Chicago Car Crashes 2019

Deepthi Vaddi
Cooper Hathaway
Lea Delacruz-Tordjeman

Problem Statement

Analyzing data from car crashes in Chicago in 2019 to determine what are the highest contributing factors to a severe injury in result of a car crash. Using these contributing factors to make recommendations for local drivers and local government to make the roads safer, as well as to make recommendations for self-driving car companies.

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

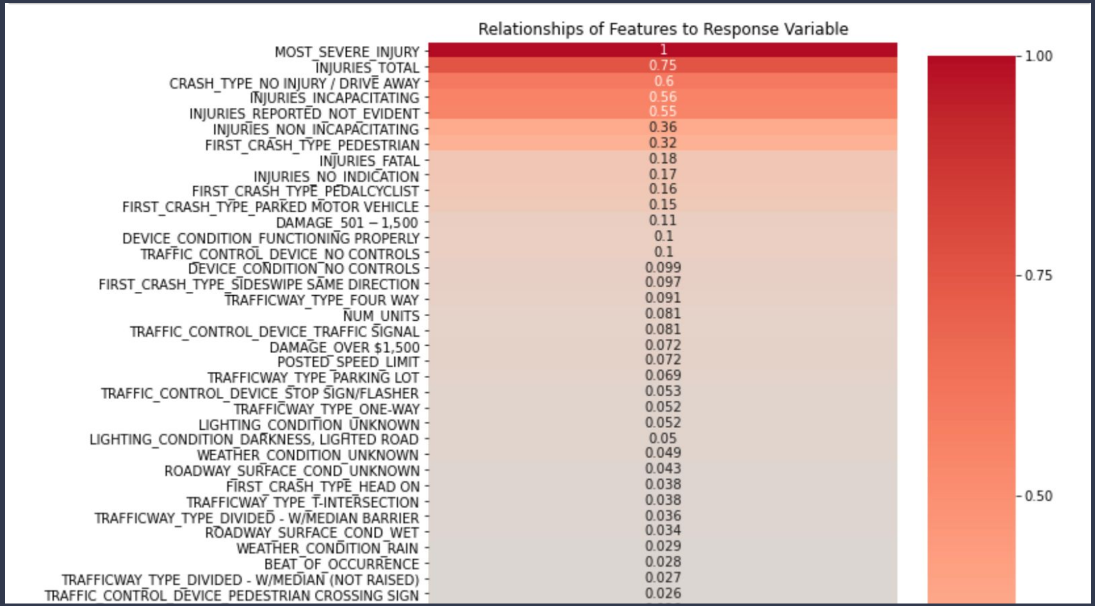
Strategy Outline

- Discussion of how to structure the data and use of like term assignments
- Consider types of classifiers relevant and applicable
- Clean data / perform EDA + Data Visualization / Heat-map for feature selection / Revisions
- Export CSV for group work sync
- Divide classifier modeling load / test models / handle class imbalances when necessary
- Test different classifiers and measure performance based on agreed upon metrics
- Discuss results and plan out next steps / Make recommendations

