<https://growingdata.com.au/a-guided-introduction-to-exploratory-data-analysis-eda-using-python/>

A guided introduction to

Exploratory Data Analysis (EDA)

using Python

Exploratory Data Analysis (EDA) is the most critical initial step for Data Scientists to analyze a new dataset, this guide describes simple & advanced techniques using Python.

**How do I get started with a Data Science/Machine learning project?**

**What is the depth and breadth of the data?**

**Is the data predictive enough for modeling?**

If you’re looking for answers to the above questions, then this blog is for you. In one word, the answer is Exploratory Data Analysis (**EDA**). So, what is **EDA**?

**EDA** is the process of performing initial investigations on data so as to:

* Uncover underlying structure & patterns in the data
* Identify important variables
* Identify anomalies
* Test a hypothesis
* Check assumptions
* Set the stage for model development

Exploratory Data Analysis is like listening to what the data can tell us before we start the actual modeling process for a head start. The outcome of this analysis is some insights presented by summarised statistics and graphical representations. Also, it is a good practice though to use multiple exploratory techniques to have more confidence in the conclusions reached about the data.

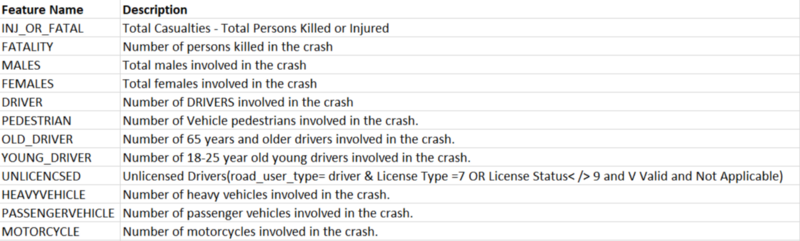
Generally speaking, any method of looking at data that does not include formal statistical modeling and inference may fall under the umbrella of exploratory data analysis. In fact, EDA can be considered the most critical step in analyzing data without which you may end up with less optimal, less interpretable and less accurate models.

#### Sample Dataset

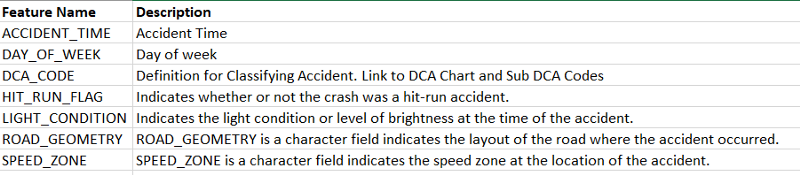
For the purpose of explaining the various EDA techniques, we utilize the Victorian road crashes [dataset](https://vicroadsopendata-vicroadsmaps.opendata.arcgis.com/datasets/c2a69622ebad42e7baaa8167daa72127_0?geometry=103.996%2C-39.619%2C-173.431%2C-33.446). This dataset comprises crash-related information such as the time, location, conditions, crash type, road user type, object hit, etc during the periods between (2010–2015). One reason for car crashes is drinking and driving. Thus, for simplicity, let’s suggest that the target of our analysis here is to predict if a given crash was due to alcohol consumption or not. [Here](http://data.vicroads.vic.gov.au/metadata/Crashes_Last_Five_Years%20-%20Open%20Data.html), you will find the textual descriptions for all fields in the dataset and the data types as well. We choose Python -the most widely acceptable machine learning adopted programming language to perform the analysis.

## **1. Understanding the Dataset**

The first step of EDA is to understand how big the data is? how many features do we have? what is the outcome of our analysis? The crash dataset comprises of 74,908 observations described by 65 features. One is the dependent variable (target class) and the rest 64 are independent variables. The data types available in this dataset are floats (4), integers (27 features) and string (text) (32 features) which can be divided into free text-based features -out of the scope of this post- or categorical features. Numeric features refer to any feature with any form of numeric values (float, integer, double, long, etc). Categorical features refer to variables that can take on one of a limited, and usually fixed number of possible values. The floating values features are all co-ordinates values referring to the crash location. For simplicity, we chose only 12 numeric features and 8 categorical features; in fact, you can perform the analysis on features of choice and you may end up selecting a different set of features. The numeric features selected are described below:

Numeric features description

Below is the description of the selected set of categorical features:

Categorical features description

### **1.1 Import libraries & load dataset**

import pandas as pd

import seaborn as sns

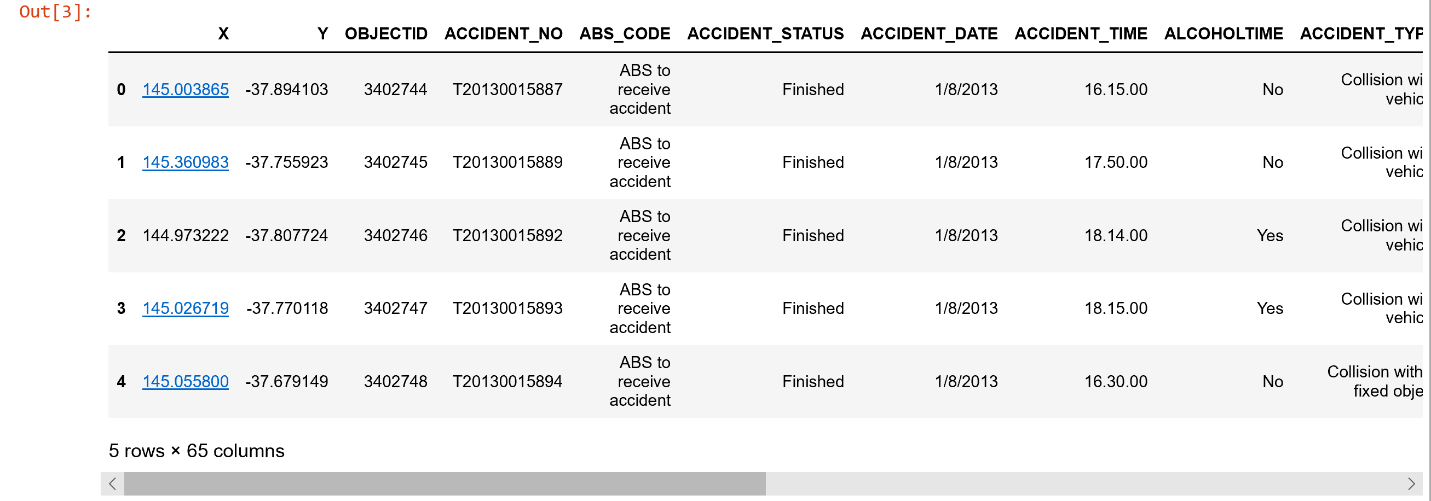
import matplotlib.pyplot as plt

import numpy as np

data = pd.read\_csv("Crashes\_Last\_Five\_Years.csv")

### **1.2 Data sample inspection**

data.head()

Data sample of the first five records

We can extract the numeric and categorical features in separate data frames for easier manipulation and vizualisation.

numeric\_features = data[["INJ\_OR\_FATAL","FATALITY","MALES",

"FEMALES","DRIVER","PEDESTRIAN",

"OLD\_DRIVER","YOUNG\_DRIVER" ,"UNLICENCSED",

"HEAVYVEHICLE","PASSENGERVEHICLE","MOTORCYCLE"

]]

categorical\_features=data[[

"ACCIDENT\_TIME","ACCIDENT\_TYPE","DAY\_OF\_WEEK",

"DCA\_CODE","HIT\_RUN\_FLAG","LIGHT\_CONDITION",

"ROAD\_GEOMETRY","SPEED\_ZONE"

]]

### **1.3 Target class distribution**

data["ALCOHOL\_RELATED"].value\_counts()

#gives each count of the alchohol related classes

data["ALCOHOL\_RELATED"].value\_counts()

#gives each count of the alchohol related classes

No 72429

Yes 2479

Name: ALCOHOL\_RELATED, dtype: int64

We can see that there is huge class imbalance which is a very important observation. This has great influence on how to select instances for training and testing. It is recommended to stratify the data when randomly selecting in order to keep the same distribution of classes in training and testing sets. In some cases techniques like oversampling or undersampling may need to be used to restore class balance for better model training. Moreover, this will even have an effect on choosing the classification model and whether it can deal with class imbalance appropriately or not.

