UK Road Safety:
Traffic Accidents and Vehicles

THE TRIO

Devikartik Illendula

Venkata Sri Athulya Gopishetty

Shivaramteja Keerthi

REPORT 4 (FINAL REPORT)

For the project, this semester our team has decided to consider Road safety as the focus area. The dataset we have taken is Road Safety data in United Kingdom from the years 2010-2014. The data has come from the Open Data website of the UK government, where they have been published by the Department of Transport. There are two datasets where one is related to accidents in United Kingdom and the other is related to vehicles used in the respective accidents. There are many attributes contributing to the data. Some of the attributes are:

- Accident_Index
- Accident_Severity
- Date
- Daytime
- Day_of_Week
- Junction_Detail
- Latitude
- Longitude
- Lights
- Number_of_Casualties
- Number_of_Vehicles
- Road_Surface_Conditions
- Region
- Road_Type
- Speed_limit
- Urban_or_Rural_Area
- Weather
- High_Wind
- Year
- Age_Band_of_Driver
- Age_of_Vehicle
- Sex_of_Driver
- Vehicle_Category

The Accidents dataset consists of data from the years 2010-2014 and is very huge comprising of 672 Mega Bytes of Memory. The vehicles dataset consists of vehicles data from the years 2005-17 and comprises around 270 Mega Bytes of Memory. There are missing values and outliers in both these datasets. Most of the attributes in these datasets are non-numeric.

Questions likely to be answered at the end of analysis:

- Which day of the week has more accidents?
- Which region has the highest accident rate?
- What kind of weather conditions result in more accidents?
- Do road surface conditions have any effect on the number of accidents?
- What is the effect of speed limit on accidents?

- What is the severity of accidents in United Kingdom?
- What are the age groups of drivers who tend to do more accidents?
- How is the accident rate in different types of junctions?
- What are the stats of accidents over the years based on gender?
- How are the trends in accidents based on geographical areas (rural/urban) over the years?
- How do the different vehicle types affect accidents?
- How is Accidents rate? Increasing or decreasing from 2010-2014?

Initial Hypothesis:

H0: Accidents happen more in the dark conditions.
H1: Accidents happen less in the dark conditions.

H0: Accidents happen more in weekends.
H1: Accidents happen less in weekends.

Data Preparation:

There are two CSV files namely Accident_Information.csv (contains data about the accident, for example, the location, severity, road conditions, etc.) and Vehicle_information.csv (contains data about the vehicle, for example, the vehicle type, the fuel type, driver information, age of vehicle, etc.).

The dataset is very huge (Accident_Information.csv with over 2 million records and Vehicle_Information.csv with over 1.4 million records), and we were constantly getting memory errors while trying to do any operations on the dataset. So, we reduced the size of the dataset by considering accidents which took place between the years 2010 and 2014. Also, some of the columns had missing information in the following forms:

- Unknown
- Data missing or out of range
- None
- Blanks

Data transformation and Data pre-processing:
Data pre-processing has the major part in our project. Values in many columns are limited to certain word dictionaries for making the analysis easy.

```
# data pre-processing
# removing unwanted column values in Weather_Conditions,Light_Conditions
# renaming few columns
def label_weather(row):
    if row['Weather_Conditions'] == 'Fine + high winds':
        return 'Fine'
    elif row['Weather_Conditions'] == 'Fine no high winds':
        return 'Fine'
    elif row['Weather_Conditions'] == 'Raining + high winds':
        return 'Raining'
    elif row['Weather_Conditions'] == 'Raining no high winds':
        return 'Raining'
    elif row['Weather_Conditions'] == 'Snowing + high winds':
        return 'Snowing'
    elif row['Weather_Conditions'] == 'Snowing no high winds':
        return 'Snowing'
    else:
        return row['Weather_Conditions']
accidents['Weather'] = accidents.apply (lambda row: label_weather(row), axis=1)
```

```
def label_high_wind(row):
    if row['Weather_Conditions'] == 'Fine + high winds':
        return 'Yes'
    elif row['Weather_Conditions'] == 'Fine no high winds':
        return 'No'
    elif row['Weather_Conditions'] == 'Raining + high winds':
        return 'Yes'
    elif row['Weather_Conditions'] == 'Raining no high winds':
        return 'No'
    elif row['Weather_Conditions'] == 'Snowing + high winds':
        return 'Yes'
    elif row['Weather_Conditions'] == 'Snowing no high winds':
        return 'No'
    else:
        return 'No'
accidents['High_Wind'] = accidents.apply (lambda row: label_high_wind(row), axis=1)
accidents.drop(['Weather_Conditions'], inplace=True, axis=1)

def label_lights(row):
    if row['Light_Conditions'] == 'Darkness - lights lit':
        return 'Darkness - lights'
    elif row['Light_Conditions'] == 'Darkness - lights unlit':
        return 'Darkness - no lights'
    elif row['Light_Conditions'] == 'Darkness - no lighting':
        return 'Darkness - no lights'
    else:
        return row['Light_Conditions']
accidents['Lights'] = accidents.apply (lambda row: label_lights(row), axis=1)
accidents.drop(['Light_Conditions'], inplace=True, axis=1)
```

```
def label_region(row):
    if row['Local_Authority_(District)'] in east_midlands:
        return 'East Midlands'
    elif row['Local_Authority_(District)'] in east_england:
        return 'East England'
    elif row['Local_Authority_(District)'] in london:
        return 'London'
    elif row['Local_Authority_(District)'] in north_east_england:
        return 'North East England'
    elif row['Local_Authority_(District)'] in north_west_england:
        return 'North West England'
    elif row['Local_Authority_(District)'] in scotland:
        return 'Scotland'
    elif row['Local_Authority_(District)'] in south_east_england:
        return 'South East England'
    elif row['Local_Authority_(District)'] in south_west_england:
        return 'South West England'
    elif row['Local_Authority_(District)'] in wales:
        return 'Wales'
    elif row['Local_Authority_(District)'] in west_midlands:
        return 'Wast Midlands'
    elif row['Local_Authority_(District)'] in yorkshire_and_the_humber:
        return 'Yorkshire and the Humber'
accidents['Region'] = accidents.apply (lambda row: label_region(row), axis=1)
accidents.drop(['Local_Authority_(District)'], inplace=True, axis=1)
```

```
def label_vehicle_category(row):
    if row['Vehicle_Type'] == 'Bus or coach (17 or more pass seats)':
        return 'Bus/minibus'
    elif row['Vehicle_Type'] == 'Minibus (8 - 16 passenger seats)':
        return 'Bus/minibus'
    elif row['Vehicle_Type'] == 'Taxi/Private hire car':
        return 'Taxi'
    elif row['Vehicle_Type'] == 'Van / Goods 3.5 tonnes mgw or under':
        return 'Van'
    elif row['Vehicle_Type'] == 'Motorcycle 125cc and under':
        return 'Motorcycle'
    elif row['Vehicle_Type'] == 'Motorcycle 50cc and under':
        return 'Motorcycle'
    elif row['Vehicle_Type'] == 'Motorcycle over 125cc and up to 500cc':
        return 'Motorcycle'
    elif row['Vehicle_Type'] == 'Motorcycle over 500cc':
        return 'Motorcycle'
    elif row['Vehicle_Type'] == 'Motorcycle - unknown cc':
        return 'Motorcycle'
    elif row['Vehicle_Type'] == 'Agricultural vehicle':
        return 'Other'
    elif row['Vehicle_Type'] == 'Electric motorcycle':
        return 'Other'
    elif row['Vehicle_Type'] == 'Goods 7.5 tonnes mgw and over':
        return 'Other'
    elif row['Vehicle_Type'] == 'Goods over 3.5t. and under 7.5t':
        return 'Other'
    elif row['Vehicle_Type'] == 'Goods vehicle - unknown weight':
        return 'Other'
    elif row['Vehicle_Type'] == 'Other vehicle':
        return 'Other'
    elif row['Vehicle_Type'] == 'Pedal cycle':
        return 'Other'
    elif row['Vehicle_Type'] == 'Ridden horse':
        return 'Other'
    elif row['Vehicle_Type'] == 'Tram':
        return 'Other'
    elif row['Vehicle_Type'] == 'Data missing or out of range':
        return 'Other'
    elif row['Vehicle_Type'] == 'Mobility scooter':
        return 'Other'
    else:
        return row['Vehicle_Type']
vehicles['Vehicle_Category'] = vehicles.apply (lambda row: label_vehicle_category(row), axis=1)
vehicles.drop(['Vehicle_Type'], inplace=True, axis=1)
```