Data Cleaning

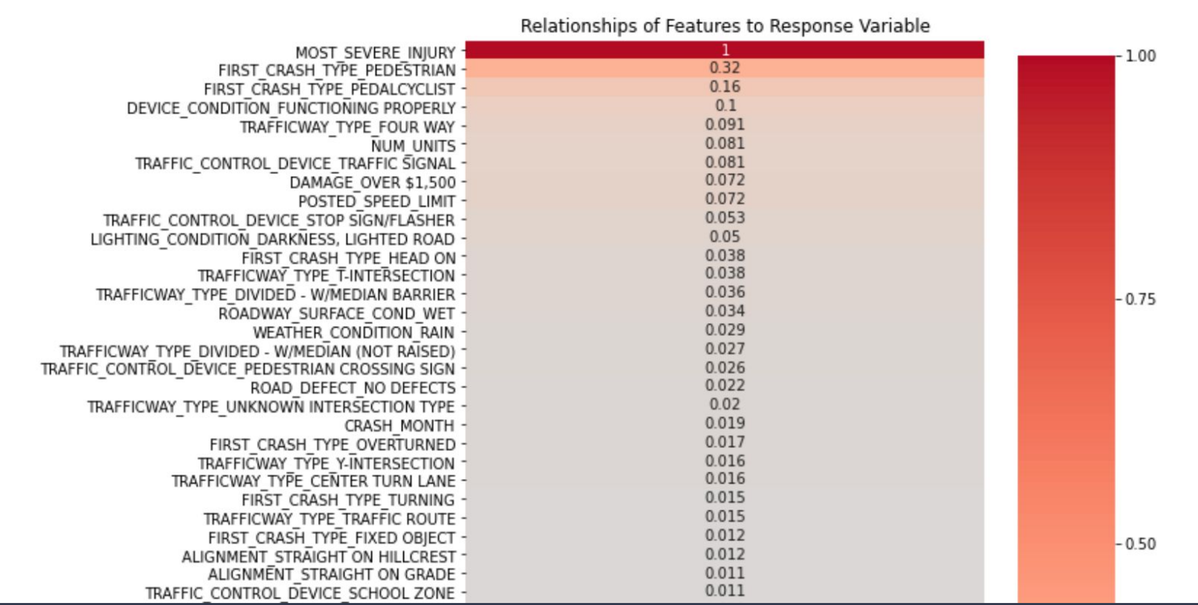
- Large Dataset (250 mb) (Had to narrow down)
- Removal of Features
  - Data Leakage
  - Interdependence of Features
  - Substandard Predictive Power / Data Type
  - Too Many Categories
- Making Sense of Ordinal Data
- Handling Severe Class Imbalance (90% No Injuries / 10% Severe Injuries)
  - SMOTE / Near Miss / Random Over-Sampler
- Numerous Missing Values (some features mostly NaNs)
- Imputing Categorical Data
- One Hot Encoding Categorical Features worth Keeping
- Manual One Hot Encoding Response Variable (for Supervised Learning)
- 5 categories mapped down to binary classification (Severe Injuries vs non-severe)

Data Cleaning: Data Leakage



- Clear signs of Data Leakage
- Injury outcome data to predict injury outcomes
- Interdependence of Features (Multicollinearity)
- Substandard Predictive Power of Many Features