### **1.4 Primary statistics**

Interestingly, you can use the describe function from Pandas to print useful statistics such as count, mean, std, min, max, 25%,50% & 75% percentiles values for each column. For non-numeric features, these values may seem meaningless.

print(data.describe())

It can also be useful if look at these statistics for each class individually for comparison

alchohol\_related\_yes = data[data["ALCOHOL\_RELATED"]=='Yes']

alchohol\_related\_yes.describe()

alchohol\_related\_no = data[data["ALCOHOL\_RELATED"]=='No']

alchohol\_related\_no.describe()

## **2. Univariate Analysis**

The simplest form of statistical analysis in the EDA process is **univariate**analysis. Only one variable is involved in this analysis. In fact, the main purpose of the analysis is to describe, understand the population distribution, detect outliers, summarize and find patterns for a single feature. Note, that this type of analysis is different toward numeric versus categorical features in terms of the characteristics of interest of data values.

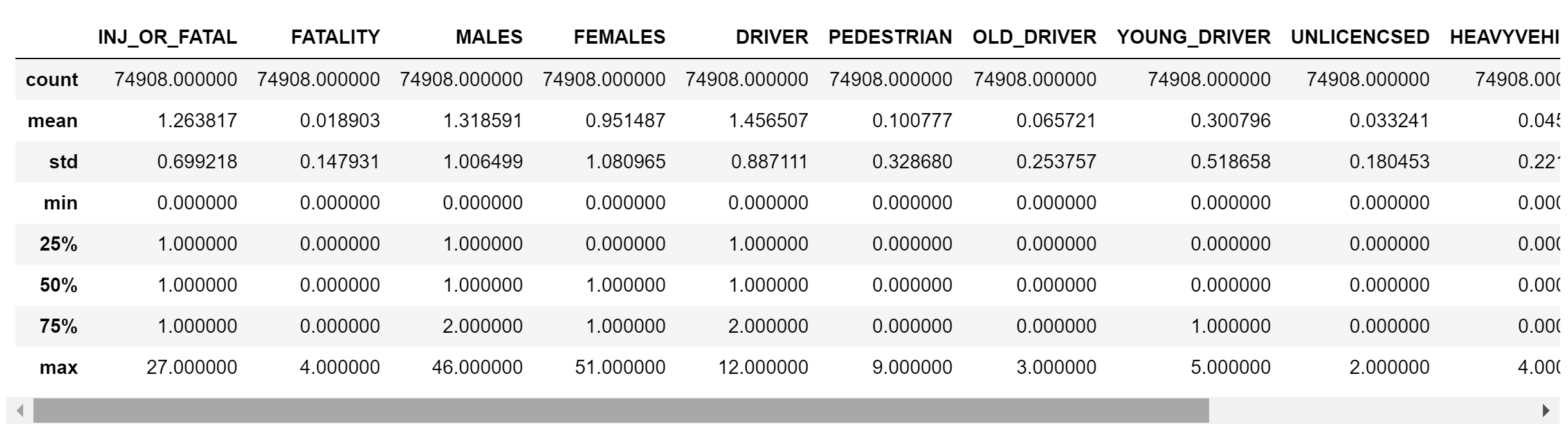
### **2.1 Numeric Features**

The main characteristics of numeric variables are the center, spread, outliers, modality (number of peaks in the probability density function), Cumulative Distribution Function, shape (including the heaviness of the tails).

#### 2.1.1 Center & Spread

The most common & useful measures of central tendency are the statistics of the arithmetic mean, median, and sometimes mode. For any symmetrically shaped distribution, the mean is the point around which the symmetry holds. On the other hand, spread refers to the variability of data. It is an indicator of how far away from the center data values are likely to be found. The spread of a distribution can be described by some known measures including variance and standard deviation. As mentioned earlier statistics like mean, median, std, and percentiles can easily be calculated by pandas as shown below:

numeric\_features.describe()