A new Column named Weekend is created in the accidents dataset to classify the difference between a weekday and a weekend.

```
# creating a new column Weekend based on Day of Week values
# Weekend - Yes / Not Weekend - No
def label_day_of_week(row):
    if row['Day_of_Week'] == 'Monday':
        return "No"
    elif row['Day_of_Week'] == 'Tuesday':
        return "No"
    elif row['Day_of_Week'] == 'Wednesday':
        return "No"
    elif row['Day_of_Week'] == 'Thursday':
        return "No"
    elif row['Day_of_Week'] == 'Friday':
        return "No"
    elif row['Day_of_Week'] == 'Saturday':
        return "Yes"
    elif row['Day_of_Week'] == 'Sunday':
        return "Yes"

accidents['Weekend'] = accidents.apply (lambda row: label_day_of_week(row), axis=1)
```

In Data pre-processing missing values are removed from the accidents data set.

```
# removing missing values and unwanted data from accidents_chunck
for chunk in accidents_chunk:
    chunk_filter = chunk[
        (chunk.Year.astype(int) >= 2010) &
        (chunk.Year.astype(int) <= 2014) &
        (chunk['Road_Type'] != "Unknown") &
        (chunk['Junction_Detail'] != "Data missing or out of range") &
        (chunk['Road_Surface_Conditions'] != "Data missing or out of range") &
        (chunk['Weather_Conditions'] != "Data missing or out of range") &
        (chunk['Latitude'].notnull()) &
        (chunk['Longitude'].notnull()) &
        (chunk['Special_Conditions_at_Site'] == "None")
    ]
    chunk_list.append(chunk_filter)
accidents = pd.concat(chunk_list)
```

Similarly, missing values are removed from the Vehicles dataset.

```
# removing missing values out of vehicles dataset
vehicles_uninteresting = vehicles[
    (vehicles['Age_Band_of_Driver'] == 'Data missing or out of range') |
    (vehicles['Age_Band_of_Driver'] == '0 - 5') |
    (vehicles['Age_Band_of_Driver'] == '6 - 10') |
    (vehicles['Age_Band_of_Driver'] == '11 - 15') |
    (vehicles['Age_of_Vehicle'].isnull()) |
    (vehicles['Engine_Capacity_.CC.'].isnull()) |
    (vehicles['make'].isnull()) |
    (vehicles['Sex_of_Driver'] == 'Not known') |
    (vehicles['Sex_of_Driver'] == 'Data missing or out of range') |
    (vehicles['Vehicle_Manoeuvre'] == 'Data missing or out of range')]
accidents_uninteresting_list = vehicles_uninteresting['Accident_Index'].tolist()
vehicles = vehicles[~vehicles['Accident_Index'].isin(accidents_uninteresting_list)]
```

In Data transformation the two datasets are merged into a single dataset named Mergedfile.csv for flexible analytics purpose. Both the datasets are merged based on a common attribute Accident_Index.

```
# combine the accidents with the vehicles table
accidents_data = pd.merge(accidents, vehicles, on='Accident_Index')
accidents_data.to_csv('Mergedfile.csv')
```

Few columns are removed from both accidents and vehicles datasets since they are irrelevant to the analysis for example InScotland, Was_vehicle_left_hand_drive etc.

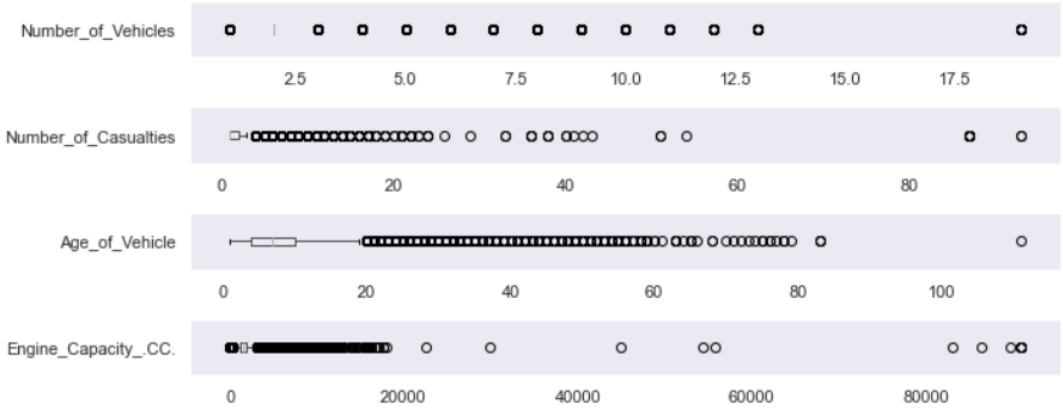
```
# drop old time column and temporary hour column
accidents = accidents.drop(columns=['Time', 'Hour'])

# drop columns we don't need
accidents = accidents.drop(columns=['1st_Road_Class','1st_Road_Number','2nd_Road_Class', '2nd_Road_Number',
                                    'Did_Police_Officer_Attend_Scene_of_Accident',
                                    'Local_Authority_(Highway)','Location_Easting_OSGR', 'Location_Northing_OSGR',
                                    'LSOA_of_Accident_Location','Pedestrian_Crossing-Human_Control',
                                    'Pedestrian_Crossing-Physical_Facilities','Police_Force','InScotland'])
```

```
# drop columns we don't need
vehicles = vehicles.drop(columns = ['Driver_IMD_Decile','Junction_Location','model','Skidding_and_Overturning',
                                    'Towing_and_Articulation','Vehicle_Leaving_Carriageway',
                                    'Vehicle_Location.Restricted_Lane','Vehicle_Reference',
                                    'Was_Vehicle_Left_Hand_Drive','X1st_Point_of_Impact','Year'])
```

The final list of the columns is as following:

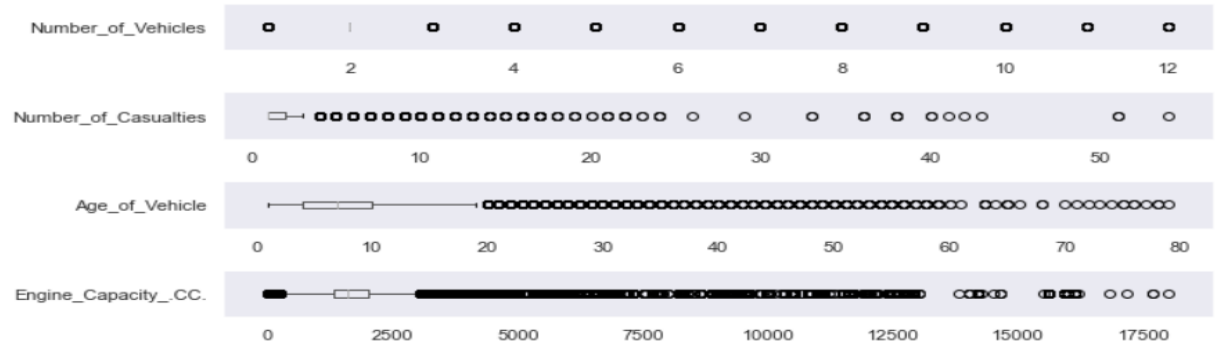
```
Accident_Index
Accident_Severity
Carriageway_Hazards
Date
Daytime
Day_of_Week
Junction_Detail
Latitude
Longitude
Lights
Number_of_Casualties
Number_of_Vehicles
Road_Surface_Conditions
Region
Road_Type
Speed_limit
Urban_or_Rural_Area
Weather
High_Wind
Year
Age_Band_of_Driver
Age_of_Vehicle
Engine_Capacity_.CC.
Journey_Purpose_of_Driver
Sex_of_Driver
Vehicle_Manoeuvre
Vehicle_Category
Weekend
```

Four columns Number_of_Vehicles, Number_of_Casualties, Age_of_Vehicle, Engine_Capacity_.CC. have outliers.