Data Cleaning: Predictive Strength



Data Cleaning: Imputing Category NaN’s

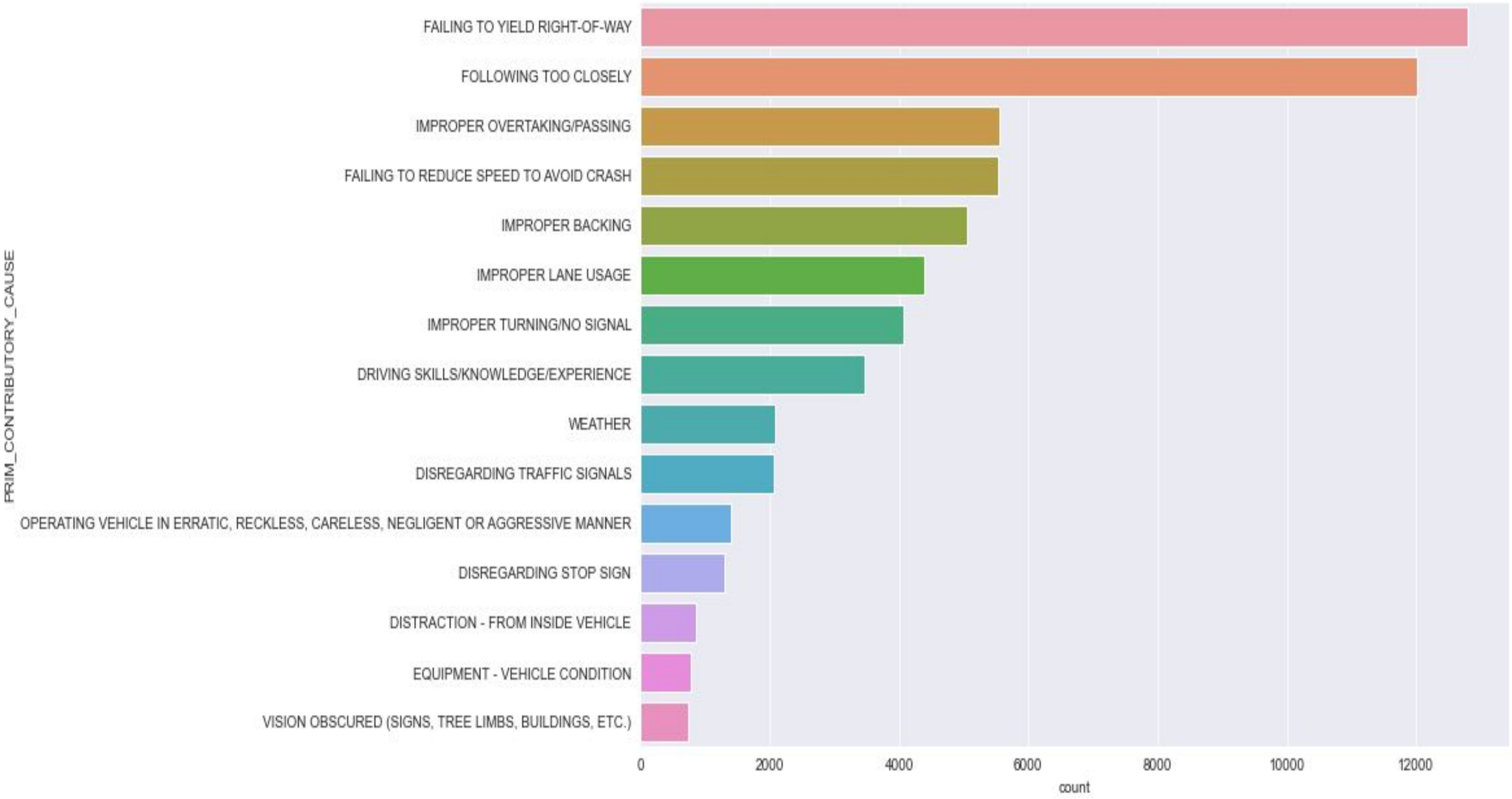
```
def impute_columns(dataset,columnslist):
    for column in columnslist:
        dataset[column] = dataset[column].map(lambda x: np.random.choice(dataset
    return dataset

crashes2019 = impute_columns(crashes2019,['MOST_SEVERE_INJURY',
                                          'INJURIES_TOTAL',
                                          'INJURIES_FATAL',
                                          'INJURIES_INCAPACITATING',
                                          'INJURIES_NON_INCAPACITATING',
                                          'INJURIES_REPORTED_NOT_EVIDENT',
                                          'INJURIES_NO_INDICATION'])
```