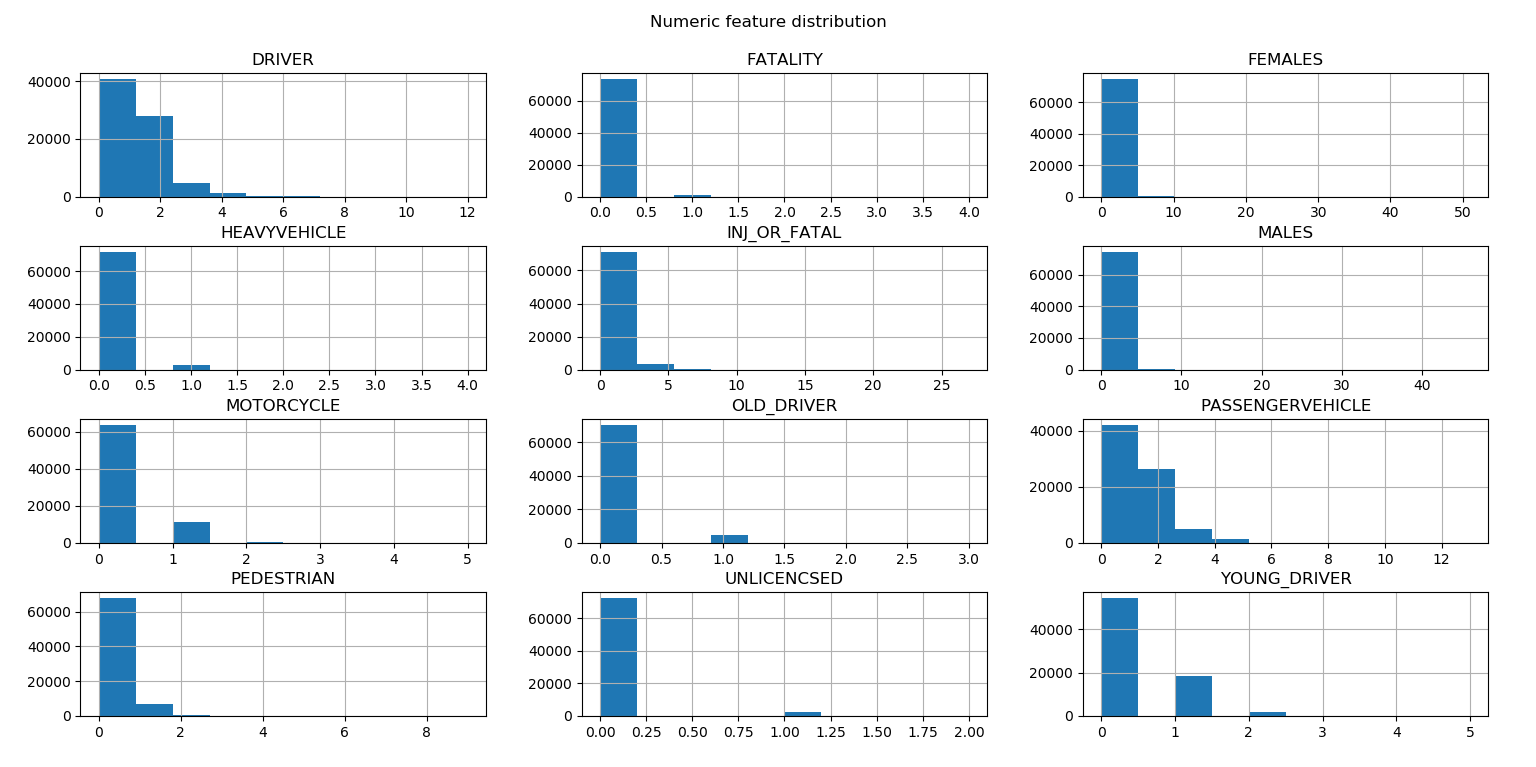
Numeric features data sample

Histograms can still be used to show the distribution of variables of interest. You can even plot the distributions for multiple variables concurrently. This will give you a holistic view of all variables for better understanding. Below, we show all numeric feature distributions using the histogram (barplot). The x-axis represents the given values and the y-axis represents their frequencies.

numeric\_features.hist(figsize=[3,3])

plt.suptitle("Numeric feature distribution")

plt.show()

Numeric features distributions

#### 2.1.2 Outliers

The definition of outliers is tricky, as there is no generally recognized one formal definition for outliers. Roughly, it refers to values that are outside the areas of a distribution that would commonly occur. In addition, another common definition considers any point away from the mean by more than a fixed number of standard deviations be an “outlier”. In other words, we can consider data values corresponding to areas of the population with low density or probability as suspected outliers. The Boxplot is very good at presenting statistical information such as outliers. The plot consists of a rectangular box bounded above and below by “hinges” that represent 75% and 25% quantiles respectively. We can view the “median” as the horizontal line through the box. You can also see the upper and lower “whiskers”. The vertical axis is in the units of the quantitative variable. Data points above the upper whiskers and far away are the suspected “outliers”. The car crash dataset may not be the best to explain this feature. However, we will include just for completeness and coverage. Below is an example of plotting boxplot for three variables of (”INJ\_OR\_FATAL”, “YOUNG\_DRIVER”, “PASSENGERVEHICLE”):

sns.boxplot(x='ALCOHOL\_RELATED',y='INJ\_OR\_FATAL',data=data)

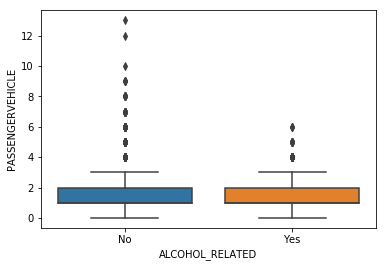
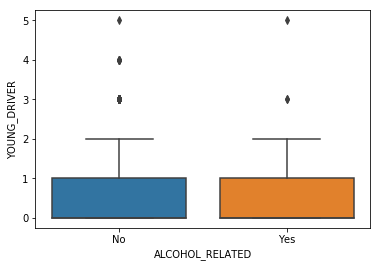
plt.show()

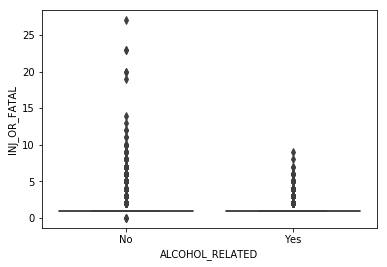
sns.boxplot(x='ALCOHOL\_RELATED',y='YOUNG\_DRIVER',data=data)

plt.show()

sns.boxplot(x='ALCOHOL\_RELATED',y='PASSENGERVEHICLE',data=data)

plt.show()

* Number of passengers BoxPlot
* Number of young drivers BoxPlot

Number of injuries BoxPlot

**Insights:**

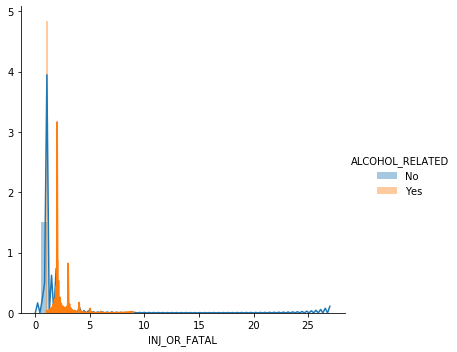
1. The number of injuries over 10 is less likely to be due to alcohol consumption
2. The more the passenger cars involved in crashes don’t really affirm it is due to alcohol consumption
3. Young drivers involved in crashes alcohol-related are between 1 and 2

#### 2.1.3 Probability Density Function (PDF)

We will start by visualizing the PDF for a single feature. As an example, we show the PDF of the number of inquiries in relation to crashes due to alcohol consumption. Below, we plot the PDF graph for the number of injuries. The x-axis represents the value ranges while the y-axis represents the percentage of data points for each target value.

sns.FacetGrid(data,hue="ALCOHOL\_RELATED",height = 5).map(sns.distplot,"INJ\_OR\_FATAL"). add\_legend()

plt.show()

Probability Density Function (PDF) for the number of injuries in relation to alcohol

**Insights:**

1. It is highly likely that crashes due to alcohol consumption results in more casualties
2. Very limited overlapping is observed between the two PDFs, which tells us that the number of casualties are visually different in relation to alcohol consumption

#### 2.1.4 Cumulative Distribution Function (CDF)

CDF is the probability that a corresponding continuous random variable has a value less than or equal to a given value. It gives the area under the probability density function. It is a similar concept to a cumulative frequency table, where the frequency is the amount of times a particular number or item takes place. Below is the code to generate a plot for CDF to the INJ\_OR\_FATAL variable.

counts\_related, bin\_edges\_related = np.histogram(alchohol\_related\_yes['INJ\_OR\_FATAL'], bins=10, density = True)

pdf\_related = counts\_related/(sum(bin\_edges\_related))

cdf\_related = np.cumsum(pdf\_related)

plt.plot(bin\_edges\_related[1:], pdf\_related)

plt.plot(bin\_edges\_related[1:], cdf\_related, label = 'Yes')

plt.xlabel('INJ\_OR\_FATAL')

counts\_not\_related, bin\_edges\_not\_related = np.histogram(alchohol\_related\_no['INJ\_OR\_FATAL'], bins=10, density = True)

pdf\_not\_related = counts\_not\_related/(sum(counts\_not\_related))

cdf\_not\_related = np.cumsum(pdf\_not\_related)

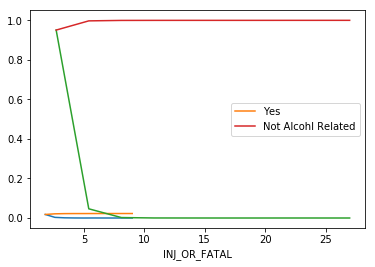
plt.plot(bin\_edges\_not\_related[1:], pdf\_not\_related)

plt.plot(bin\_edges\_not\_related[1:], cdf\_not\_related, label = 'Not Alcohl Related')

plt.xlabel('INJ\_OR\_FATAL')

plt.legend()

plt.show()

