type". The area that had the most fatal motorcycle accidents was found by joining the vehicles table to the accidents table when the accident indices were equivalent and then counting the number of instances when "Accident_Severity" $=1$, "Vehicle_Type" >1 and "Vehicle_Type" <6.

3.2 Spark Implementation via Databricks

Spark is a cluster computing framework, which supports applications with working sets while providing similar scalability and fault tolerance properties to MapReduce [11]. Databricks Community Edition (CE) was adopted for Spark implementation in this project. Databricks CE is a cloud computing service which runs on top of Apache Spark. In most cases a SparkContext is required to tell Spark how and where to access a cluster; however, Databricks CE does this automatically [12]. Figure 1 shows the cluster configuration used on Databricks CE and a Python 3 notebook was used to interface with the cluster. The data tables were manually uploaded using the Databricks graphical interface and imported into the Python 3 notebook using the script shown in Appendix A, listing 2.



Databricks Runtime Version
5.2 (includes Apache Spark 2.4.0, Scala 2.11)

Python Version
3

Driver Type
Community Optimized 6.0 GB Memory, 0.88 Cores, 1 DBU

Instance
Free 6GB Memory: As a Community Edition user, your cluster will automatically terminate after an idle period of two hours. For more configuration options, please upgrade your Databricks subscription.

Availability Zone
us-west-2c

Figure 2: Databricks CE cluster configuration for Road Safety Project.

To find the top 3 police forces which have the most fatal accidents the "accidents" table was filtered such that instances where "Accident_Severity" was equal to 1 were returned. The "groupby" and "count" functions were then used to aggregate the number of fatal accidents for each police station. A similar approach was taken to find the number of serious accidents by local authority; however, this problem required the accidents table to be filtered where "Accident_Severity" was equal to 2. The "groupby" and "count" functions were then used to aggregate the number of severe accidents for each local authority. The vehicle type that had the most accidents was found by joining the vehicles and accidents tables where the accident indices were the same and aggregating the vehicle type instances when the accident severity was 1 or 2. To find the age band which had the most fatal accidents the accidents and vehicles tables were joined in the same fashion and the number of instances of "Age_Band_of_Casualty" were aggregated. The number of slight accidents by day of week in the West Yorkshire police force area was calculated by first filtering the accidents table such that "Accident_Severity" was equal to 3, "Police_Force" was equal to 13 and the "Day_of_Week" variable was greater than or equal to 0. The number of accidents were then counted and grouped. The peak hour that has the most fatal accidents on dual carriage ways was found by first filtering the accidents table such that instances which contained values of "Accident_Severity" = 1 and "Road_Type" = 3 were returned. The "posexplode" function was used to substring the "Time" variable into two separate columns based on the ":" delimiter. The "posexplode" function creates two columns, one represents the position of a value, e.g. before or after the delimiter and the other column returns the value. Values with position "0" corresponded to hours and position 1 corresponded to minutes. The number of fatal accidents for each hour were grouped and counted to find the number of fatal accidents in each hour band. Finally, the area that has the most fatal motorcycle accidents was found by first joining the vehicles and accidents tables where "Accident_Index" values were equivalent and then filtering the tables to return instances where "Vehicle_Type" was equal to 2,3,4,5 and "Accident_Severity" was equal to 1. The number of fatal accidents were grouped by "Local_Authority_(District)".

