September 15 2020

Coursera Capstone Project Report

Ferrer García Erick

Table of contents

1. Introduction 3

1.1 Main problem 3

1.2 Data understanding 3

1.3 Methodology 3

2. Data analysis 4

2.1 Overview of the data 4

2.2 Cleaning the data 4

3. Modeling 10

3.1 Decision Tree 10

3.2 KNN 12

3.3 Logistic Regression 13

4. Results 14

5. Conclusions 14

1. Introduction

For this project we have to find a dataset about car accidents and predict the severity of an accident due to it's features so I find a dataset called ***US_Accidents_June20.csv*** that complies some requirements like this:

1. Must have 'severity' as attribute.
2. It's big enough.
3. The main objective of the dataset is predict the severity.

1.1 Main problem

With this dataset i have the data enough, so now defining the problem, is predict the severity of a car accident, so we can create a solution given the factors that affect the severity of itself, this can safe lives, money, insurance and time wasted on accidents.

1.2 Data understanding

This is a countrywide car accident dataset, which covers 49 states of the USA. The accident data are collected from February 2016 to June 2020, using two APIs that provide streaming traffic incident (or event) data. These APIs broadcast traffic data captured by a variety of entities, such as the US and state departments of transportation, law enforcement agencies, traffic cameras, and traffic sensors within the road-networks. Currently, there are about 3.5 million accident records in this dataset.
So in this project the dataset fits perfect to the objective of the capstone project, which is predict the severity, based on the result we can see how we can improve the roads to make them safer.

Link to the dataset on kaggle: https://www.kaggle.com/sobhanmoosavi/us-accidents

1.3 Methodology

In this project our main target is predict the severity of an accident, after finish cleaning the data we have 470,000 entries all from the same state california, so now we have to put the steps of the process to modeling and do predictions.

- First step: We have to collect the required data, that we already have, a dataset about accidents with severity as an attribute, and many others to predict.

- Second step: We need to clean our data since it have missing entries, or noise like the ID attribute which it doest'n help us to predict, then an analisys of the data that is left will be nice to understand which algorithm is the best to do model and predict.

- Third step: The final step will be to notice areas of improvement, which is what could most affect a serious accident and how to avoid it if possible, have feedback from the project qualifiers, and get to write the conclusions.

2. Data analysis

2.1 Overview of the data

First in the project i import the libraries that will be required for the the analysis and plotting of the information. To understand more the data i create a table of the features with the name, description and the type of data. And then print how many data was in the data set before the cleaning.

```
print('The size of this data frame is :' , df.shape)
```

```
The size of this data frame is : (3513617, 49)
```

All the data came of 2 sources, "Bing" and "MapQuest", they have diferent levels of severity, Bing with 4 and MapQuest with 5, giving a look at the data and some kaggle post there are some big differences like:

- The overall duration and impacted distance of accidents reported by Bing are much longer than those by MapQuest.
- Second, same severity level holds different meanings for MapQuest and Bing. MapQuest seems to have a clear and strict threshold for severity level 4, cases of which nevertheless only account for a tiny part of the whole dataset. Bing, on the other hand, doesn't seem to have a clear-cut threshold, especially regards duration.
- The data is more balanced in Bing.

Since a lot of people say that we cannot use both i decide choose the MapQuest data, even when is not the most balanced about their cases of severity, i like it since they are more consistent in the handle of the duration and severity. So we drop the data from Bing, even if this can seem like an error we can performance some techniques to manage the unbalanced data.

2.2 Cleaning the data

Now since we want to predict, everything that happens after the crash is useless, because it already happen, so i drop some variables like end time, end latitude, and that’s the idea everything with “End” has to go. Now on the other hand i have categorical attributes like Country, since this is a dataset about U.S.A the value is always 1 we can drop the feature, the same with Turning_loop.

Now looking in the kaggle repository the people says that there’s a mess on 2 categorical features that we can see in the next image.