CDF of the number of injuries in relation to alcohol-related crashes

**Insights:**

1. Almost 95% of crashes not related to alcohol consumption result in less than 5 casualties

### **2.2 Categorical Features**

The characteristics of interest for a categorical variable are simply the range of values and the frequency of occurrence of each value. One form of useful such univariate analysis is the tabulation of the frequencies. Usually accompanying the frequences the calculation of the fraction (or %) of data that fall in each category. We should expect that the proportions add up to 1.00 (or 100%). This can be very helpful for finding mistakes and missing data. Below are the proportions of the unique variables for the three parameters of ACCIDENT\_TYPE and LIGHT\_CONDITION.

counts = categorical\_features["ACCIDENT\_TYPE"].value\_counts()

percent100 = categorical\_features["ACCIDENT\_TYPE"].value\_counts(normalize=True).mul(100).round(1).astype(str) + '%'

light\_conditions=pd.DataFrame({'counts': counts, 'Percent': percent100})

print(light\_conditions)

counts = categorical\_features["LIGHT\_CONDITION"].value\_counts()

percent100 = categorical\_features["LIGHT\_CONDITION"].value\_counts(normalize=True).mul(100).round(1).astype(str) + '%'

light\_conditions=pd.DataFrame({'counts': counts, 'Percent': percent100})

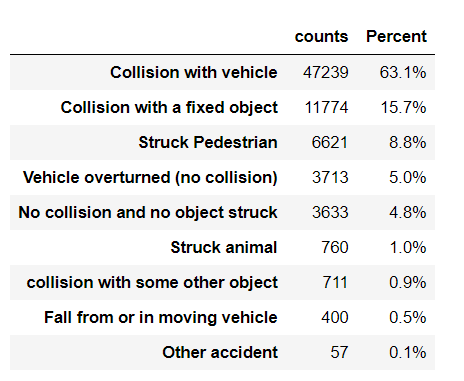
print(light\_conditions)

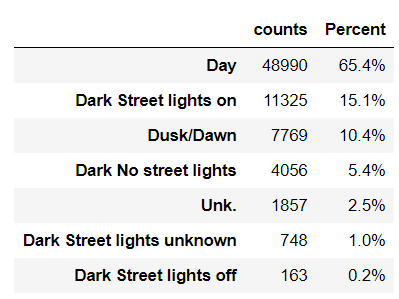
counts = categorical\_features["DAY\_OF\_WEEK"].value\_counts()

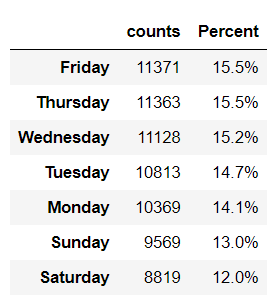
percent100 = categorical\_features["DAY\_OF\_WEEK"].value\_counts(normalize=True).mul(100).round(1).astype(str) + '%'

acc\_day=pd.DataFrame({'counts': counts, 'Percent': percent100})

print(acc\_day)

Crashes broken by types

Crashes broken by lighting conditions

Crashes broken by weekday

**Insights:**

1. The majority of crashes 63% occur due to collision with other moving vehicles while only 8% striking pedestrians.
2. 65% of accidents take place during the day time and only 5% take place in streets with no light.
3. Crashes on the weekend are lower than on weekdays while percentages get higher closer to the end of the week.

## **3. Bi-Variate analysis**

**Bivariate analysis** is another step in our EDA process, where the analysis takes place between two variables (features). The main purpose of such analysis is to explore the concept of the relationship between two variables. This covers the association, strength and whether there are differences and their significance.

### **3.1 Numerical Features**

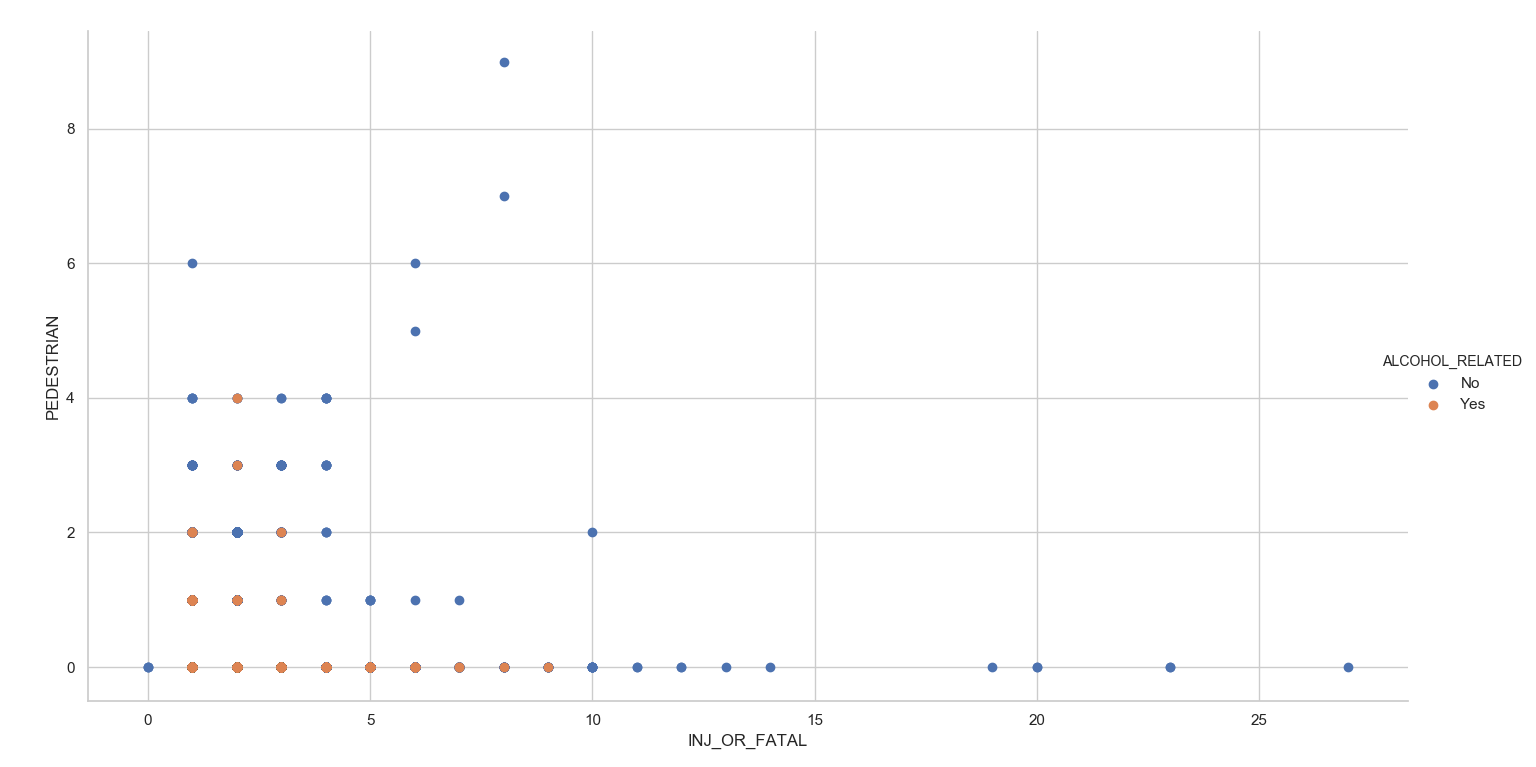
#### 3.1.1 Scatter Plot

A scatter plot can be a very useful representation to visualize the relationship between two numerical variables. In fact, it is most beneficial to plot it before fitting a regression model to inspect the potential linear correlation. The resulting pattern indicates the type (linear or non-linear) and strength of the relationship between two variables. We can add more information to the 2d scatter plot. For example, we may label points in relation to crashes due to alcohol-related or not. Below is the scatter plot of the number of injuries versus the number of pedestrians.

