CHICAGO TRAFFIC CRASHES

Bharati Kandakumar | Divya Shankar | Lasya Pathuri | Snigdha Jain

Dataset: Chicago Traffic Crashes

Traffic crashes on city streets within City of Chicago city limits under jurisdiction of Chicago Police Department

Data shown is from Electronic reporting system at CPD

Number of datapoints in the Dataset

```
In [5]:  1  df.size
Out[5]: 14091023
```

Size of Dataset (Rows x Columns)

```
In [6]:  1  df.shape
Out[6]: (299809, 47)
```

Dataset: Chicago Traffic Crashes

```
In [4]:  1 df.head(5)
```

Out[4]:

RD_NO	CRASH_DATE_EST_I	CRASH_DATE	POSTED_SPEED_LIMIT	TRAFFIC_CONTROL_DEVICE	DEVICE_CONDITION	WEATHER_CONDITION	LIGHTING_CONI
JC273063	NaN	05/22/2019 12:20:00 AM	25	STOP SIGN/FLASHER	FUNCTIONING PROPERLY	OTHER	DARKNESS, LI
JC273065	NaN	05/22/2019 12:00:00 AM	30	TRAFFIC SIGNAL	FUNCTIONING PROPERLY	RAIN	DARKNESS, LI
JC273046	NaN	05/21/2019 11:40:00 PM	30	TRAFFIC SIGNAL	FUNCTIONING PROPERLY	RAIN	DAR
JC273001	NaN	05/21/2019 09:30:00 PM	15	NO CONTROLS	NO CONTROLS	CLEAR	DARKNESS, LI
JC272944	NaN	05/21/2019 09:24:00 PM	30	TRAFFIC SIGNAL	FUNCTIONING PROPERLY	CLEAR	DAR

5 rows × 47 columns

Data Cleaning

Data Cleaning

Drop Unnecessary columns

Dropping Unnecessary Columns

```
df.drop(['CRASH_DATE_EST_I', 'PHOTOS_TAKEN_I' , 'DOORING_I', 'WORK_ZONE_I','WORK_ZONE_TYPE','WORKERS_PRESENT_I']\
        , axis = 1, inplace = True)

```

Dropping some rows which have NaN values

Dropping Rows that have NUM_UNITS as Nan

```
df = df[np.isfinite(df['NUM_UNITS'])]
```

Data Cleaning

Change Data type of few columns to make suitable for analysis

Changing Hit and Run value to Boolean and dropping the old column

```
1 df['HIT_AND_RUN'] = (df.HIT_AND_RUN_I == 'Y')
2 del df['HIT_AND_RUN_I']
```

Converting to datetime format

```
1 df['date'] = pd.to_datetime(df['CRASH_DATE'])
```

Adding a column ‘Year’

Creating a column Year

```
1 df['year'] = df['date'].dt.year
```

Data Cleaning

Replacing NaN with zero for some columns

Updated dataset

Replacing Nan with 0 for some columns

```
df['MOST_SEVERE_INJURY'].fillna(0 , inplace = True)
df['INJURIES_TOTAL'].fillna(0 , inplace = True)
df['INJURIES_FATAL'].fillna(0 , inplace = True)
df['INJURIES_INCAPACITATING'].fillna(0 , inplace = True)
df['INJURIES_NON_INCAPACITATING'].fillna(0 , inplace = True)
df['INJURIES_REPORTED_NOT_EVIDENT'].fillna(0 , inplace = True)
df['INJURIES_NO_INDICATION'].fillna(0 , inplace = True)
df['INJURIES_UNKNOWN'].fillna(0 , inplace = True)
```