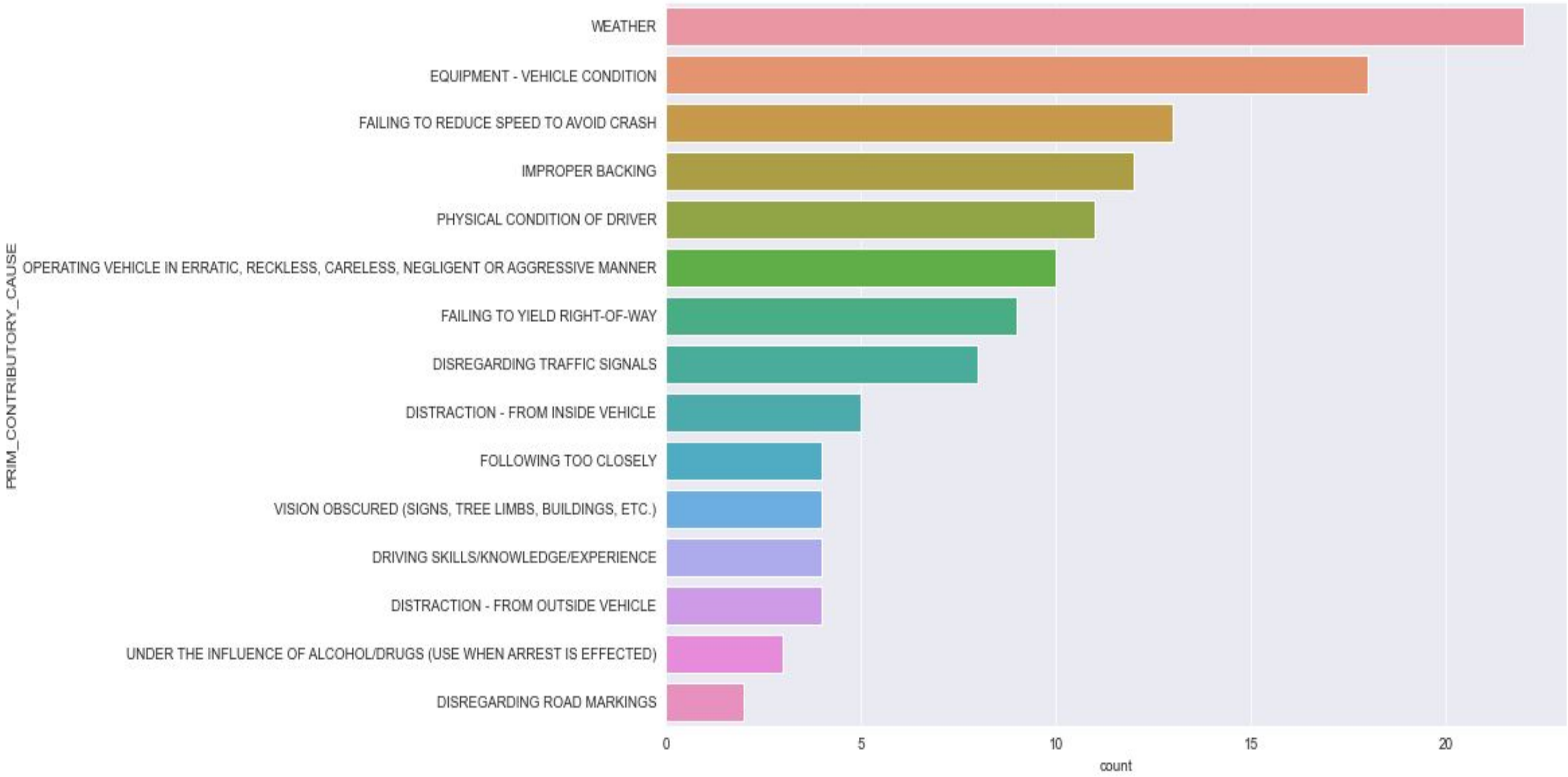
In order to properly perform our EDA and data visualization, we needed to handle instances of missing categorical values and NaNs in general. This function was imperative to maintaining the integrity of the categorical features, by filling with random.choice based on class balance / distribution.