sns.set\_style("whitegrid")

sns.FacetGrid(data, hue = "ALCOHOL\_RELATED" , height = 6).map(plt.scatter,"INJ\_OR\_FATAL","PEDESTRIAN").add\_legend()

plt.show()

Number of injuries versus those towards pedestrians in relation to alcohol consumption

**Insights:**

1. The higher number of pedestrians being injured is not due to crashes involving alcohol consumption

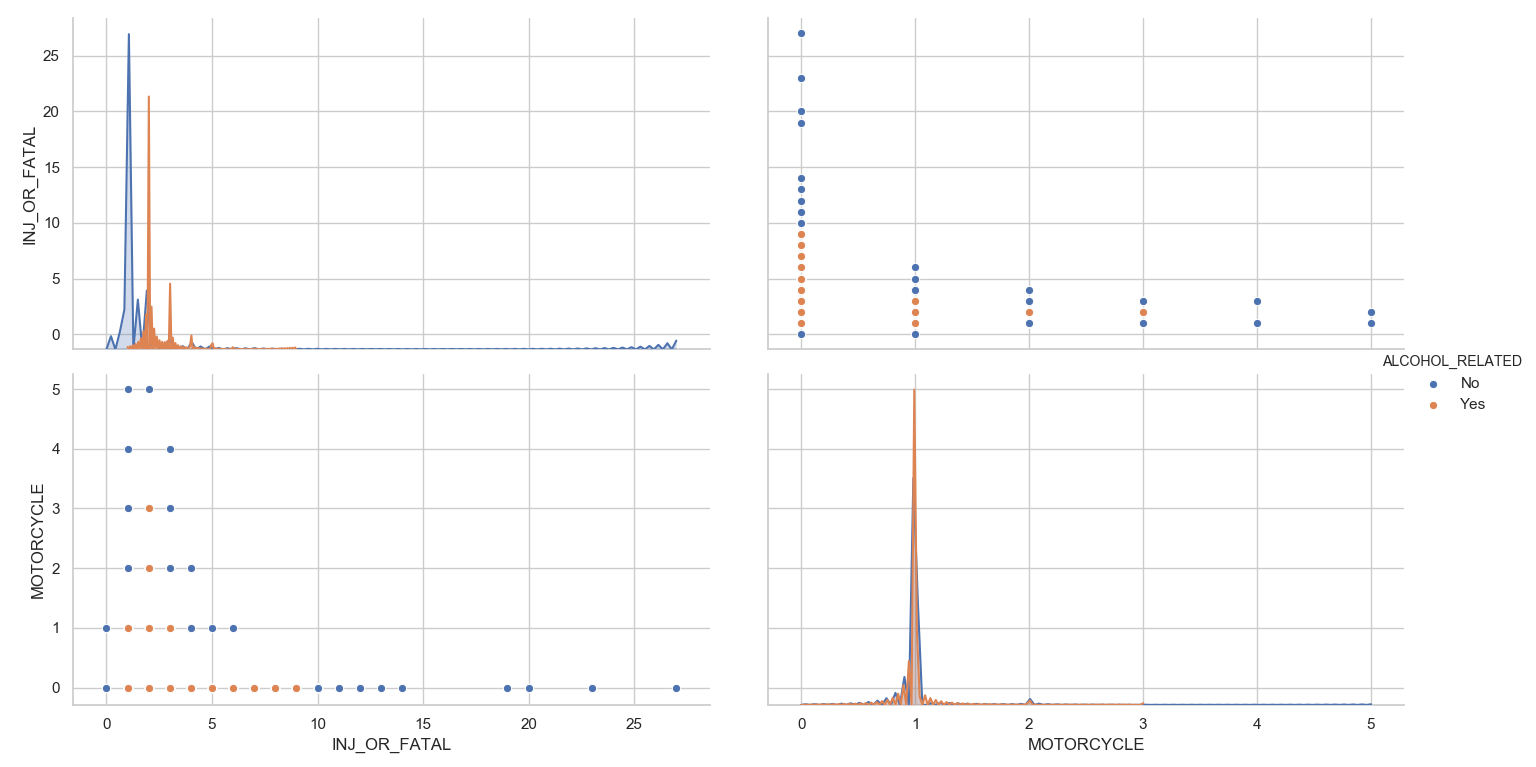
Moreover, we can plot pairs of variables for a better understanding of variables associations. Below, we plot the pair association for the number of injuries against the number of motorcyclists involved in the crash.

selected\_numeric\_features = data[["INJ\_OR\_FATAL","MOTORCYCLE","ALCOHOL\_RELATED"]]

sns.set\_style("whitegrid")

sns.pairplot(selected\_numeric\_features, hue = "ALCOHOL\_RELATED", height = 5)

plt.show()

Number of injuries vs number motorcyclists involved in crashes in relation to alcohol consumption

**Insights:**

1. Very low number of injuries related to motorcyclists being involved in crashes involving alcohol consumption

#### 3.1.2 Correlation matrix

A correlation matrix is “square matrix” with the same variables shown in the rows and columns. The level of correlation between the variables is highlighted with different colour intensities. The numeric values for the correlation range from- 1 “not correlated or negatively correlated ” to 1 “highly correlated”.  Among the use cases of a correlation matrix is to summarize data to a more advanced analysis.  As a result, some key decisions can be made when creating a correlation matrix. One is the choice of relevent correlation statistics and the coding of the variables. Another is the treatment of the missing data and dropping highly correlated variables from feature sets is another use case.  Below, we plot the correlation heatmap for all pairs of numeric features.

corr = numeric\_features.corr()

# Generate a mask for the upper triangle

mask = np.zeros\_like(corr, dtype=np.bool)

mask[np.triu\_indices\_from(mask)] = True

# Set up the matplotlib figure

f, ax = plt.subplots(figsize=(11, 9))

# Generate a custom diverging colormap

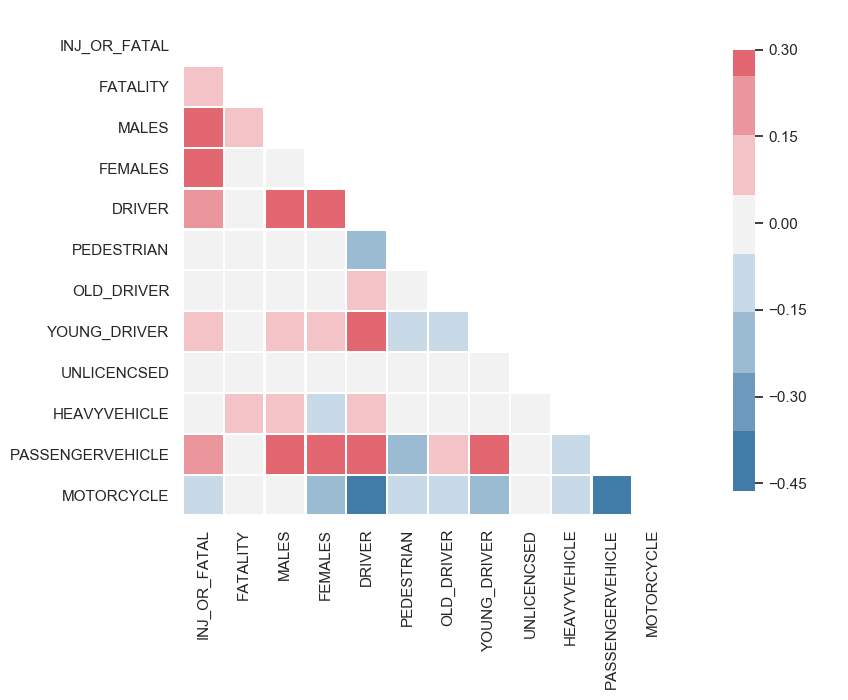
cmap = sns.diverging\_palette(240, 10, n=9)