4 Results

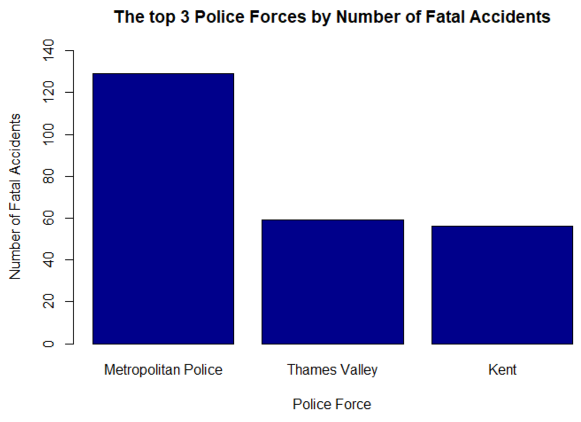
The top 3 police forces by number of accidents are shown in Figure 3(a). The Metropolitan Police Force had the highest number of fatal accidents. Table 1 shows the number of fatal accidents for the top 5 police forces, Metropolitan Police Force had 129 fatal accidents in the year 2017. Table 2 shows the top 5 local authorities by number of serious accidents, they were: Birmingham, Leeds, Sheffield, Westminster and Cornwall. Birmingham had 375 fatal accidents in the year 2017, making it the local authority which had the highest number of fatal accidents. Figure 3(c) shows the top 5 vehicle types by number of fatal accidents. Cars were involved in 25,596 accidents, making them the vehicle type with the most accidents. Figure 3(d) shows the age bands of casualties in order of how many fatal accidents they were involved with. The 26-35 age band had the most fatal accidents, this age band totalled 325 fatal accidents.

Local Authority	Fatal Accidents (N)
Metropolitan Police	129
Thames Valley	59
Kent	56
West Midlands	54
Devon and Cornwall	54

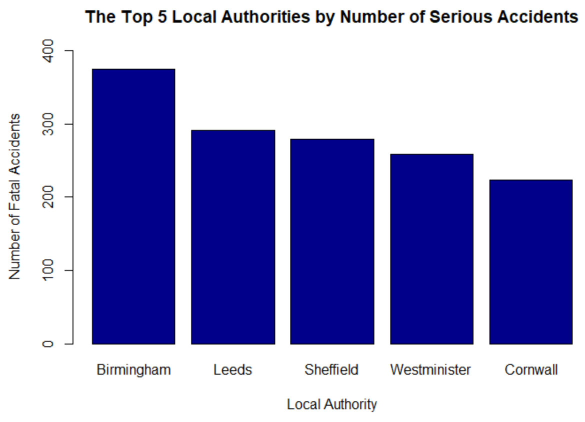
Table 1: The top 5 police forces which had the greatest number of fatal accidents

Local Authority	Fatal Accidents (N)
Birmingham	375
Leeds	291
Sheffield	279
Westminster	259
Cornwall	224

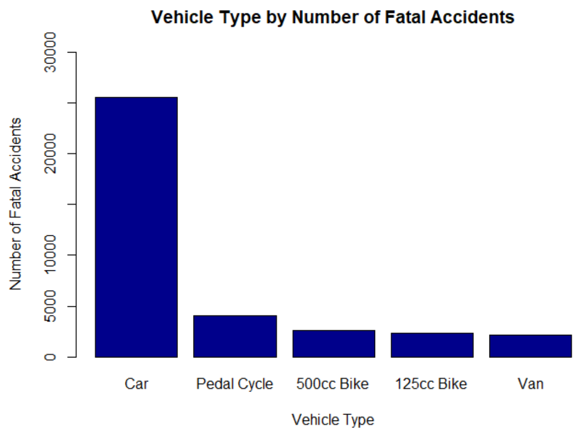
Table 2: The top 5 local authorities which had the greatest number of fatal accidents



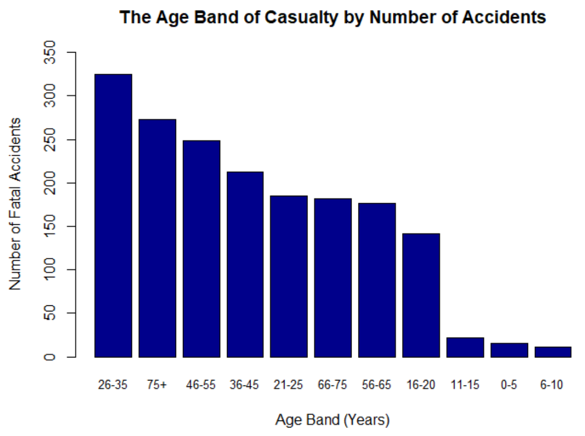
(a)