Exploratory Data Analysis

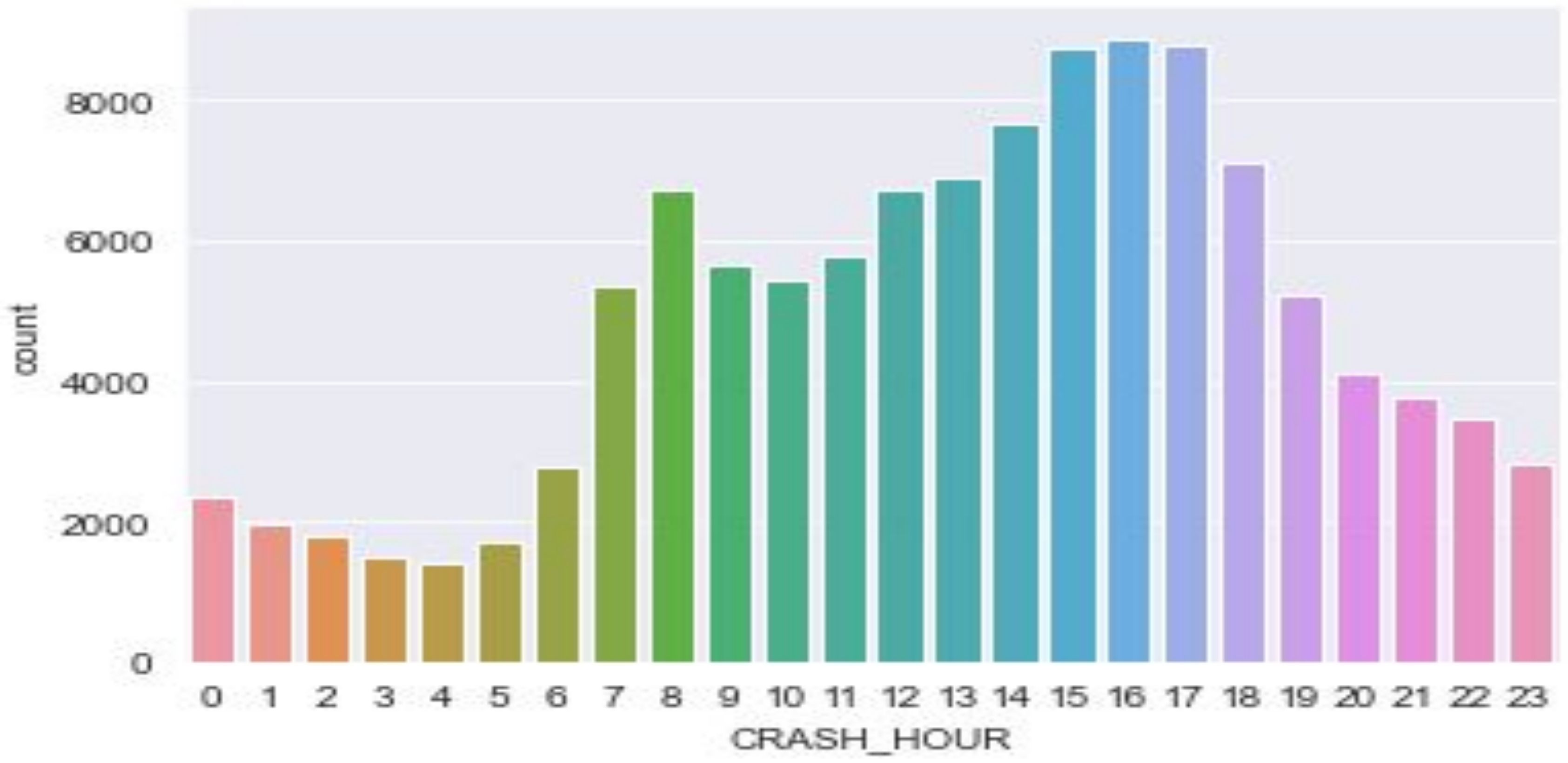
Top 15 Causes for a Crash



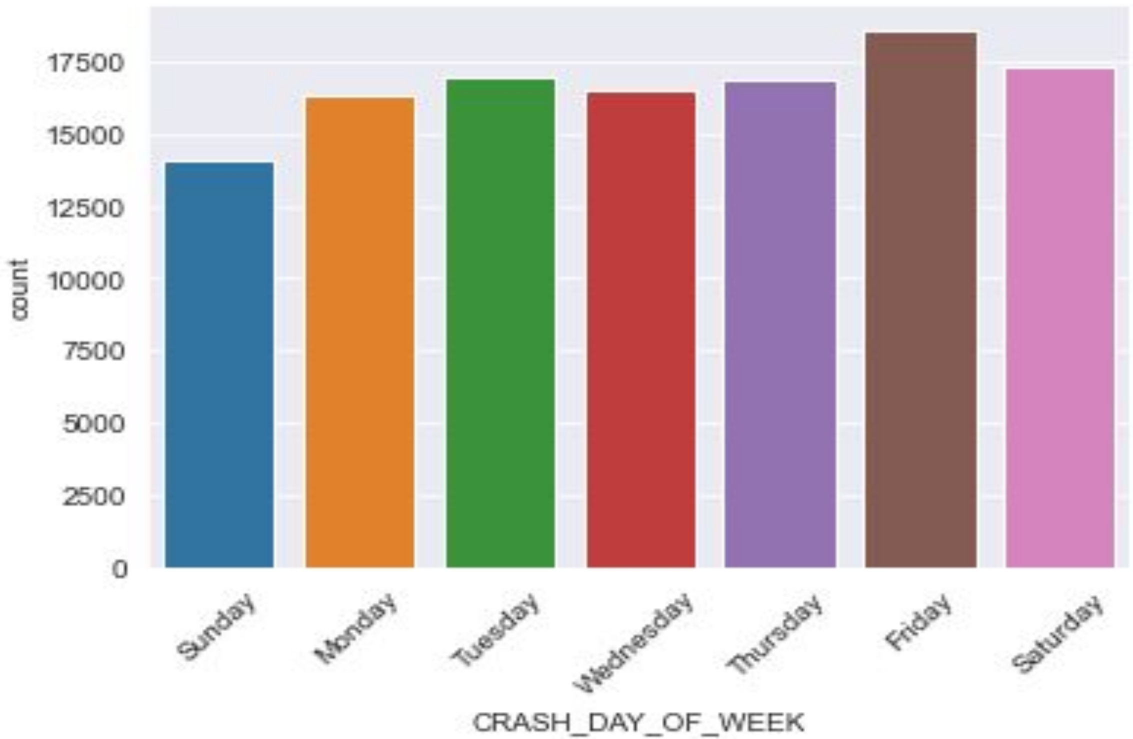
Leading causes for Fatal Crashes



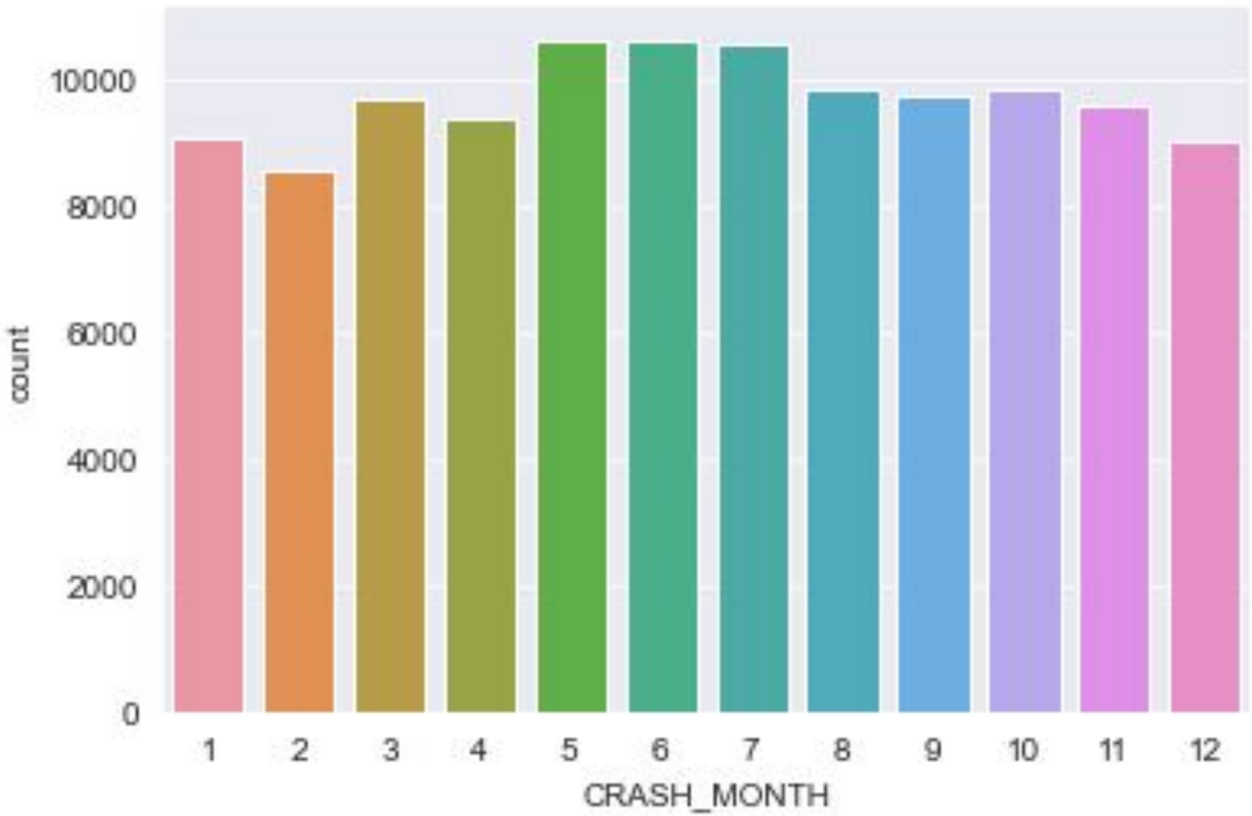
Crashes by Time of Day



Crashes by Day Of Week



Crashes by Month



Models and Imbalanced Class Handling

No Class Balancing:

Logistic Regression:

- **Train Set Accuracy**: 0.918
- **Test Set Accuracy**: 0.917
- **Recall**: 0.287

Ada Boost:

- **Train Set**: 0.931