Size of dataset after cleaning

```
df.shape
```

```
(299095, 41)
```

Updated Columns

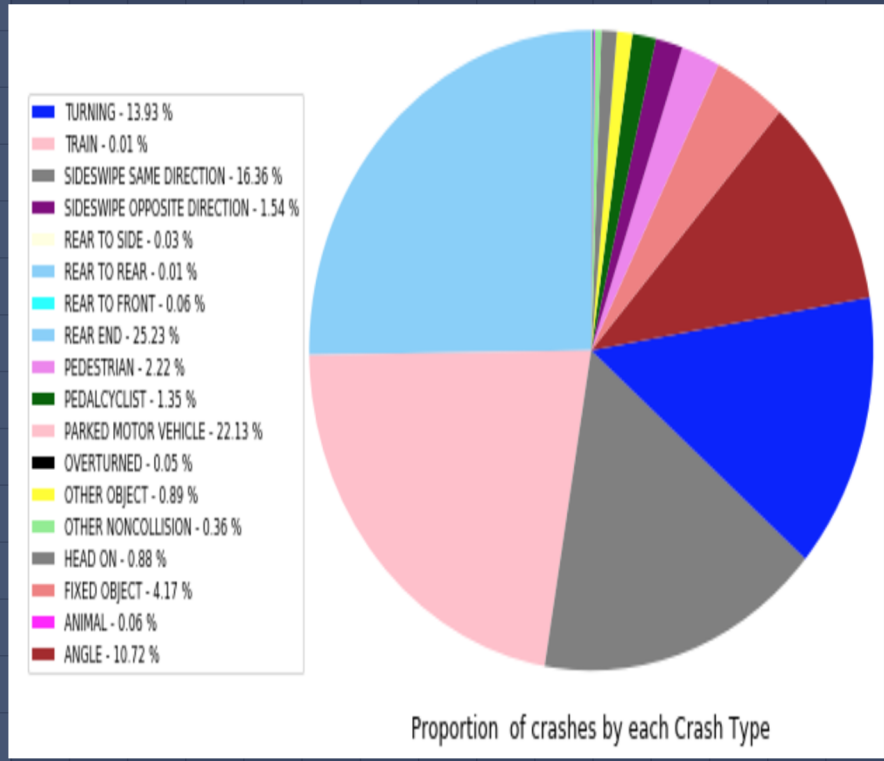
```
df.columns
```

```
Index(['CRASH_DATE', 'POSTED_SPEED_LIMIT', 'TRAFFIC_CONTROL_DEVICE',
       'DEVICE_CONDITION', 'WEATHER_CONDITION', 'LIGHTING_CONDITION',
       'FIRST_CRASH_TYPE', 'TRAFFICWAY_TYPE', 'LANE_CNT', 'ALIGNMENT',
       'ROADWAY_SURFACE_COND', 'ROAD_DEFECT', 'REPORT_TYPE', 'CRASH_TYPE',
       'INTERSECTION_RELATED_I', 'NOT_RIGHT_OF_WAY_I', 'HIT_AND_RUN_I',
       'DAMAGE', 'DATE_POLICE_NOTIFIED', 'PRIM_CONTRIBUTORY_CAUSE',
       'SEC_CONTRIBUTORY_CAUSE', 'STREET_NO', 'STREET_DIRECTION',
       'STREET_NAME', 'BEAT_OF_OCCURRENCE', 'STATEMENTS_TAKEN_I', 'NUM_UNITS',
       'MOST_SEVERE_INJURY', 'INJURIES_TOTAL', 'INJURIES_FATAL',
       'INJURIES_INCAPACITATING', 'INJURIES_NON_INCAPACITATING',
       'INJURIES_REPORTED_NOT_EVIDENT', 'INJURIES_NO_INDICATION',
       'INJURIES_UNKNOWN', 'CRASH_HOUR', 'CRASH_DAY_OF_WEEK', 'CRASH_MONTH',
       'LATITUDE', 'LONGITUDE', 'LOCATION'],
      dtype='object')
```

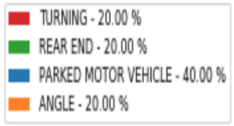
Data Analysis

Insight 1:

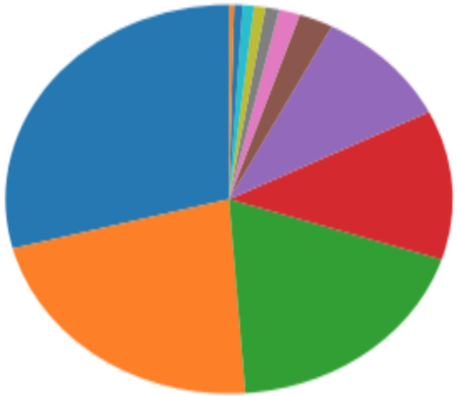
Finding proportion of each crash type with reference to total number of crashes and proportion of each type of crash for each year