Outliers are eliminated and the data is settled within the range.

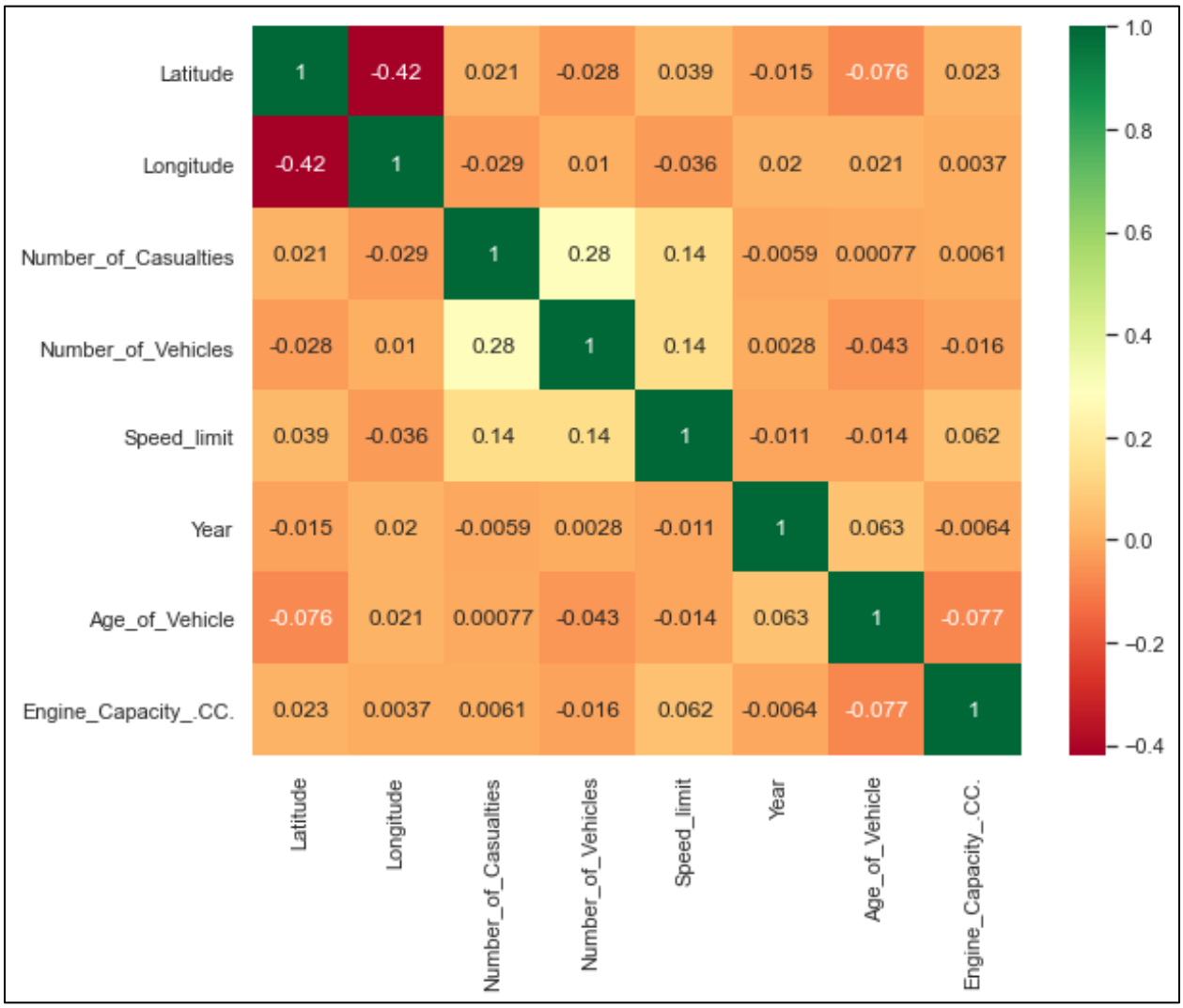
```
# keep only records that meet the condition and don't fall within extreme outliers
accidents_data = accidents_data[(accidents_data['Engine_Capacity_.CC.'] < 20000)&
             (accidents_data['Age_of_Vehicle'] < 80) &
             (accidents_data['Number_of_Casualties'] < 60) &
             (accidents_data['Number_of_Vehicles'] < 13.0)]
accidents_data.shape
sns.set(style='darkgrid')
fig, axes = plt.subplots(4,1, figsize=(10,4))

for ax, col in zip(axes, num_cols):
    accidents_data.boxplot(column=col, grid=False, vert=False, ax=ax)
    plt.tight_layout();
```



Now data is transformed, cleaned and is ready for further analysis (Exploratory analysis followed by Visualizations and Data Modeling).

Exploratory Data Analysis:

1) Correlation Plot:



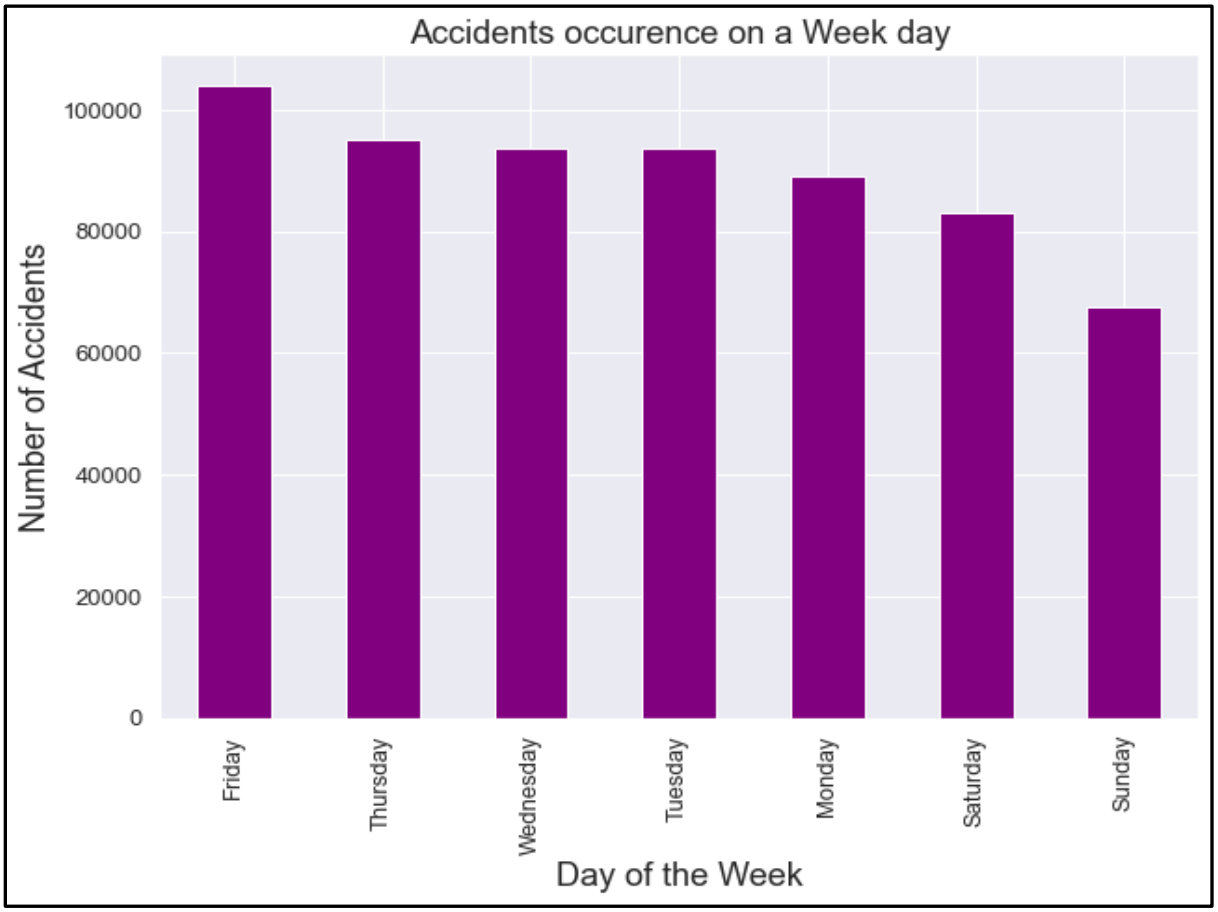
The above heat map shows the correlation plot for the merged dataset. There are no significantly correlated attributes since most of them are weakly linked together. It would be hard to implement an unsupervised learning model like Clustering.