# Draw the heatmap with the mask and correct aspect ratio

sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,

square=True, linewidths=.5, cbar\_kws={"shrink": .8})

plt.show()

Numeric features correlation

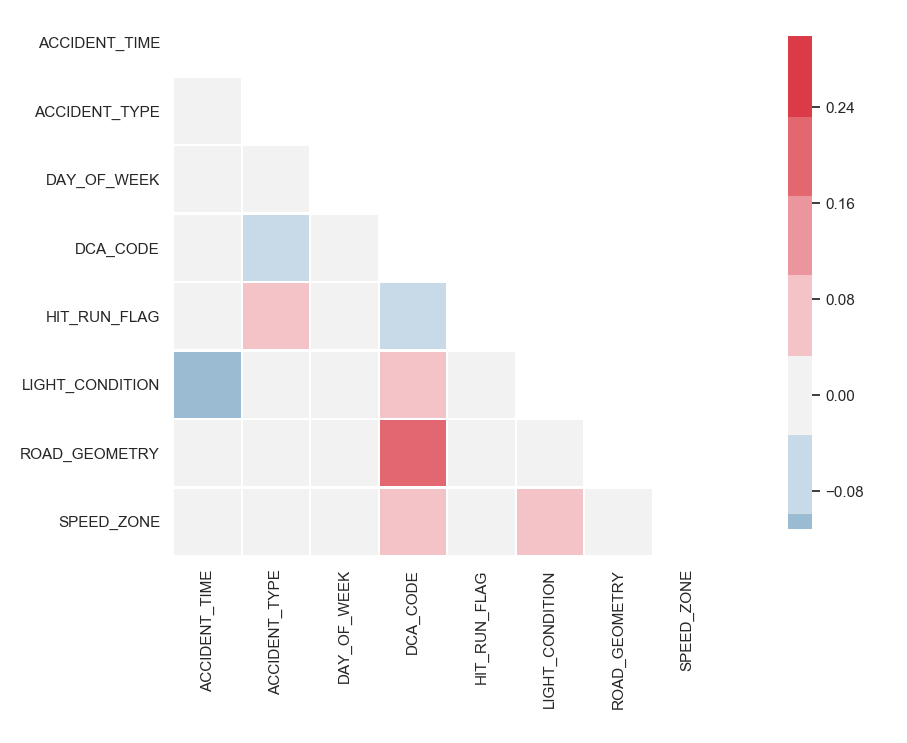
**Insights:**

1. High correlation exists between number of males involved in the crash and the number of injuries
2. Higher correlation is witnessed between number of young drivers involved in the crash and number of drivers.
3. Weak correlation exists between number of motorcycles and number of passenger vehicles

### **3.2 Categorical Features**

#### 3.2.1 Correlation matrix

In order to calculate the correlation matrix for pairs of categorical features, we need first to encode the textual values into numeric ones to be able to plot it. We can do so using SciKit’s “LabelEncoder” for simplicity.

                                            Categorial Correlations

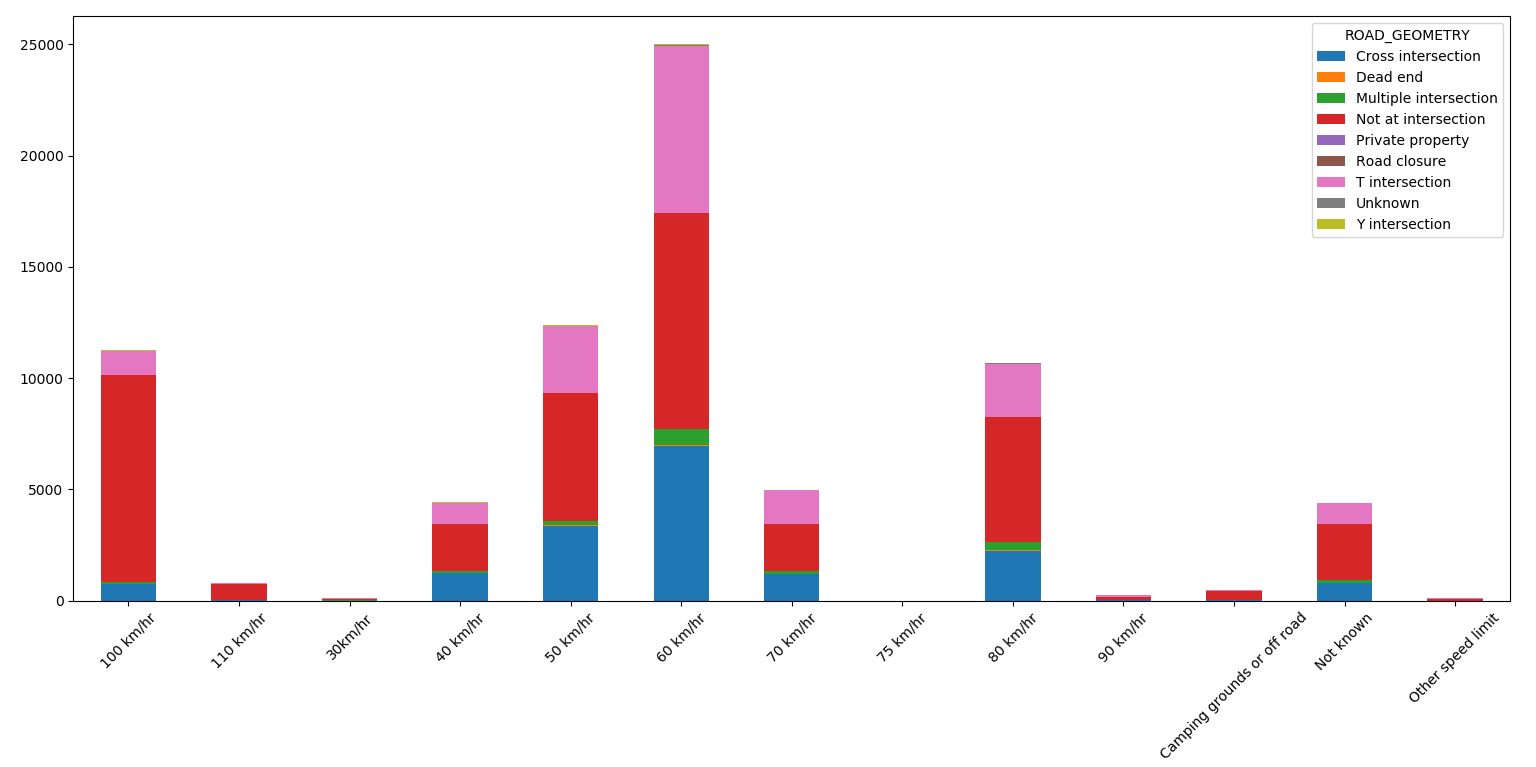
**Insights:**

1. High correlation is noticed between DCA codes (e.g. REAR END, VEHICLE OFF FOOTPATH STRIKES VEH ON CARRIAGEWAY) and Road Geometry  which totally make sense as road geometry has a say in the crash type.
2. Weak correlation is seen between accident time and lighting condition