(b)



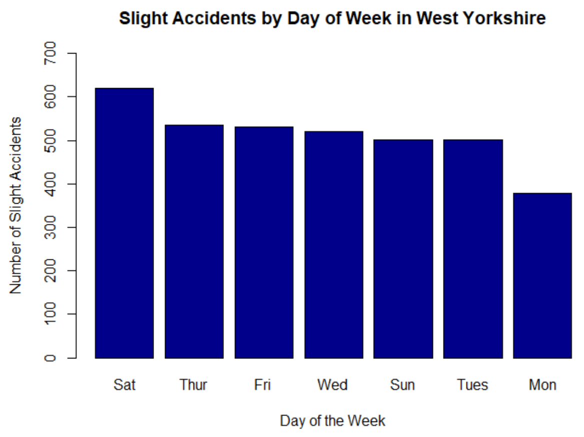
(c)



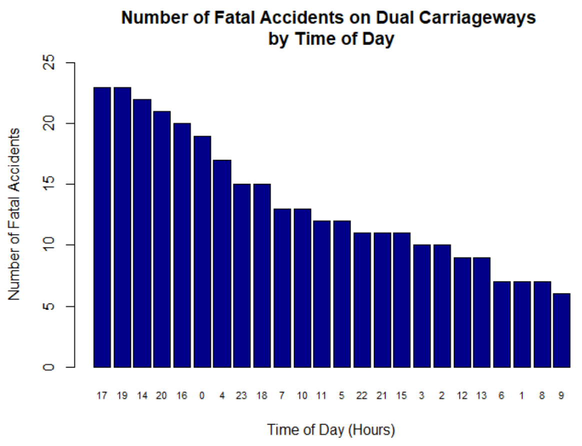
(d)

Figure 3: Graph (a): the top 3 police forces by number of fatal accidents. Graph (b): the top 3 local authorities by number of serious accidents. Graph (c): the number of fatal accidents by vehicle type. Graph (d): the age band of casualty by number of accidents, in order of frequency.

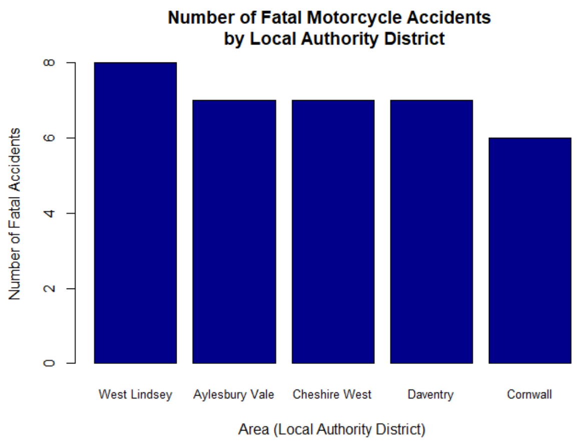
Figure 4(a) shows that Saturday was the day in which most slight accidents occurred in West Yorkshire. Figure 4(b) shows the number of fatal accidents on dual carriageways by time of day. The periods between 17:00-18:00 and 19:00-20:00 were the periods in which there was peak incidence of road accidents. The period between 09:00-10:00 had the least amount of accidents. The area which had the most fatal motorcycle accidents was West Lindsey, shown in Figure 4(c).



(a)



(b)



(c)

Figure 4: Graph (a): The number of slight accidents by day of the week in West Yorkshire. Graph (b): number of fatal accidents by time of day on dual carriageways, ordered by highest number of accidents. Graph (c): the number of fatal motorcycle accidents of all engine sizes by local authority district.