2) Data Visualizations:

After performing Data processing and transforming, the major questions of the project are answered by the visualizations.

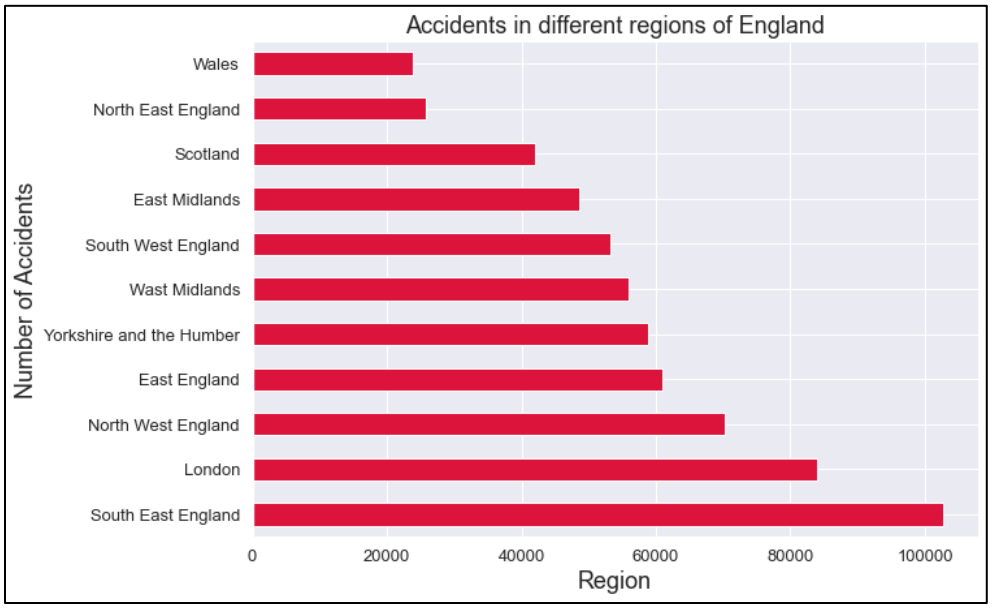
Questions answered based on visualizations:

- **Which day of the week has more accidents?**



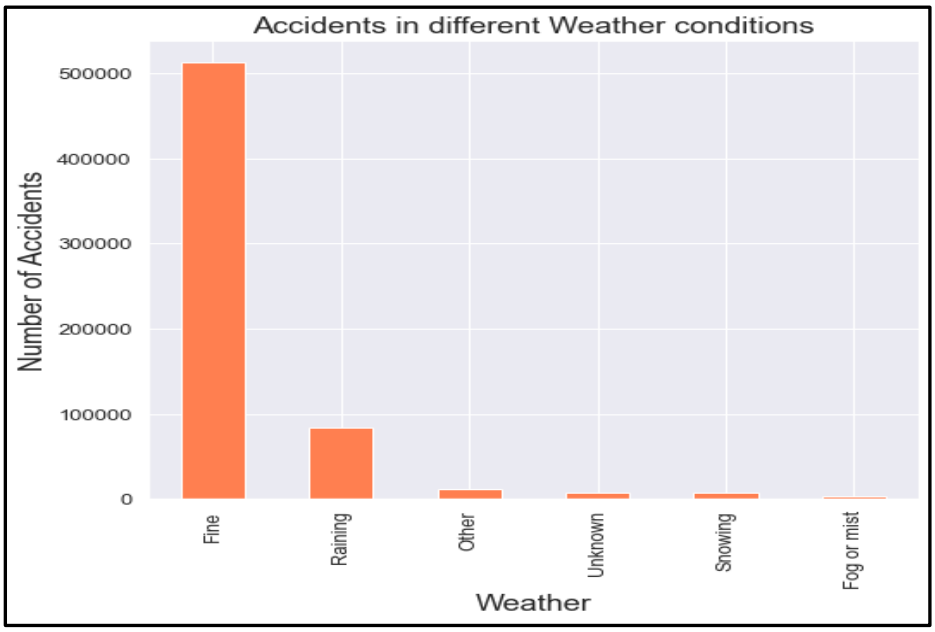
A bar chart is chosen to represent Accidents count on different days of a week. Friday, amongst all the days in a week has the highest amount of accidents recorded, followed by Thursday, Wednesday, Tuesday, Monday, Saturday, Sunday, respectively.

- **Which region has the highest accident rate?**



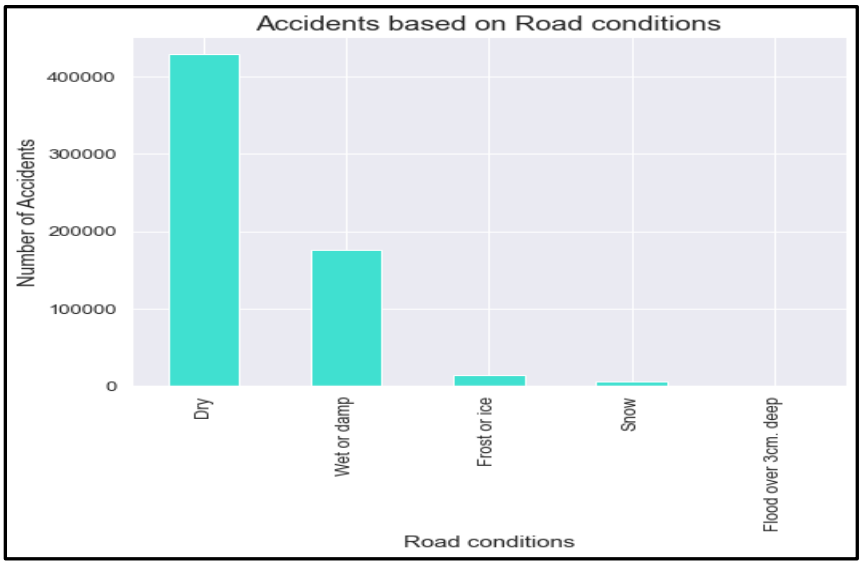
A horizontal bar chart is used to represent Accidents in different region of England. It looks like South East England has the highest accident rate followed by the country's capital London.