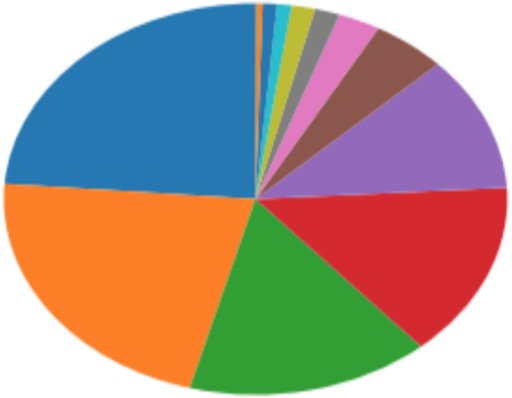
Year 2014



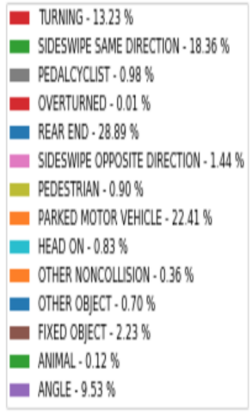
Year 2016

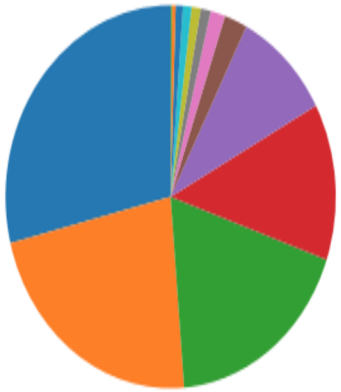


Year 2018

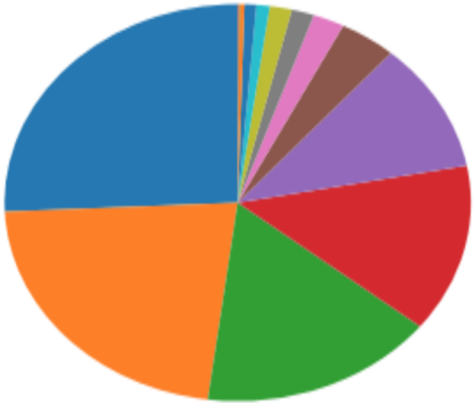


Year 2015

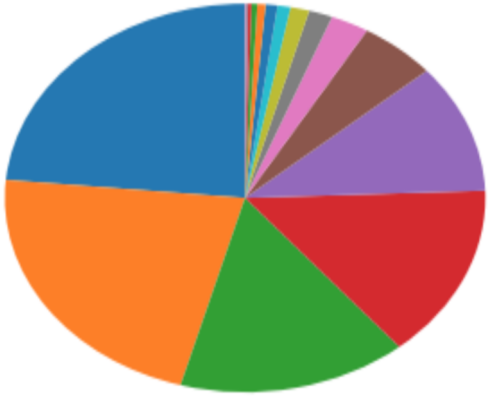




Year 2017



Year 2019



Counting total number of Crashes by each crash type

In [15]:
```
data_plot  = df.FIRST_CRASH_TYPE.value_counts()
```

Calculating the percentage of occurence of each crash type

In [16]:
```
den = data_plot.sum()
percent = data_plot*100/den
percent
```

Out[16]:
```
REAR END                       25.231448
PARKED MOTOR VEHICLE           22.127752
SIDESWIPE SAME DIRECTION       16.358013
TURNING                        13.934034
ANGLE                          10.724352
FIXED OBJECT                    4.172253
PEDESTRIAN                      2.223040
SIDESWIPE OPPOSITE DIRECTION    1.540313
PEDALCYCLIST                    1.349738
OTHER OBJECT                    0.885337
HEAD ON                         0.878651
OTHER NONCOLLISION              0.360421
ANIMAL                          0.063191
REAR TO FRONT                   0.056838
OVERTURNED                      0.047811
REAR TO SIDE                    0.030759
REAR TO REAR                    0.011033
TRAIN                           0.005015
Name: FIRST_CRASH_TYPE, dtype: float64
```

Plotting and Saving the Crash Type Percentages as a Pie Chart

In [17]:
```
colors = ['lightskyblue','pink','grey','blue','brown','lightcoral','violet','purple', 'darkgreen','yellow','grey','ligh
explode = (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
patches, texts = plt.pie(data_plot, startangle=90,explode = explode,colors=colors, radius=1.5)
labels = ['{0} - {1:1.2f} %'.format(i,j) for i,j in zip(data_plot.index,percent)]
sort_legend = True
if sort_legend:
    patches, labels, dummy =  zip(*sorted(zip(patches, labels, data_plot.index),
                                          key=lambda data_plot: data_plot[2],
                                          reverse=True));

plt.legend(patches, labels, loc='upper right', bbox_to_anchor=(-0.1, 1.),
           fontsize=8);
plt.title('Proportion  of crashes by each Crash Type', y = -0.25  );
plt.savefig('piechart.png', bbox_inches='tight');
```