5 Discussion

Insight into the origins of road accidents were hindered in some cases because the data used was not normalized for population size. An example of where this manifests includes the top 5 local authorities that have the most serious accidents. Naturally, we would expect the most populated local authorities to predominate the local authorities which have the highest number serious accidents. This problem can be addressed by finding the top 5 local authorities which have the most accidents per unit 100,000 of the population; by doing so, anomalous local authorities could be more easily identified. This approach would also be useful for providing more meaningful insights into the number of fatal motorcycle accidents by local authority district. The total number of fatal motorcycle accidents could be determined on a national scale and then used to calculate a national average per unit 100,000 of the population. The rate of fatal accidents per unit 100,000 of the population for each area could then be compared to the national average, which would highlight areas which have a disproportionate rate of fatal motorcycle accidents.

Future research could involve more sophisticated analysis, for example Rovsek et al identified the key risk factors of traffic accident injury severity on Slovenian roads by using the CART algorithm [13]. The CART algorithm is a binary recursive partitioning procedure capable of processing continuous and nominal attributes as targets and predictors [14]. In CART splitting, an instance which meets the splitting criterion is branched left and instances which do not meet the splitting criterion are branched right. At a node t, the best split s is chosen to maximise the a splitting criterion $\Delta\text{i}(s,\ t)$. When the impurity measure for a node can be defined, the splitting criterion corresponds to a decrease in impurity. The Gini impurity at a node t is defined as

$$i(t) = \Sigma_{i,j} C(i \mid j) p(i \mid t) p(j \mid t) \tag{1}$$

Where $C(i \mid j)$ is the cost of misclassifying a class j case as a class i case, $p(i \mid t)$ is the probability of case i at node t and $p(j \mid t)$ is the probability of case j at node t.

$$\Delta i(s,t) = i(t) - p_L \cdot i(t_L) - p_R \cdot i(t_R) \tag{2}$$

Where p_L and p_R are the respective probabilities of a case being branched to the left daughter node t_L or right daughter node t_R [15]. The CART algorithm could be useful for determining variable importance with respect to fatal accidents. For instance, the accidents table contains information on weather, lighting and surface conditions; a recursive partitioning model using the CART algorithm could identify which of these variables has the most weighting on an accident outcome, e.g fatal or severe. Furthermore, CART splitting could determine the probability of an accident being fatal with a set of given conditions, making it a viable avenue for future research.

6 Conclusion

In the UK, there were a total of 174,510 road accident casualties of all severities with 27,010 of those being fatal or serious in the year 2017. The total value of prevention of reported road accidents due to loss of property and healthcare in the UK for 2012 was estimated to be £15.1 billion. Road Safety Data in the UK from the year 2017 were analysed using BD technologies in order to provide insight into the origins of traffic accidents. The Metropolitan Police Force had the most fatal accidents reported. Birmingham was the local authority with the greatest number of serious accidents. Cars were the vehicle type which had the greatest number of fatal accidents. 26-35 year old drivers were the age group which were involved in the greatest number of fatal accidents. Saturday was the day in which most slight accidents occured in West Yorkshire. 17:00-18:00 and 19:00-20:00 were the time periods in which most fatal accidents occured on dual carriageways and West Lindsey was the area which had the most fatal motorcycle accidents.

References

[1] R. Ivers, K. Brown, R. Norton, and M. Stevenson, “Road traffice injuries,” *International Encyclopedia of Public Health*, pp. 393–400, 2017.

[2] A. Dhani and S. Reynolds, “Report road casualties in great britain: quarterly provisional estimates year ending september 2017,” in *Road accidents and safety statistics.* UK Department for Transport, 2017, pp. 1–6.

[3] W. Weijermars, N. Bos, and H. Stipdonk, “Health burden of serious road injuries in the netherlands,” *Traffic injury prevention*, vol. 17, no. 8, pp. 863–869, 2016.

[4] D. Lloyd, “Reported road casualties in great britain: 2012 annual report,” in *Road accidents and safety statistics.* UK Department for Transport, 2012, pp. 1–12.

[5] J.-K. Choi, K.-H. Jeon, Y. Won, and J.-J. Kim, “Application of big data analysis with decision tree for the foot disorder,” *Cluster Computing*, vol. 18, no. 4, pp. 1399–1404, 2015.