- **What kind of weather conditions result in more accidents?**



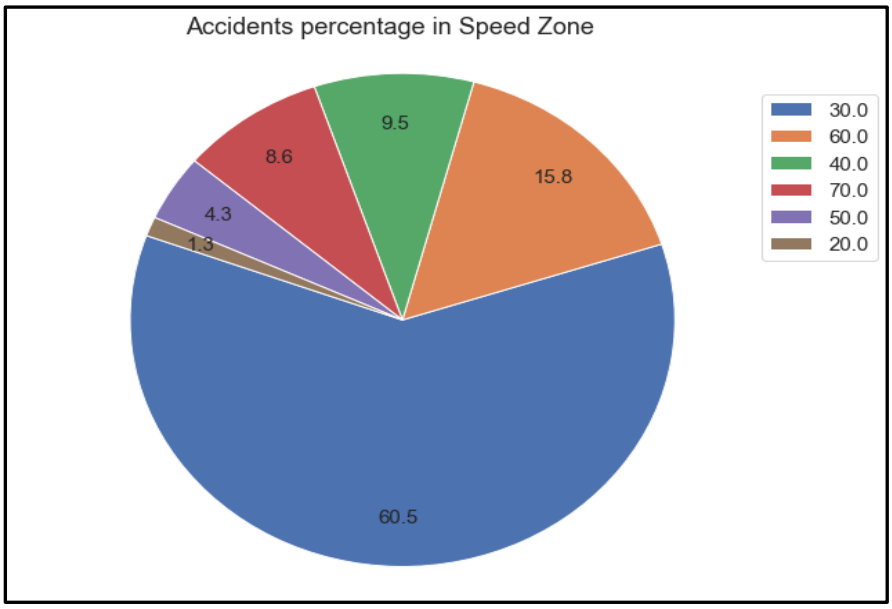
According to the above bar chart, weather conditions do not affect the accident rate much. Most of the accidents happen more often under normal(fine) weather conditions in agreement with the data. Rainfall affects the accidents rate a bit compared to other weather conditions which are almost insignificant.

- **Do road surface conditions have any effect on the number of accidents?**



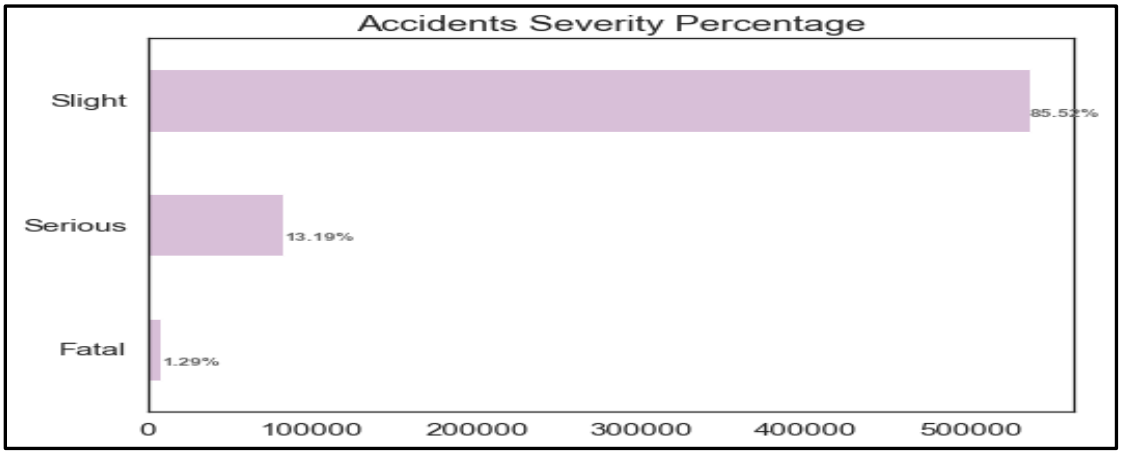
Above visualization removes the presumption that damp roads cause accidents. According to the data, Dry road conditions have more probability of accident occurrence than any other road surface condition. Accidents also happen in frost roads but are in insignificant amount.

- **What is the effect of speed limit on accidents?**



This is an interesting plot that disrupts our misconception on Speed limit effect on the accidents. In general, there is a banal ideology related to Speed limit, the higher the limit, more is the probability of crashing while driving. But according to our pie chart, accidents happen more within the Speed limit of 30mph rather than 60 or 70 mph, 60% of accident occurrence is under speed limit having 30mph.

- **What is the severity of accidents in United Kingdom?**

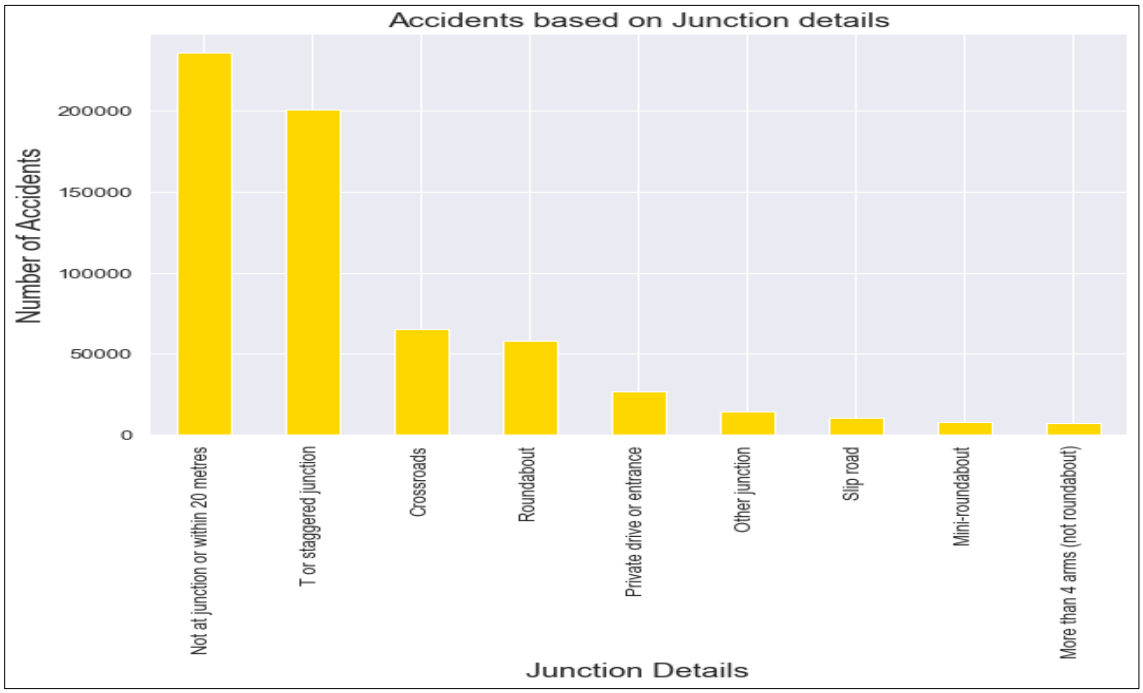


Accident Severity is depicted using a horizontal bar chart. Having slightly severe accident is more probable and takes around 85.52% share in severity section followed by serious (13.19%) and fatal (1.29%).

- **What are the age groups of drivers who tend to do more accidents?**

This part of visualization concurs with the general notion of young drivers are involved in accidents. According to the plot, drivers whose age group is between 26-35 have done more accidents than any other group. Surprisingly followed by older and responsible age bands (36-45) and (46-55).

- **How is the accident rate in different types of junctions?**

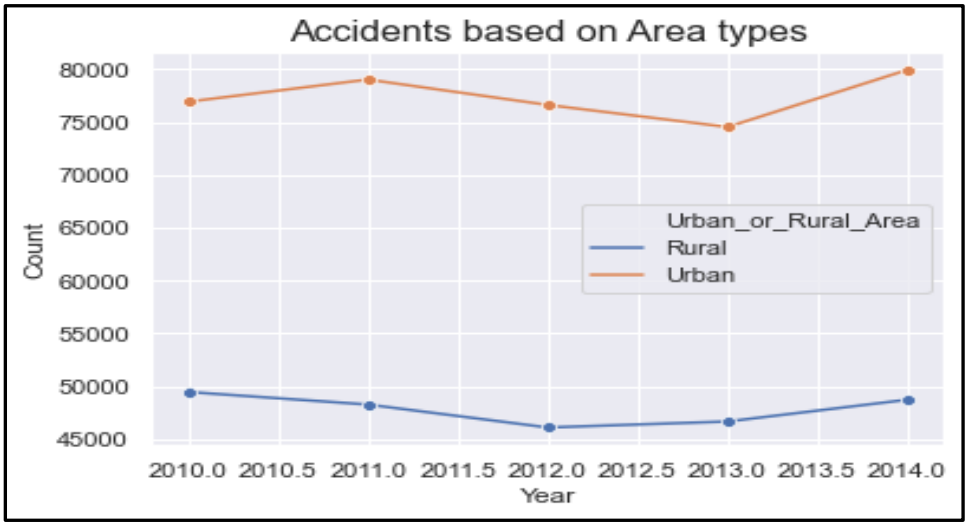


Most of the accidents occur in the absence of a junction T or staggered junction the second highest spot where a greater number of accidents occurred.