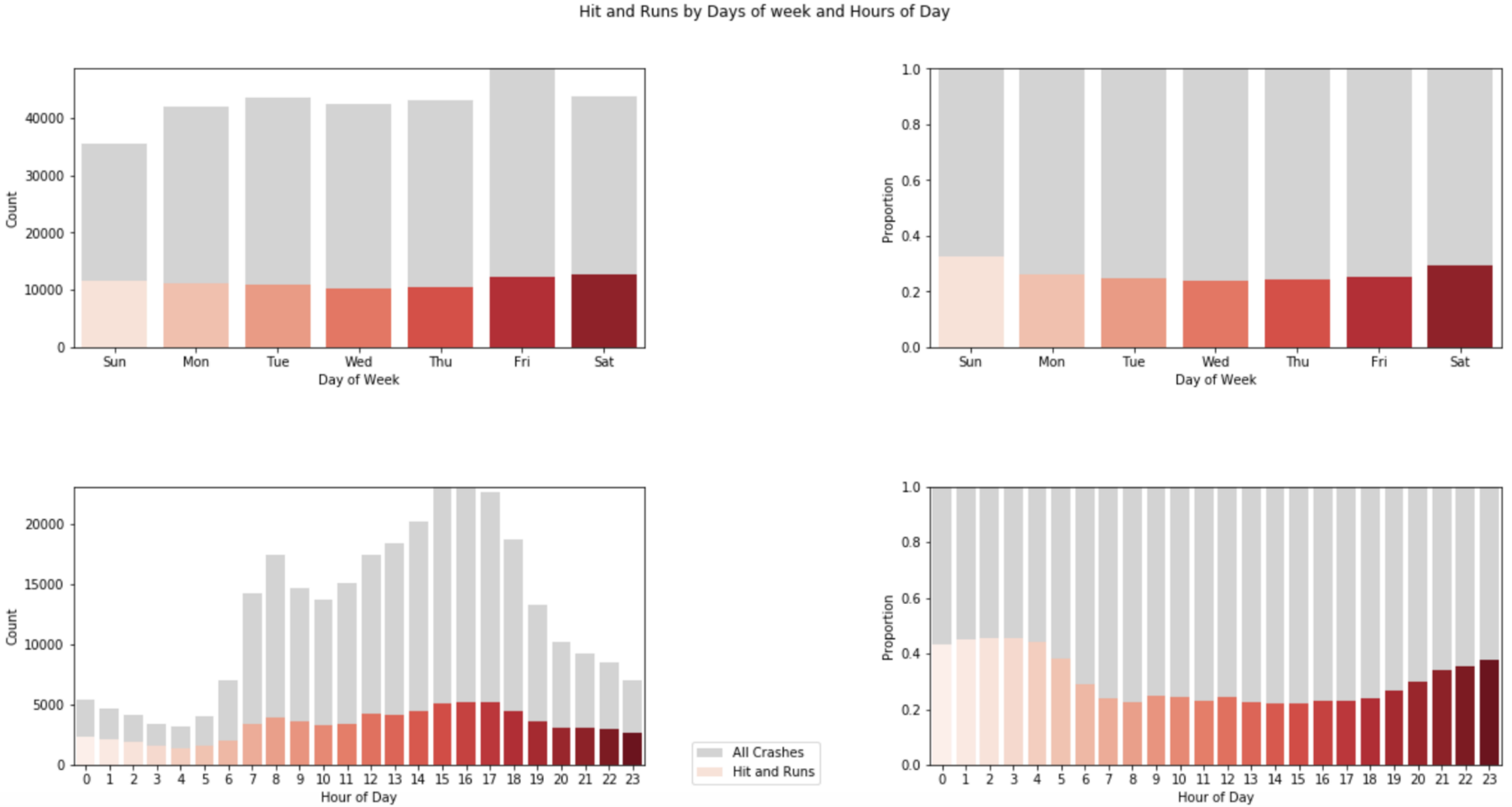
Inference 1:

- Maximum number of crashes occur due to "rear end" ~25.23% of total number of crashes, followed by "parked motor vehicle" and "sideswipe same direction"
- Crashes due to "parked motor vehicles" is the highest in 2014. May be due to:
  - Limited availability of data for beginning years
  - Later years show that number of crashes increases, and crashes due to "rear end" take a lead, fluctuating between 23%-29% over next couple of years
- **Suggestion**: Chicago Police Department, could conduct awareness campaigns on safe driving distance and distracted driving

Insight 2:

Analyzing Hit and Run Cases in the following ways:

- Identifying proportion of hit and run cases to number of crashes that occur by each day of week
- Identifying proportion of hit and run cases to number of crashes that occur by hour of day


Hit and Runs by Days of week and Hours of Day
Count
Day of Week
Sun
Mon
Tue
Wed
Thu
Fri
Sat
Proportion
Hour of Day
All Crashes
Hit and Runs

Label for X- axis of chart

In [27]:
```
weekday_order = ["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"]
```