#### 3.2.2 Stacked Column Chart

A stacked column plot can be useful in visualizing the relationship between two categorical variables. We can compare the percentages that each category from one variable contributes to a total across categories of the second variable.  Below is a stacked plot for comparing crashes taking place on different road geometries versus speed zone limits.

day\_vs\_acc\_type = pd.crosstab(index=data["SPEED\_ZONE"], columns=data["ROAD\_GEOMETRY"]) day\_vs\_acc\_type.plot(kind="bar", figsize=(8,8),stacked=True)



Crashes on different road geometry types with different speed limits

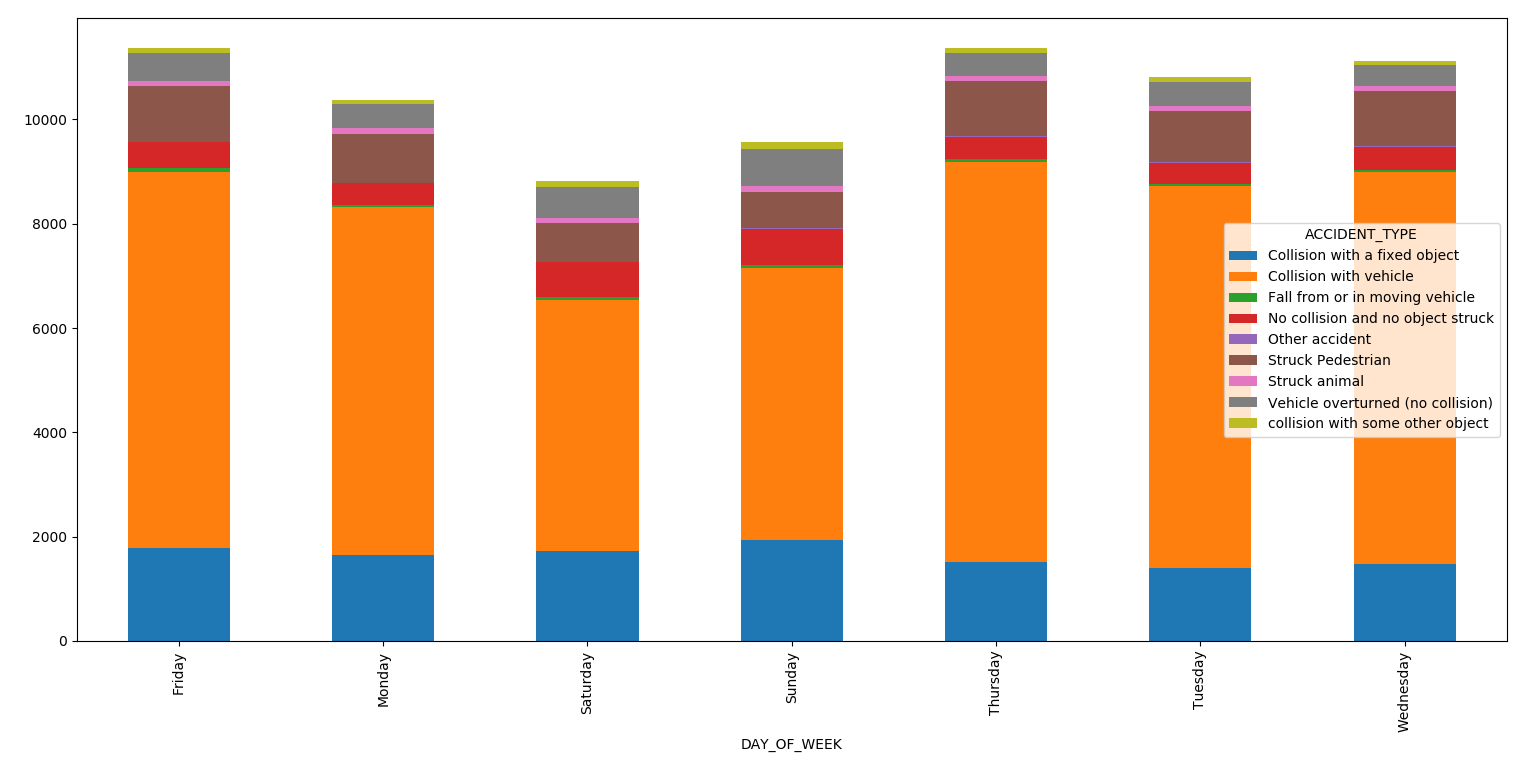
**Insights:**

1. More crashes at T-intersections on 60 km speed zones
2. Crashes at higher speed zones of 100 km usually occur at roads with no intersection

Another example for analyzing pair of variables is for looking at the accident types and the day of week. Below,is the stacked bar chart for such analysis:

day\_vs\_acc\_type = pd.crosstab(index=data["DAY\_OF\_WEEK"], columns=data["ACCIDENT\_TYPE"])

day\_vs\_acc\_type.plot(kind="bar", figsize=(8,8),stacked=True)



Number of crashes for each day of week broken by accident type

**Insights:**

1. Two car collisions is the dominant accident type at all times
2. Collision with fixed object is highest on the weekend (especially Saturday)
3. Less pedestrians are struck on weekends

## **4. Multivariate Analysis**

In the process of EDA, sometimes, the inspection of single or pairs of variables won’t suffice to rule out certain hypothesis (or outliers & anomalous cases) from your dataset. That’s why multivariate analysis come in play. This type of analysis generally shows the relationship between two or more variables using some statistical techniques.  There comes the need for taking in consideration more variables at a time during analysis to reveal more insights in your data.

### **4.1 Contour Plot**

A contour line or isoline of a function of two variables is a curve along which the function has a constant value. It is a cross-section of the three-dimensional graph of the function f(x, y) parallel to the x, y plane. Lets plot this graph between the number of injuries and the time of day crashes took place. The features in this dataset may not be the best to show the best visualization for such plot, but for completeness we will include it.