- **What are the stats of accidents over the years based on gender?**

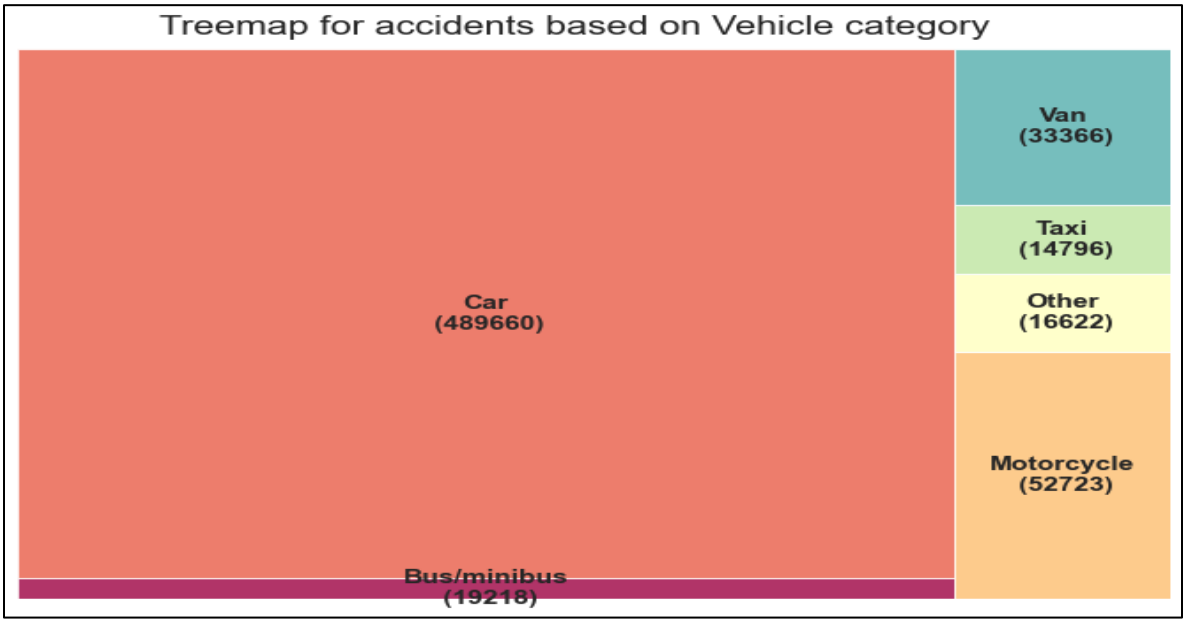
Above picture depicts Males consistency to be involved in accidents has not changed much over the years from 2010-2014 in England. Whereas, females have similar distribution of accidents occurrence over the years. It is obvious that Male drivers are more involved in the accidents than Female drivers.

- **How are the trends in accidents based on geographical areas (rural/urban) over the years?**



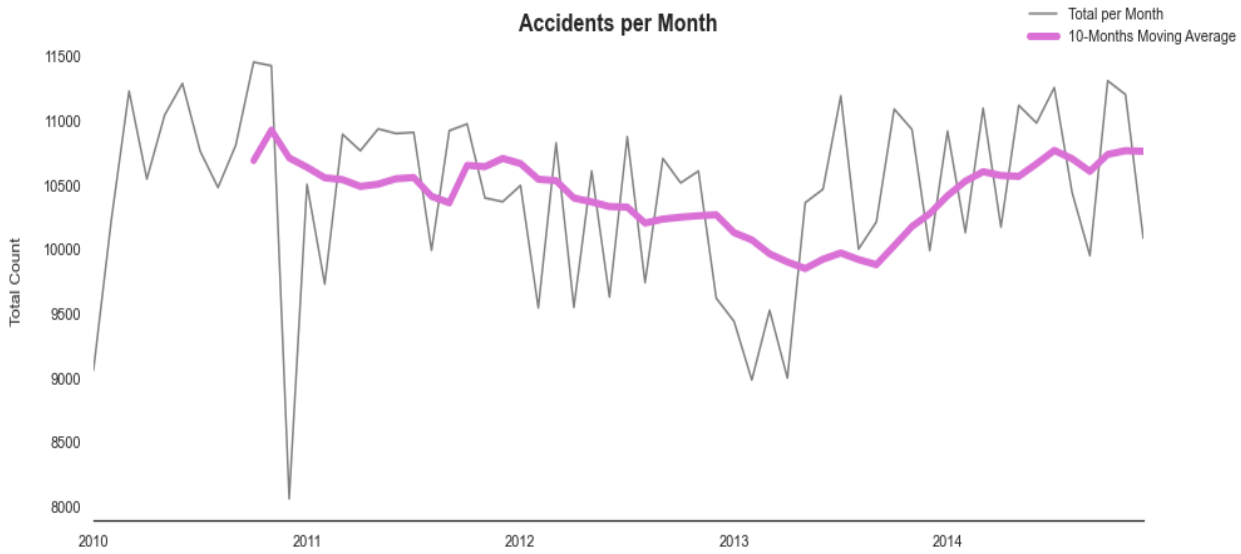
From the above graph, it is evident that Urban areas are vulnerable to the accidents than Rural areas. The trends are almost consistent over the years having insignificant level of highs and lows.

- **How do the different vehicle types affect the accidents?**



A tree map is used to reveal the most accident affecting vehicle category. It looks like Cars are tangled with the most accidents followed by Motorcycle, Van, Bus, Taxi and Other category.

- **How is Accidents rate? Increasing or decreasing from 2010-2014 ?**

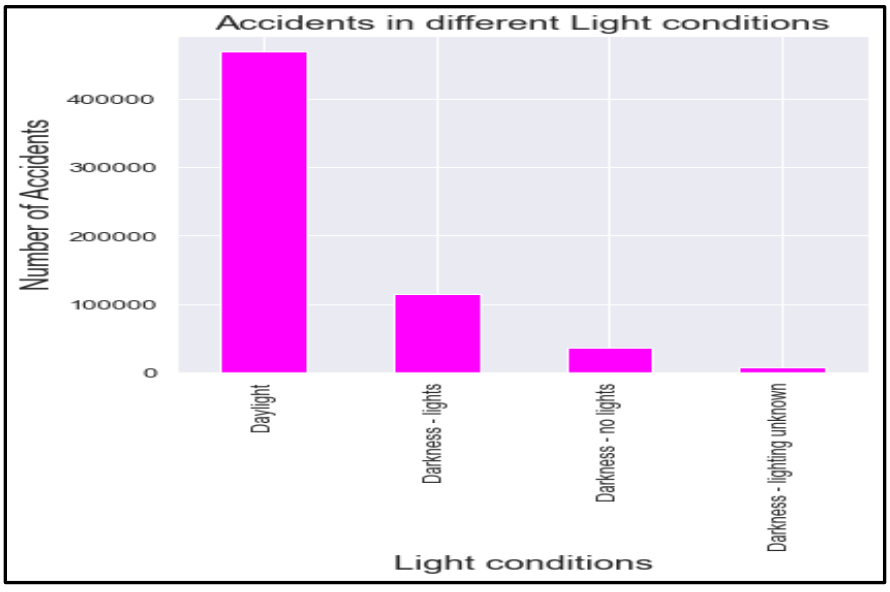


Above plot shows accidents per month for years 2010-2014 with a 10-months moving average. Accidents seems almost constant with little fluctuations around 2013-2014. The tip of moving average shows will be increase in accidents rate for the coming year.