Plotting Hit and Run Cases by each day of the week

In [28]:
```
# Subplots to show two plots side by side
fig, ax = plt.subplots(2, 2);

# Find sum of hit and run for each  day of the week
group_dow_sum = df.groupby('CRASH_DAY_OF_WEEK')['HIT_AND_RUN'].sum();

#Plot total crashes each day
plot_1 = sns.barplot(
    x=group_dow_sum.index, y=df.groupby("CRASH_DAY_OF_WEEK")["CRASH_DATE"].count().values,
    color="lightgray", ax=ax[0][0], label="All Crashes"
);
#Plot hit and run crashes each day
plot_1_Overlap = sns.barplot(
    x=group_dow_sum.index, y=group_dow_sum.values,
    palette="Reds", ax=ax[0][0], label="Hit and Runs"
);

# Set x and y  axis for first chart
ax[0][0].set_xticklabels(weekday_order);
ax[0][0].set_xlabel("Day of Week");
ax[0][0].set_ylabel("Count");
ax[0][0].set_ylim((0, df.groupby("CRASH_DAY_OF_WEEK")["CRASH_DATE"].count().max()));

# Set chart properties

fig.suptitle("Hit and Runs by Days of week and Hours of Day", y=0.95);
fig.subplots_adjust(wspace=0.5 , hspace = 0.5);
fig.set_size_inches((20, 10));
fig.legend(bbox_to_anchor=(0.45, 0.11));

# Find mean of hit and run for each  day of the week
group_dow_mean = df.groupby("CRASH_DAY_OF_WEEK")["HIT_AND_RUN"].mean();

#Plot total crashes each day
plot_2 = sns.barplot(
    x=group_dow_mean.index, y=np.ones(len(group_dow_mean.index)),
    color="lightgray", ax=ax[0][1]
);
#Plot proportion of hit and run each day
plot_2_Overlap = sns.barplot(
    x=group_dow_mean.index, y=group_dow_mean.values,
    palette="Reds", ax=ax[0][1]
);

# Set X and Y axis for second chart
ax[0][1].set_xticklabels(weekday_order);
ax[0][1].set_xlabel("Day of Week");
ax[0][1].set_ylabel("Proportion");
ax[0][1].set_ylim((0, 1));
```

```
group_hour_sum = df.groupby('CRASH_HOUR')['HIT_AND_RUN'].sum();
sns.barplot(
    x=group_hour_sum.index, y=df.groupby('CRASH_HOUR')['CRASH_DATE'].count().values,
    color="lightgray", ax=ax[1][0], label="All Crashes"
);
sns.barplot(
    x=group_hour_sum.index, y=group_hour_sum.values,
    palette="Reds", ax=ax[1][0], label="Hit and Runs"
);
ax[1][0].set_xlabel("Hour of Day");
ax[1][0].set_ylabel("Count");
ax[1][0].set_ylim((0, df.groupby('CRASH_HOUR')['CRASH_DATE'].count().max()));
group_hour_mean = df.groupby('CRASH_HOUR')['HIT_AND_RUN'].mean();
sns.barplot(
    x=group_hour_mean.index, y=np.ones(len(group_hour_mean.index)),
    color="lightgray", ax=ax[1][1]
);
sns.barplot(
    x=group_hour_mean.index, y=group_hour_mean.values,
    palette="Reds", ax=ax[1][1]
);
ax[1][1].set_xlabel("Hour of Day");
ax[1][1].set_ylabel("Proportion");
ax[1][1].set_ylim((0, 1));

plt.show();
```

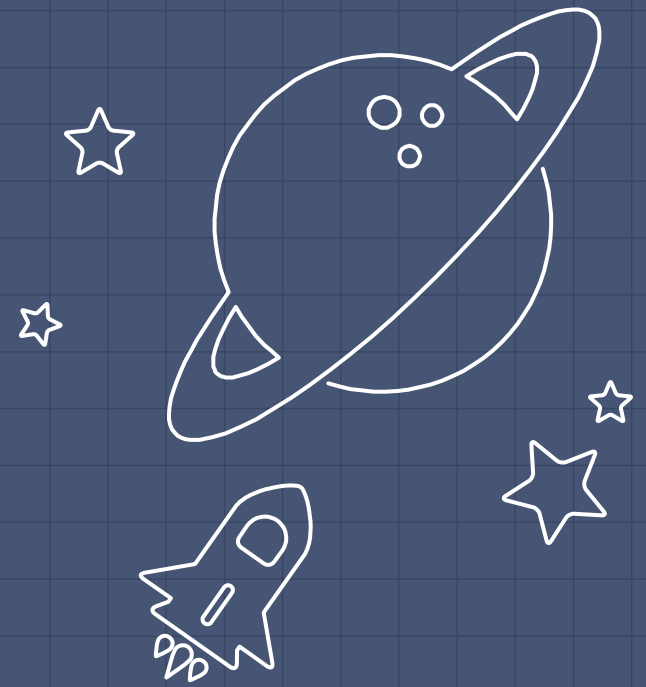
Inference 2:

- The proportion of hit and run cases is higher on Saturday and Sunday.
- The analysis by hours of day reveals that Hit and run cases are higher in late hours of day (10pm to 4am).
- This could be due to following reasons:
  - Lack of lighting
  - Lack of traffic monitoring at night
  - Increased activity on holidays
- **Suggestion:** Chicago Police Department could arrange for improved supervision during night hours and holidays.