- **Test Set**: 0.923

Baseline Accuracy:

- **Plurality Class:**
  - **0 - 0.903**
  - **1 - 0.962**

Near Miss:

Under-Sampling the Majority Class

Logistic Regression:

- **Train Set**: 0.886
- **Test Set**: 0.334
- **Recall**: 84%

Random Forest:

- **Train Set**: 0.984
- **Test Set**: 0.346
- **Recall**: 84%

Gradient Boosting Classifier:

- **Train Set**: 0.862
- **Test Set**: 0.376
- **Recall**: 83%

Random OverSampler:

Over-Sampling the Minority Class

Logistic Regression:

- **Train Set**: 0.753
- **Test Set**: 0.771
- **Recall**: 71%

Random Forest:

- **Train Set**: 0.998
- **Test Set**: 0.900
- **Recall**: 33%

Gradient Boosting Classifier:

- **Train Set**: 0.757
- **Test Set**: 0.761
- **Recall**: 73%

SMOTE:

Synthetic Minority Over-sampling Technique

Logistic Regression:

- **Train Set**: 0.753
- **Test Set**: 0.768
- **Recall**: 72%

Random Forest:

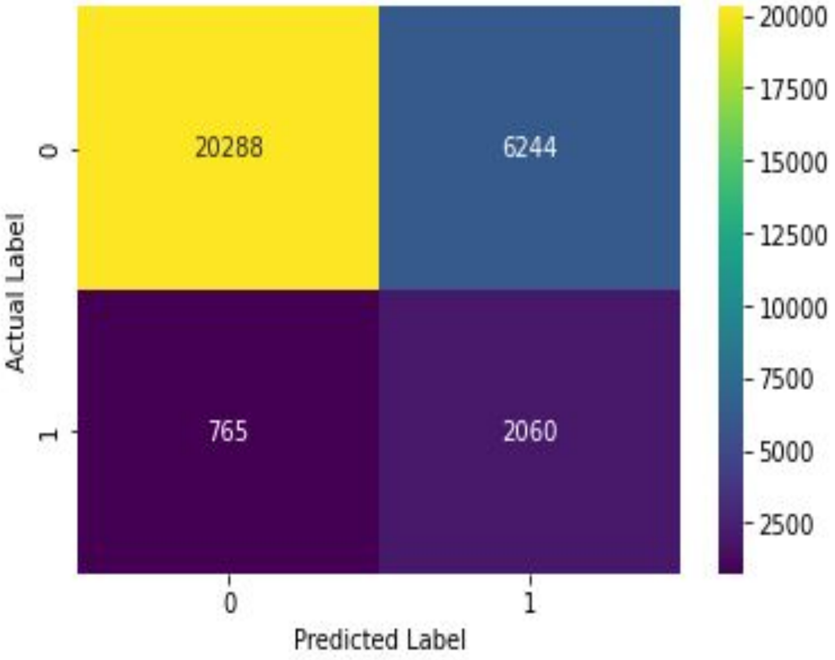
- **Train Set**: 0.998
- **Test Set**: 0.902
- **Recall**: 30%