3) Hypothesis:

In our project Hypothesis are demonstrated using visualizations. They are supported or rejected based on their respective plots.

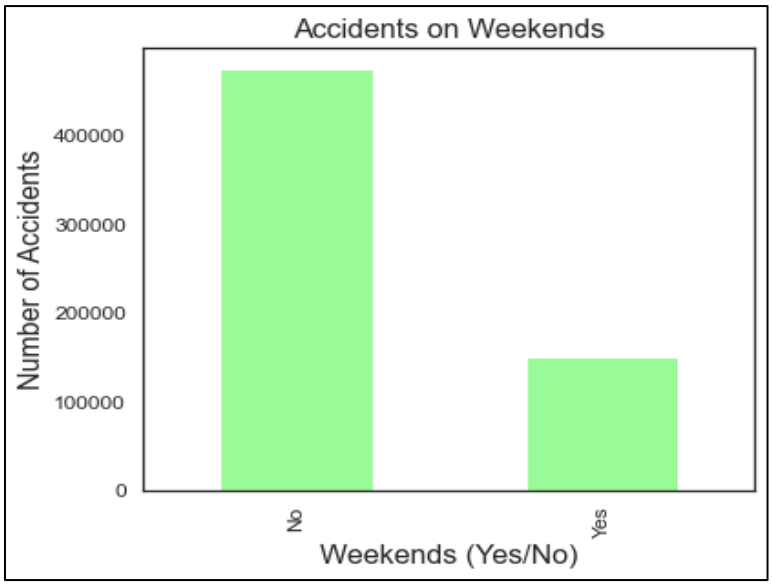
- H_0: Accidents happen more in the dark conditions.
  H_1: Accidents happen less in the dark conditions.



Our null hypothesis for this part is that Accidents happen more in the dark conditions and the alternative one is Accidents happen less in the dark conditions.

We can see from the above bar chart that accidents tend to occur frequently in daylight rather than in dark conditions. More than 400000 accidents happen in day light and no other factor come this peak so far. According to the chart, the **null hypothesis (H0) is rejected** as the accidents happen often in less dark conditions.

- H_0: Accidents happen more in weekends.
  H_1: Accidents happen less in weekends.



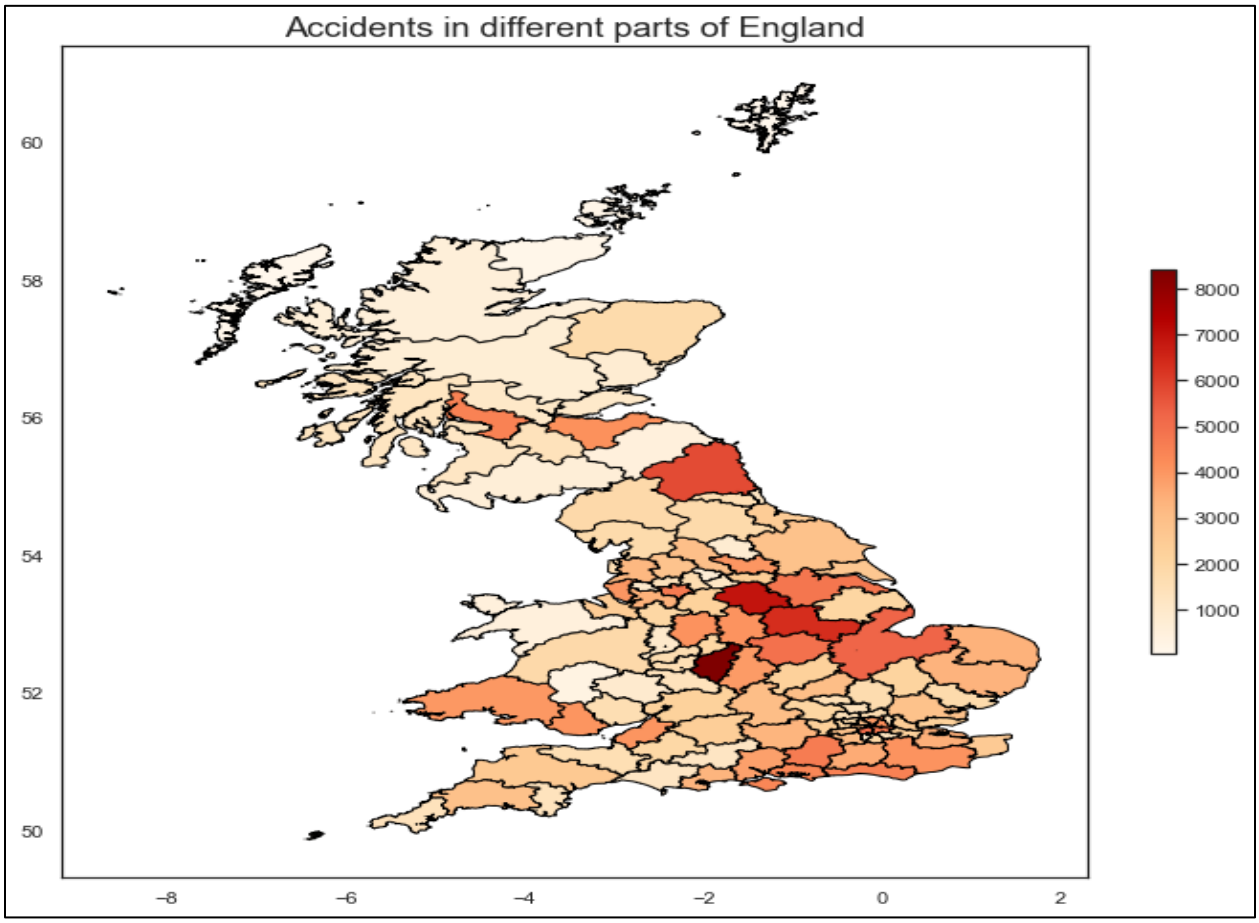
Our null hypothesis for this part is that Accidents happen more in weekends and the alternative one is Accidents happen less in the dark conditions.

To demonstrate this hypothesis, a new column named Weekend is created to separate the weekends and weekdays with a binary kind of response.

We can see from the above bar chart that accidents tend to occur often on weekdays which count more than 400000 altogether. According to the chart, the **null hypothesis (H0) is rejected** as the accidents occur happen less in the weekends.

Overall Accident rate across England for years 2010 -2014

This is visualization created in python using Shapely and Geopandas libraries along with UK post codes boundaries file. It simply counts the accidents across different geographical locations spotted with the help of Latitude and Longitude features and rightly configured using a shape file reader for post codes files.



It indicates that the central and southern parts of England are vulnerable to accidents than the northern parts.

4) Data Modeling:

Since the dataset is not suitable for Clustering algorithms, a Random forest is selected to predict Number of causalities in Accidents with different independent features like Accident_Severity, Dayof_week, Daytime, Road_Type, Speed_limit, Urban_or_Rural_Area which kind of correlated in a significant way than other attributes.

In the first step, the target variable is picked and named. Categorical and numerical variables are taken into a column.

```
# define numerical feature column
num_col = ['Number_of_Vehicles']

# define categorical feature columns
cat_cols = ['Accident_Severity', 'Day_of_Week', 'Daytime', 'Road_Type', 'Speed_limit',
            'Urban_or_Rural_Area']

# define target column
target_col = ['Number_of_Casualties']

cols = cat_cols + num_cols + target_col

# copy dataframe
df_model = accidents_data[cols].copy()
df_model.shape
```

Categorical variables are changed into dummy variables and are then concatenated to the main data frame df_model.

```
# create dummy variables from the categorical features
dummies = pd.get_dummies(df_model[cat_cols], drop_first=True)
df_model = pd.concat([df_model[num_cols], df_model[target_col], dummies], axis=1)
df_model.shape
df_model.isna().sum().sum()
```

Here the independent columns are taken into features variable and the dependent column, Number of Causalities are taken into target variable. Sklearn library is used to split the variables into train and test data in the ratio 80:20 respectively.