Insight 3

Use Machine Learning “Classification Model” to classify severity of crashes in Chicago City based on following parameters:

- Weather Condition
- Lighting Condition
- Most Severe Injury
- Time of Crash/Crash Hour



Creating new data frame with columns - crash day of week and total injuries

In [29]:
```
df['MOST_SEVERE_INJURY'] = df['MOST_SEVERE_INJURY'].apply(lambda x: 0 if x == 'NO INDICATI
```

In [30]:
```
df['WEATHER_CONDITION'] = df['WEATHER_CONDITION'].apply(lambda x: 0 if x == 'CLEAR' else 1
```

In [31]:
```
df.LIGHTING_CONDITION = df.LIGHTING_CONDITION.replace('DAYLIGHT',1)
df.LIGHTING_CONDITION = df.LIGHTING_CONDITION.replace('DARKNESS, LIGHTED ROAD',0)
df.LIGHTING_CONDITION = df.LIGHTING_CONDITION.replace('DARKNESS',0)
df.LIGHTING_CONDITION = df.LIGHTING_CONDITION.replace('DUSK',0.5)
df.LIGHTING_CONDITION = df.LIGHTING_CONDITION.replace('DAWN',0.5)
```

In [32]:
```
df = df[df.LIGHTING_CONDITION != 'UNKNOWN'][['MOST_SEVERE_INJURY','CRASH_HOUR','WEATHER_CO
```

Taking out the dependent variable "Crash Day" from X

In [41]:
```
X = df_ml.drop('LIGHTING_CONDITION',axis=1)
Y = df_ml.LIGHTING_CONDITION
Y = Y.astype('int')
```

Building the Classification Tree

In [42]:
```
dt = tree.DecisionTreeClassifier(max_depth=5)
dt.fit(X,Y)
```

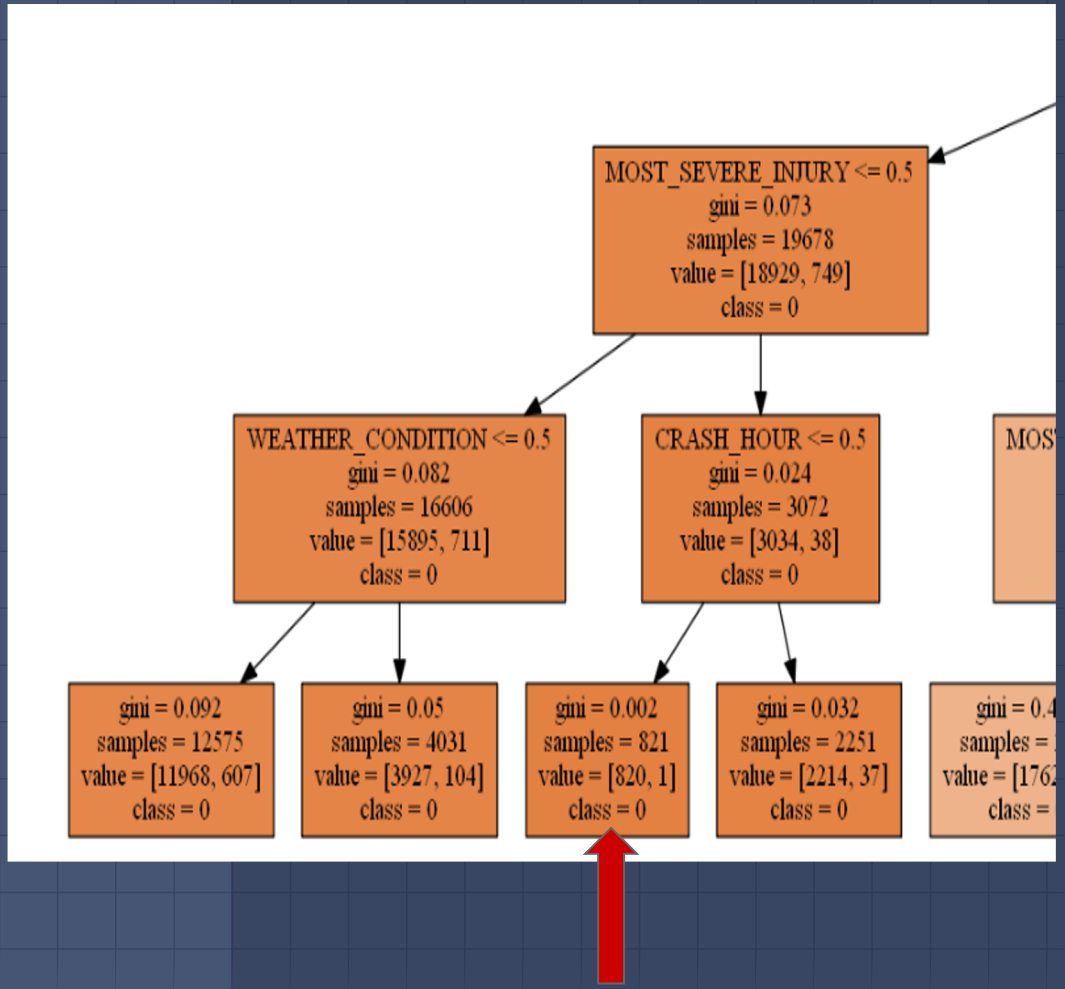
Out[42]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=5,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

This code will visualize a decision tree dt, trained with the attributes in X and the class labels in Y