[6] D. Borthakur, J. Gray, J. S. Sarma, K. Muthukkaruppan, N. Spiegelberg, H. Kuang, K. Ranganathan, D. Molkov, A. Menon, S. Rash *et al.*, “Apache hadoop goes realtime at facebook,” in *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data.* ACM, 2011, pp. 1071–1080.

[7] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy, “Hive: a warehousing solution over a map-reduce framework,” *Proceedings of the VLDB Endowment*, vol. 2, no. 2, pp. 1626–1629, 2009.

[8] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, N. Zhang, S. Antony, H. Liu, and R. Murthy, “Hive-a petabyte scale data warehouse using hadoop,” in *2010 IEEE 26th international conference on data engineering (ICDE 2010).* IEEE, 2010, pp. 996–1005.

[9] H. Bansal, S. Chauhan, and S. Mehrotra, *Apache Hive Cookbook.* Packt Publishing Ltd, 2016, pp. 19–36.

[10] G. singh Bhathal and A. Dhiman, “Big data solution: Improvised distributions framework of hadoop,” *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, pp. 35–38, 2018.

[11] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, “Spark: Cluster computing with working sets.” *HotCloud*, vol. 10, no. 10-10, p. 95, 2010.

[12] K. Pritwani, K. Wasley, and J. Woo, “Spark big data analysis of world development indicators,” *Global Journal of Computer Science and Technology*, vol. 18, 2018.

[13] V. Rovšek, M. Batista, and B. Bogunović, “Identifying the key risk factors of traffic accident injury severity on slovenian roads using a non-parametric classification tree,” *Transport*, vol. 32, no. 3, pp. 272–281, 2017.

[14] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, S. Y. Philip *et al.*, “Top 10 algorithms in data mining,” *Knowledge and information systems*, vol. 14, no. 1, pp. 1–37, 2008.

[15] L. Breiman, *Classification and Regression Trees*, 1st ed. New York: Routledge, 1984.

A Raw Scripts

Listing 1: VM Data Loading Script

```
#### Loading data in MySQL

# Login into MySQL server (password:"training")

mysql -u training -p

# Create new database to store files

create database bdtt_coursework;

# Check that it worked

show databases;

# Exite MySQL shell

exit

# Change cd to where the db_bdtt_ac is stored

cd "/home/training/Desktrop/Assignment_files"

# Load the file into the MySQL database

mysql -u training -p bdtt_coursework < db_bdtt_ac

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

#### Loading data to be analyzed with sqoop
#### connect to database and check that previous steps worked

sqoop list-tables \
--connect jdbc:mysql://localhost/bdtt_coursework \
--username training \
--password training

# This shows the accidents and casualties tables are located in the
   database
# The next step is to import the accidents table from MySQL into
   HDFS

sqoop import-all-tables \
--connect jdbc:mysql://localhost/bdtt_coursework \
--username training \
--password training \
--warehouse-dir /coursework/loadHDFS \
-m 1

## Check it worked

hdfs dfs -ls /coursework/loadHDFS

## The HDFS dir contains both databases
## Vehicles is now loaded into the HDFS

cd "/home/training/Desktrop/Assignment_files/"
hdfs dfs -put Veh.csv /coursework/loadHDFS
hdfs dfs -put

## Check it worked

hdfs dfs -ls /coursework/loadHDFS

## The directory now contains vehicles


#### Importing to Hive

sqoop import-all-tables \
--connect jdbc:mysql://localhost/bdtt_coursework \
--username training \
--password training \
--warehouse-dir /coursework/Hive \
--hive-import \
-m 1

#### Import CSV into Hive


CREATE EXTERNAL TABLE IF NOT EXISTS bdttvehicles
(
Accident_Index STRING,
Vehicle_Reference INT,
Vehicle_Type INT,
Towing_and_Articulation INT,
Vehicle_Manoeuvre INT,
Vehicle_Location_Restricted_Lane INT,
Junction_Location INT,
Skidding_and_Overturning INT,
Hit_Object_in_Carriageway INT,
Vehicle_Leaving_Carriageway INT,
Hit_Object_off_Carriageway INT,
1st_Point_of_Impact INT,
Was_Vehicle_Left_Hand_Drive INT,
Journey_Purpose_of_Driver INT,
Sex_of_Driver INT,
Age_of_Driver INT,
Age_Band_of_Driver INT,
Engine_Capacity INT,
Propulsion_Code INT,
Age_of_Vehicle INT,
Driver_IMD_Decile INT,
Driver_Home_Area_Type INT,
Vehicle_IMD_Decile INT
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
LOCATION '/coursework/loadHDFS/Veh.csv'
TBLPROPERTIES ("skip.header.line.count"="1");


#### input data
#### Change to hive storage
LOAD DATA INPATH '/coursework/loadHDFS/Veh.csv' OVERWRITE INTO TABLE
    Veh_text;

# Check it was successful

SELECT FROM Vehicle_Reference FROM Veh_text LIMIT 5;

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

INSERT OVERWRITE TABLE Veh SELECT * FROM Veh_text;
```