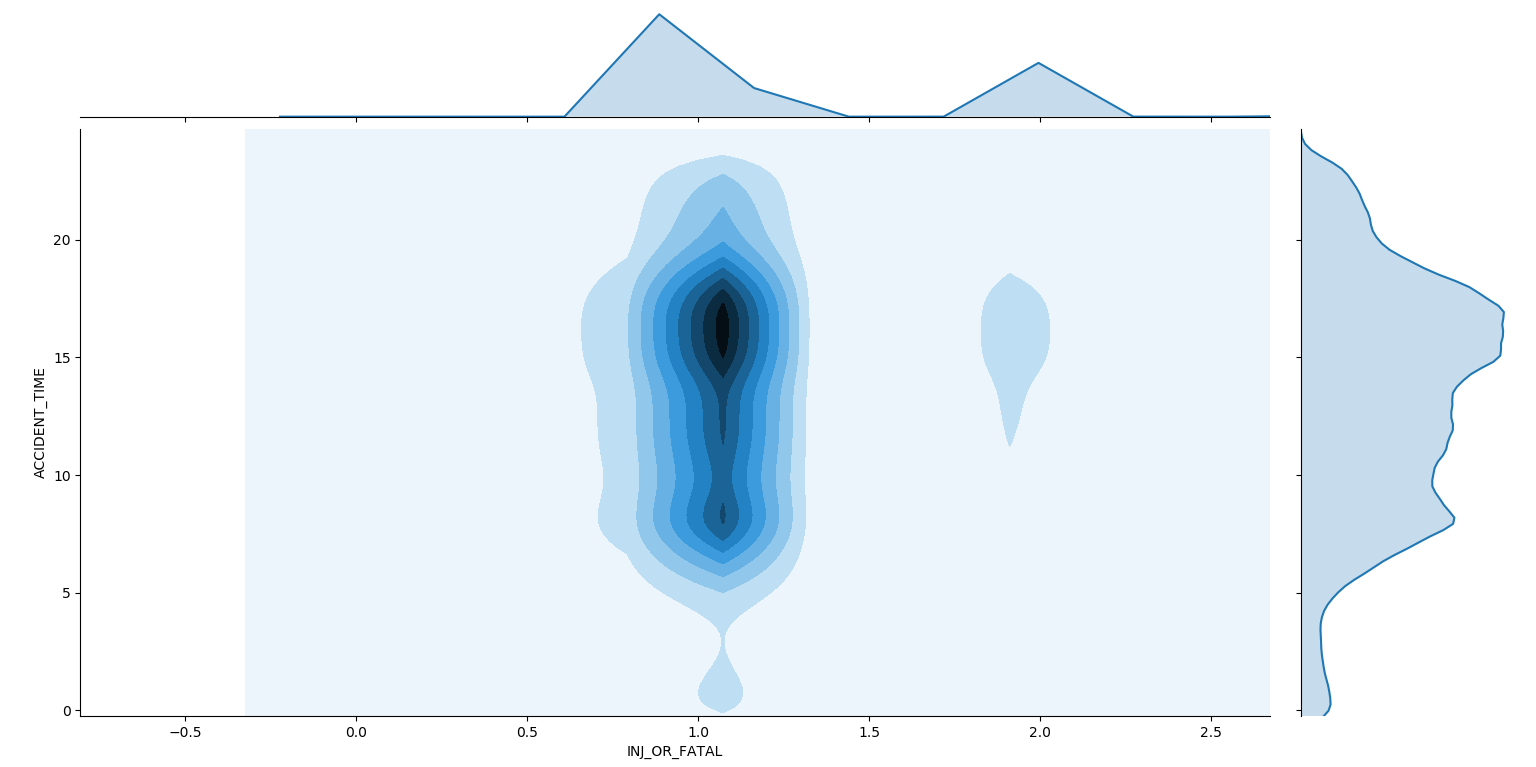
data['ACCIDENT\_TIME'] = data['ACCIDENT\_TIME'].str[:2]#just the hour ofday

print(data['ACCIDENT\_TIME'].values)

data.ACCIDENT\_TIME = data.ACCIDENT\_TIME.astype(float).fillna(0.0)

sns.jointplot(x = "INJ\_OR\_FATAL", y = "ACCIDENT\_TIME", data = data, kind = "kde")

plt.show()



Contour plot for accident time versus number of injuries

**Insights:**

1. The darkest area or the highest in density in the context of this plot means the highest number of injuries take place in the afternoon around the times from 3 pm to to 8 pm.

This plot can also be visualized in 3D where it will show hill like structure where hill top has maximum density of point and density decreases as hill slope getting decreases.

### **4.2 Principal Component Analysis (PCA)**

PCA is a statistical data transformation procedure. It employs an orthogonal transformation to convert a set of observations of correlated variables into a set of values of linearly uncorrelated variables called principal components. It is a very common way of speeding up a machine learning algorithm is which reduces the dimensionality of input features which more often is a reasonable choice. In fact, it can completely restructure the data, removing redundancies and ordering newly obtained components according to the amount of the original variance that they express. Lets reduce the 12 numerical features to 5 components to investigate whether we can capture most of the variation between samples using a smaller number of new variables. First, we need to standardize the variables under study using the scale() function which is necessary if the input variables have very different variances. Then, we perform the principal component analysis. In order to decide how many principal components should be retained, it is common to summarize the results by making by plotting the components against the variance as shown below.

def screeplot(pca, standardised\_values):

y = np.std(pca.transform(standardised\_values), axis=0)\*\*2

x = np.arange(len(y)) + 1

plt.plot(x, y, "o-")

plt.xticks(x, ["Comp."+str(i) for i in x], rotation=60)

plt.ylabel("Variance")

plt.show()

X = numeric\_features

standardisedX = scale(X)

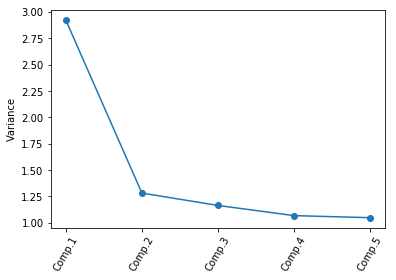
standardisedX = pd.DataFrame(standardisedX, index=X.index, columns=X.columns)

standardisedX.apply(np.std)

dim\_reduction = PCA(n\_components=5)

Xc = dim\_reduction.fit\_transform(scale(X))

screeplot(dim\_reduction, standardisedX)



The variance of different PCA components

**Insights:**

1. The most obvious change in slope in the scree plot occurs at component 2, which is the “elbow” of the scree plot which suggests that the first two components should be enough to retain.

Moreover, PCA can give great insights about how the set of features collectively collaborate to describe the analysis outcome (target). Lets convert the 12 numerical features into 2 PCA components.

dim\_reduction = PCA(n\_components=2)

Xc = dim\_reduction.fit\_transform(scale(numeric\_features))

colors = ['navy', 'darkorange']

target\_names = ['non-Alcohol Related','Alcohol Related']

lw = 2

y = data["ALCOHOL\_RELATED"].values

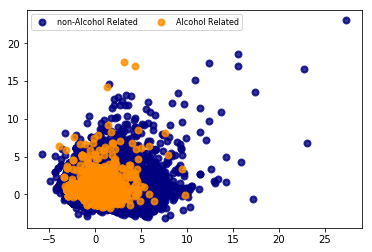
y = pd.Series(np.where(data.ALCOHOL\_RELATED.values == 'Yes', 1, 0),data.index)

print(y.shape)

for color, i, target\_name in zip(colors, [0, 1], target\_names):

plt.scatter(Xc[y == i, 0], Xc[y == i, 1], color=color, alpha=.8, lw=lw,label=target\_name)

plt.show()



2 PCA components to represent 12 numeric features in relation to alcohol consumption

**Insights:**

1. The predictive power of the numeric features can not easily define a classifying pattern of whether a crash is due to alcohol consumption or not.
2. This suggests the need for other supporting features (e.g. categorical features)or
   1. Add more features that were previously removed and perform some more feature engineering on them
   2. Utilize external data sources.

## **Conclusion**

We could see that it is difficult to actually understand the dataset and make conclusions without looking through the entire data set. In fact, we have shown that spending more time exploring the dataset is well invested time. It may be a tedious preliminary step to processing data, but it is a necessary evil. However, when you begin to actually see interesting insights you will appreciate every single minute invested in such a process. EDA enables Data scientists to immediately understand key issues in the data and be able to guide deeper analysis in the right directions. Successfully exploring the data ensures to stakeholders that they won’t be missing out on opportunities to leverage their data. They can easily pinpoint risks including poor data quality, unreliable, poor feature engineering, and other uncertainties. In this article, we covered a range of useful introductory data exploratory/data analysis methods and visualization techniques. The shown EDA guidelines should allow Data scientists to have insightful and deeper understanding of the problem at hand and decide on the next move confidently.