Our notion is that the model evaluation metric better suited to imbalanced classes: confusion matrices, precision, recall, F1 scores, or ROC curves instead of accuracy.

```
# define our features
features = df_model.drop(['Number_of_Casualties'], axis=1)

# define our target
target = df_model[['Number_of_Casualties']]

from sklearn.model_selection import train_test_split

# split our data
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2)

# import regressor
from sklearn.ensemble import RandomForestRegressor

# import metrics
from sklearn.metrics import mean_squared_error, r2_score

# import evaluation tools
from sklearn.model_selection import RandomizedSearchCV

# create RandomForestRegressor
forest = RandomForestRegressor(random_state=4, n_jobs=-1)

# train
forest.fit(X_train, y_train)

# predict
y_train_preds = forest.predict(X_train)
y_test_preds  = forest.predict(X_test)

# evaluate
RMSE = np.sqrt(mean_squared_error(y_test, y_test_preds))
print(f"RMSE: {round(RMSE, 4)}")

r2 = r2_score(y_test, y_test_preds)
print(f"r2: {round(r2, 4)}")

# look at parameters used by our current forest
print('Parameters currently in use:\n')
print(forest.get_params())
```

Initial results:

```
RMSE: 0.9255
r2: -0.0443
Parameters currently in use:

{'bootstrap': True, 'ccp_alpha': 0.0, 'criterion': 'mse', 'max_depth': None, 'max_features': 'auto', 'max_leaf_nodes': None,
'max_samples': None, 'min_impurity_decrease': 0.0, 'min_impurity_split': None, 'min_samples_leaf': 1, 'min_samples_split':
2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100, 'n_jobs': -1, 'oob_score': False, 'random_state': 4, 'verbose': 0,
'warm_start': False}
```

Our next step was to find out best hyper parameters to use them in next model fit.

```
# create range of candidate numbers of trees in random forest
n_estimators = [100, 150]

# create range of candidate max. numbers of levels in tree
max_depth = [3, 4, 5]

# create range of candidate min. numbers of samples required to split a node
min_samples_split = [10, 15, 20]

# create dictionary with hyperparameter options
hyperparameters = dict(n_estimators=n_estimators, max_depth=max_depth, min_samples_split=min_samples_split)
hyperparameters
```

From the above result the best parameters have n_estimators = 50, max_depth = 5, min_split = 20. Out of them few are used for the next model fit.

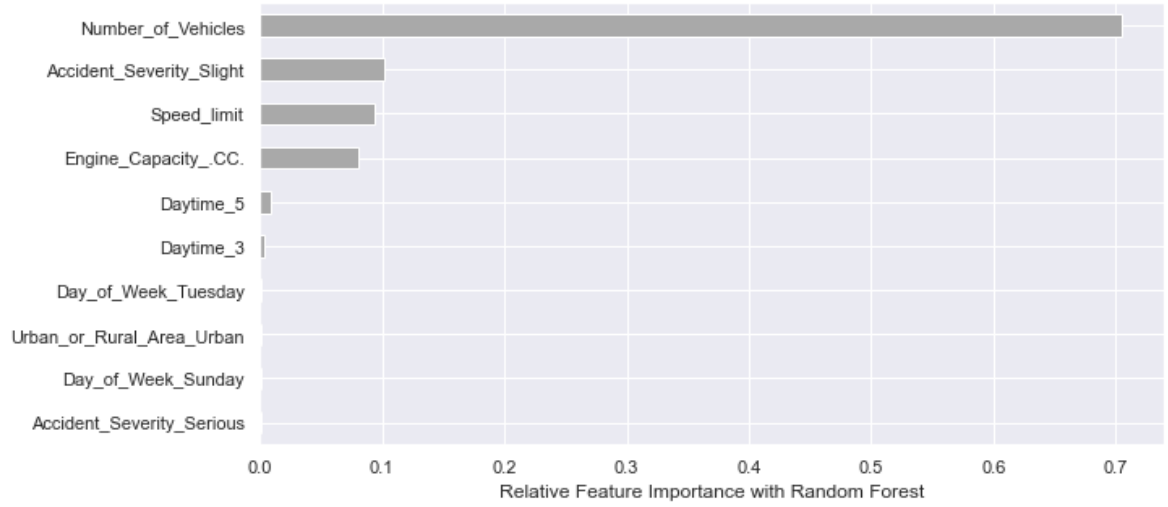
```
# create RandomForestRegressor with best found hyperparameters
forest = RandomForestRegressor(n_estimators=150, max_depth=5, random_state=4, n_jobs=-1)

# train
forest.fit(X_train, y_train)

# predict
y_train_preds = forest.predict(X_train)
y_test_preds  = forest.predict(X_test)

# evaluate
RMSE = np.sqrt(mean_squared_error(y_test, y_test_preds))
print(f"RMSE: {round(RMSE, 4)}")

r2 = r2_score(y_test, y_test_preds)
print(f"r2: {round(r2, 4)}")
```

Result for the second model fit:

```
RMSE: 0.8498
r2: 0.1195
```

Using the model with hyper parameters decreased the root mean squared error (RMSE) and increased the variability r^2.

Variable importance plot:



It shows that Number of Vehicles has highest involvement in the output followed by Slight value in Accident severity, Speed limit, Engine capacity, Daytime_5(hour >= 5 and hour < 10).

Overall approach to your analysis effort:

We chose road safety as our focus area and took the dataset of accidents in UK. We initially had 2 different datasets, one containing information on accidents and another dataset contains information on vehicles. There were many columns which were not needed for our analysis. We deleted all those unnecessary columns. Then we merged both the dataset files (csv files) using a common attribute (i.e., accident index). Also, we extracted a portion of dataset from this merged dataset because of memory limitations in our laptops. Then we used this final extracted dataset for our analysis.

We did not have perfect analysis process, yet we yearned to complete the project successfully and we did.

Our analysis includes: 1) Data preprocessing 2) Hypothesis demonstration using Visualizations 3) Searching answers for the research questions 4) Predictive modelling to estimate Number of Causalities using best related columns 5) Document the analysis done.

Analysis Effectiveness:

Our analysis has gone through major evolutions in each step. At first, we used Tableau for visualizations and later on we switched to python to meet the standards. The more we went into analysis, the more we understood the data, modeling and representation. It was a great process of learning. We can count using Tableau for visualizations as an additional effort.
We can say our analysis was successful as we were able to create visualizations of various attributes which helped achieving our all required answers to the initial set of questions and hypothesis.

Lessons learnt from the project:

- We tried implementing K-means Clustering but the algorithm does not accept such a huge dataset. We do not know if there is any other alternative method, but every method that was tried did not give output. So, we came back to supervised learning algorithm process. The lesson learnt here is clustering not always works for large datasets. Large dataset slowed down the system completely. To make our task easy the dataset from 2005-17 has been reduced to data between years 2010 and 2014.
- We found that even a small plot or visualization can help answer a complex question in our mind
- We learnt that a bigger dataset with more attributes is very complex and if we don't remove unnecessary attributes, we could be wasting a lot of time by using the full complete dataset.
- Each time, to improve the project we tried to implement new features which gave us new exposure to the tools like Tableau and libraries like Pytorch.

References

Thanasis. "UK Road Safety: Traffic Accidents and Vehicles." *Kaggle*, 15 Jan. 2019, www.kaggle.com/tsiaras/uk-road-safety-accidents-and-vehicles.