Listing 2: Databricks Python Script

```
# Cluster JSON Configuration

{
"num_workers": 0,
"cluster_name": "Road Safety Project",
"spark_version": "5.2.x-scala2.11",
"spark_conf": {
"spark.databricks.delta.preview.enabled": "true"
},
"aws_attributes": {
"first_on_demand": 0,
"availability": "ON_DEMAND",
"zone_id": "us-west-2c",
"spot_bid_price_percent": 100,
"ebs_volume_count": 0
},
"node_type_id": "dev-tier-node",
"driver_node_type_id": "dev-tier-node",
"ssh_public_keys": [],
"custom_tags": {},
"spark_env_vars": {
"PYSPARK_PYTHON": "/databricks/python3/bin/python3"
},
"autotermination_minutes": 120,
"enable_elastic_disk": false,
"cluster_source": "API",
"init_scripts": [],
"cluster_id": "0421-084342-taper614"
}

# File location and type
Acc_location = "/FileStore/tables/Acc.csv"
Veh_location = "FileStore/tables/Veh.csv"
Cas_location = "FileStore/tables/Cas.csv"
file_type = "csv"

# CSV options
infer_schema = "false"
first_row_is_header = "True"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be
    ignored.
Acc = spark.read.format(file_type) \
.option("inferSchema", infer_schema) \
.option("header", first_row_is_header) \
.option("sep", delimiter) \
.load(Acc_location)

Veh = spark.read.format(file_type) \
.option("inferSchema", infer_schema) \
.option("header", first_row_is_header) \
.option("sep", delimiter) \
.load(Veh_location)

Cas = spark.read.format(file_type) \
.option("inferSchema", infer_schema) \
.option("header", first_row_is_header) \
.option("sep", delimiter) \
.load(Cas_location)

from pyspark.sql import functions as f


fatal_accidents_by_pf = Acc.filter((f.col("Accident_Severity")=="1"))\
.groupBy("Police_Force") \
.count().sort(f.desc("count"))
display(fatal_accidents_by_pf)

serious_accidents_by_la = Acc.filter((f.col("Accident_Severity")=="2"))\
.groupBy("Local_Authority_(District)") \
.count().sort(f.desc("count")) \
.limit(5)
display(serious_accidents_by_la)

VehProb3filtered = Veh.filter((f.col("Vehicle_Type").isin
    (1,2,3,4,5,8,9,10,11,16,17,18,19,20,21,22,23,90,97,98)))
accidents_by_vehicle = Acc[(Acc.Accident_Severity <= "2")]\
.join(VehProb3filtered, Acc.Accident_Index == VehProb3filtered.Accident_Index)
    \
.groupBy(VehProb3filtered.Vehicle_Type)\
.count().sort(f.desc("Count"))
display(accidents_by_vehicle)

fatal_by_age_band = Cas.filter((f.col("Casualty_Severity")=="1"))\
.join(Acc, Cas.Accident_Index == Acc.Accident_Index) \
.groupBy(Cas.Age_Band_of_Casualty,"Casualty_Severity") \
.count().sort(f.desc("count"))
display(fatal_by_age_band)

slight_accidents_WYpolice = Acc[(Acc.Accident_Severity=="3") & (Acc.
    Police_Force =="13") & (Acc.Day_of_Week >= "0")]\
.groupBy("Day_of_Week","Accident_Severity", "Police_Force") \
.count().sort(f.desc("count"))
display(slight_accidents_WYpolice)

Accfiltered =  Acc[(Acc.Accident_Severity=="1") & (Acc.Road_Type =="3")]
Accfiltered2 = Accfiltered.select(
"Time", "Accident_Severity", "Road_Type",
f.posexplode(f.split("Time", ":")).alias("Pos", "Hr")
)
Accfiltered3 = Accfiltered2[(Accfiltered2.Pos=="0")]

Acc4 = Accfiltered3\
.groupBy("Accident_Severity", "Road_Type", "Hr")\
.count().sort(f.desc("count"))
```

```
VehProb7filtered = Veh.filter((f.col("Vehicle_Type").isin(2,3,4,5)))
area_by_motorcycle = Acc.filter((f.col("Accident_Severity")=="1"))\
.join(VehProb7filtered, Acc.Accident_Index == VehProb7filtered.Accident_Index)
    \
.groupBy("Local_Authority_(District)")\
.count().sort(f.desc("Count"))
```