In [43]:
```
dt_feature_names = list(X.columns)
dt_target_names = [str(s) for s in Y.unique()]
tree.export_graphviz(dt, out_file='tree.dot',
    feature_names=dt_feature_names, class_names=dt_target_names,
    filled=True)
graph = pydotplus.graph_from_dot_file('tree.dot')
Image(graph.create_png())
```

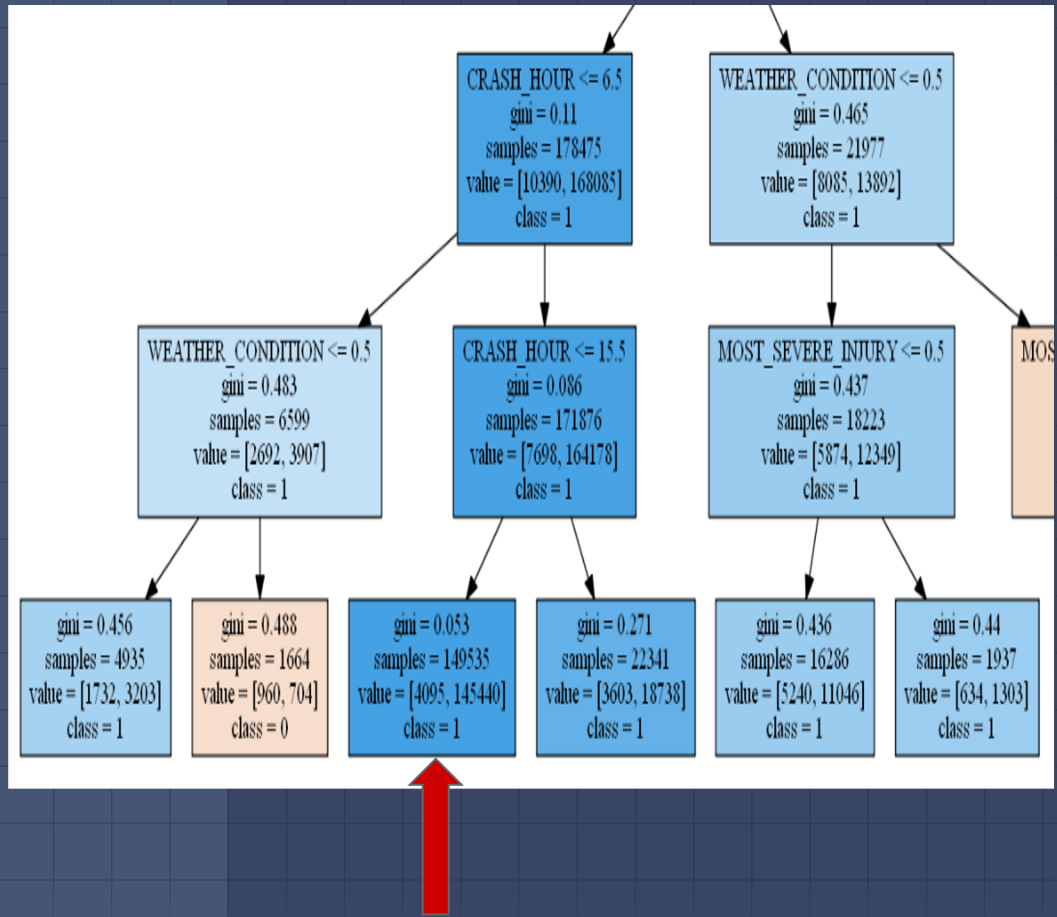
Inference

- Despite Good lighting conditions, ~ quarter (27%) of severe crashes occur around midnight
- **Reasons:** Tired/Distracted driving, over-speeding due to reduced traffic, assuming ideal driving conditions, driving under influence.
- **Suggestion:** increased patrolling & resources around midnight



Inference

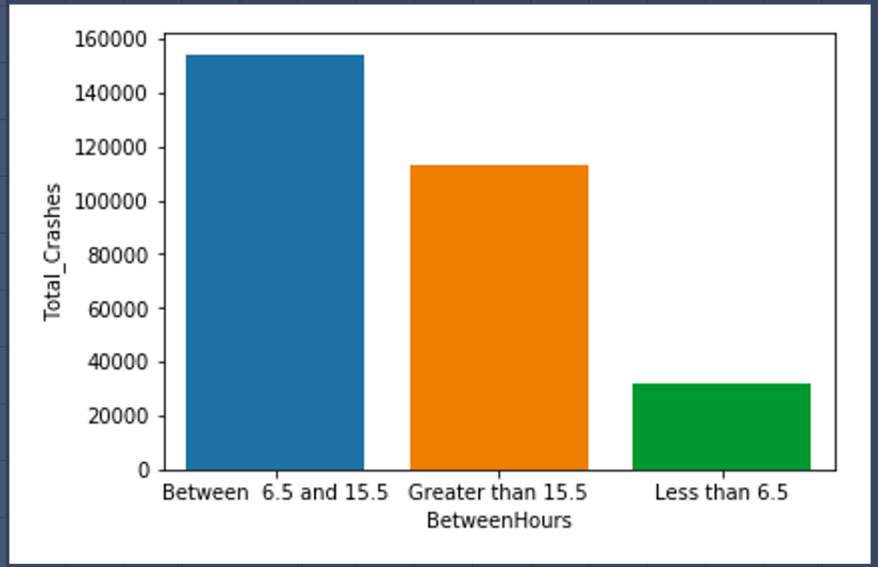
- Almost 50% of the crashes that took place in Chicago between 2013 to 2019 have occurred in daylight with poor lighting conditions
- **Suggestion**: Make daytime headlight mandatory in poor lighting condition hours/zones