Gradient Boosting Classifier:

- **Train Set**: 0880
- **Test Set**: 0.845
- **Recall**: 54%

Best Model with a Sampling Technique

- Recall for Injury: 73%
- Recall for No Injury: 76%
- Training set: 0.757
- Testing set: 0.761



Unsupervised:

K-Means

K = 2 gave the best Silhouette score.

It seems to strongly group by injuries.

But no clear difference was seen amongst other features

cluster	0	1
POSTED_SPEED_LIMIT	28.274437	29.658109
NUM_UNITS	2.015842	2.135112
MOST_SEVERE_INJURY	0.00019	0.876289
CRASH_HOUR	13.120324	13.182735
CRASH_DAY_OF_WEEK	4.118832	4.117971
CRASH_MONTH	6.514302	6.759094
INJURIES_TOTAL	0.033539	1.464981
INJURIES_FATAL	0.000000	0.008377
INJURIES_INCAPACITATING	0.000000	0.184053
INJURIES_NON_INCAPACITATING	0.000019	0.977895
INJURIES_REPORTED_NOT_EVIDENT	0.033520	0.294656
INJURIES_NO_INDICATION	2.121252	1.3736

Best Classifier and Summary

Demonstration

We as a team had a goal of discovering relevant and applicable insights into the world of vehicular accidents. With these insights, we hope to inform local governments with information that could lead to better commuting recommendations, preventative laws, and data driven policies. We also see a great deal of potential application of our classifiers with private enterprise in the driverless vehicle industry, regarding how to build better predictive driving models. The machine learning classifiers that we have developed could contribute to faster reaction times, or when accidents are inevitable, to intentionally make decisions that would reduce the accidents fatality or injury rate based on the accident data the classifier was taught on. The ability for self driving vehicles to make decisions that reduce lethality is invaluable, and we see this work potentially contributing to this growing industry.

Constraints

During this process, we ran into a couple of challenges including:

- Numerous NaNs
- Data Leakage
- Size of Data Set (Time for Model to Fit was Lengthy)
- Severely Imbalanced Data
- Inconsistent data entries - revealed by EDA

Next Steps and Recommendations

- We found that the most crashes take place at 3pm, 4pm, and 5pm, presumably related to rush hour traffic. A strong recommendation would be to try to avoid being on the road at these times, or to carpool so minimize the amount of cars on the road.
- The most fatalities were attributed to weather. When the weather is severe, it would be wise to stay off the road or to wait for conditions to improve to increase commuter safety and chances of survival.
- Local governments and agencies could enforce stricter policies regarding Equipment - Vehicle Condition, the second leading cause of fatalities.
- Third most fatalities were attributed to failure to reduce speed to avoid crashing. This is likely related to texting and driving, or driving while distracted. Local communities could host informational seminars on phone features like Do Not Disturb While Driving, in order to reduce fatal accidents caused by texting and driving.
- While outside features are absolutely a contributor, a significant amount of the crashes are essentially caused by operator error. The focus should be on creating software, features, and policies that encourage more responsible driving behaviors.
- Implement neural nets and other SVM variants to test other model performance
- Compare year 2019 to year 2020 to evaluate how covid has affected accident rates