Listing 3: Hive Queries

```
## Problem 1
SELECT police_force,
COUNT(*)/2 AS Fatal_Accidents
FROM accidents WHERE accident_severity = 1
GROUP BY police_force;

## Problem 2

SELECT local_authority_district,
COUNT(*) AS fatal_accidents
FROM accidents WHERE accident_severity = 1
GROUP BY local_authority_district
ORDER BY fatal_accidents DESC LIMIT 5;

## Problem 3
SELECT accident_severity, vehicle_type,
COUNT(accident_severity)/2 AS Number_of_fatal_accidents
FROM bdttvehicles, accidents
WHERE accidents.accident_index=bdttvehicles.accident_index
AND accident_severity = 1 AND vehicle_type != -1
GROUP BY accident_severity, vehicle_type
ORDER BY Number_of_fatal_accidents DESC;

## Problem 4

SELECT age_band_of_casualty, casualty_severity,
COUNT( casualty_severity)/4 AS number_of_fatalities
FROM casualties, accidents
WHERE casualties.accident_index=accidents.accident_index
AND age_band_of_casualty !=-1 AND casualty_severity=1
GROUP BY age_band_of_casualty, casualty_severity
ORDER BY number_of_fatalities DESC;

## Problem 5

SELECT accident_severity, police_force, day_of_week,
COUNT(day_of_week)/2 AS Total_accidents_on_day
FROM accidents WHERE accident_severity = 3 AND police_force = 13
GROUP BY accident_severity, police_force, day_of_week
ORDER BY Total_accidents_on_day DESC;

## Problem 6
CREATE TEMP TABLE problem6hours IF NOT EXIST
SELECT *
substr(Time, ":") AS ("Hours", "Minutes")
FROM Acc;
```

```
SELECT accident_severity, hours, road_type,
COUNT(accident_severity)/2 AS Number_of_accidents
FROM problem6hours WHERE accident_severity = 1 AND road_type = 3
GROUP BY accident_severity, hours, road_type
ORDER BY Number_of_accidents DESC;

## Problem 7

SELECT local_authority_district,
COUNT(accident_severity)/2 AS number_of_accidents
FROM accidents, bdttvehicles
WHERE accident_severity = 1 AND vehicle_type >1 AND vehicle_type <6
AND bdttvehicles.accident_index=accidents.accident_index
GROUP BY local_authority_district
ORDER BY number_of_accidents DESC;
```