Validation

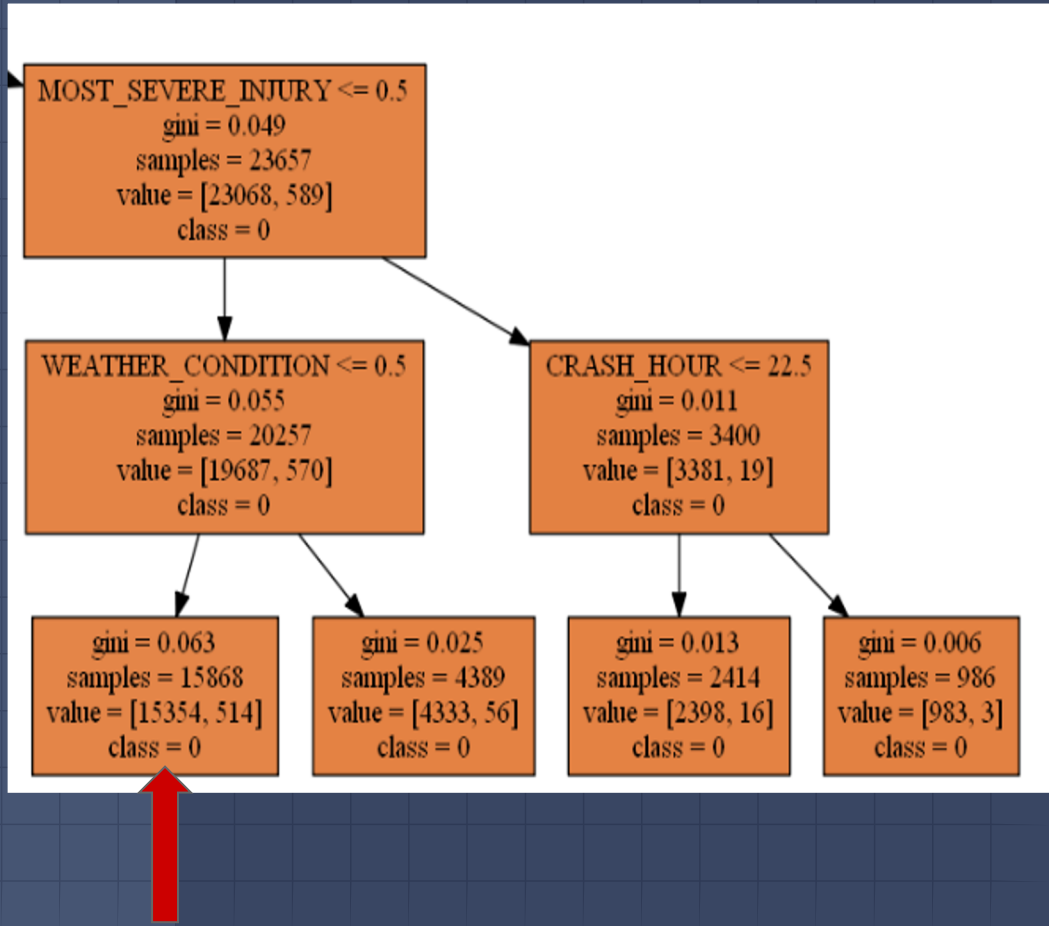
Most number of crashes occur in Chicago City during daylight hours (between 6:30am to 3:15pm)

BetweenHours	Total_Crashes
Between 6.5 and 15.5	154319
Greater than 15.5	112789
Less than 6.5	31987



Inference

- 5% of total severe traffic crashes in Chicago City occur despite good weather & lighting conditions.
- **Reasons**: Poor driving
- **Suggestion**: Create awareness on defensive driving, improve licensing procedures



References

- https://data.cityofchicago.org/Transportation/Traffic-Crashes-Crashes/85ca-t3if/data

  https://stackoverflow.com

  https://seaborn.pydata.org

  https://pandas.pydata.org/pandas-docs/stable/

  https://www.slidescarnival.com

  https://freshspectrum.com/how-pictures-help-discover-the-unexpected/

  http://www.animatedimages.org/cat-cleaning-1095.htm

THANK YOU

Any Questions?