In [9]:
```
print("Wind Direction: ", df['Wind_Direction'].unique())
```

```
Wind Direction:  ['Calm' 'SW' 'SSW' 'WSW' 'WNW' 'NW' 'West' 'NNW' 'NNE' 'South' 'North'
 'Variable' 'SE' 'SSE' 'ESE' 'East' 'NE' 'ENE' 'E' 'W' nan 'S' 'VAR'
 'CALM' 'N']
```

In [10]:
```
# show distinctive weather conditions

weather ='!'.join(df['Weather_Condition'].dropna().unique().tolist())

weather = np.unique(np.array(re.split(
    "!|\s/\s|\sand\s|\swith\s|Partly\s|Mostly\s|Blowing\s|Freezing\s",
    weather))).tolist()

print("Weather Conditions: ", weather)
```

```
Weather Conditions:  ['', 'Clear', 'Cloudy', 'Drizzle', 'Dust', 'Dust Whirlwinds', 'Fair', 'Fog', 'Funnel Cloud', '
Hail', 'Haze', 'Heavy ', 'Heavy Drizzle', 'Heavy Ice Pellets', 'Heavy Rain', 'Heavy Rain Showers', 'Heavy Sleet', '
Heavy Smoke', 'Heavy Snow', 'Heavy T-Storm', 'Heavy Thunderstorms', 'Ice Pellets', 'Light ', 'Light Drizzle', 'Ligh
t Fog', 'Light Hail', 'Light Haze', 'Light Ice Pellets', 'Light Rain', 'Light Rain Shower', 'Light Rain Showers', '
Light Sleet', 'Light Snow', 'Light Snow Grains', 'Light Snow Shower', 'Light Snow Showers', 'Light Thunderstorm', '
Light Thunderstorms', 'Low Drifting Snow', 'Mist', 'N/A Precipitation', 'Overcast', 'Partial Fog', 'Patches of Fog'
, 'Rain', 'Rain Shower', 'Rain Showers', 'Sand', 'Scattered Clouds', 'Shallow Fog', 'Showers in the Vicinity', 'Sle
et', 'Small Hail', 'Smoke', 'Snow', 'Snow Grains', 'Snow Showers', 'Squalls', 'T-Storm', 'Thunder', 'Thunder in the
Vicinity', 'Thunderstorm', 'Thunderstorms', 'Tornado', 'Volcanic Ash', 'Widespread Dust', 'Windy', 'Wintry Mix']
```

As you can see we can blend some of this attributes like rain and storm, and that's exactly what i do to reduce the amount of disorder and data in this 2 features.

- First the Wind_Direction.

In [11]:
```
df.loc[df['Wind_Direction']=='Calm','Wind_Direction'] = 'CALM'
df.loc[(df['Wind_Direction']=='West')|(df['Wind_Direction']=='WSW')|(df['Wind_Direction']=='WNW'),'Wind_Direction']
df.loc[(df['Wind_Direction']=='South')|(df['Wind_Direction']=='SSW')|(df['Wind_Direction']=='SSE'),'Wind_Direction']
df.loc[(df['Wind_Direction']=='North')|(df['Wind_Direction']=='NNW')|(df['Wind_Direction']=='NNE'),'Wind_Direction']
df.loc[(df['Wind_Direction']=='East')|(df['Wind_Direction']=='ESE')|(df['Wind_Direction']=='ENE'),'Wind_Direction']
df.loc[df['Wind_Direction']=='Variable','Wind_Direction'] = 'VAR'
print("Wind Direction after simplification: ", df['Wind_Direction'].unique())
```

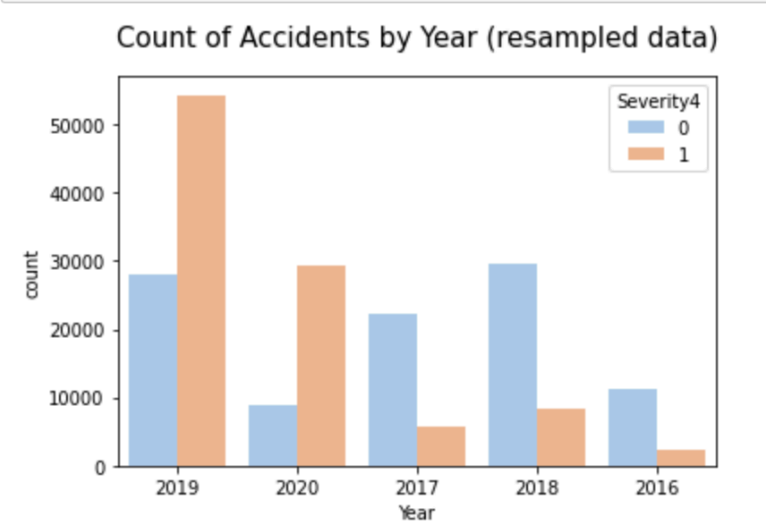
```
Wind Direction after simplification:  ['CALM' 'SW' 'S' 'W' 'NW' 'N' 'VAR' 'SE' 'E' 'NE' nan]
```

- Second the Wind_Direction by blend some of them, like Cloud and Overcast, or Rain and Storm.

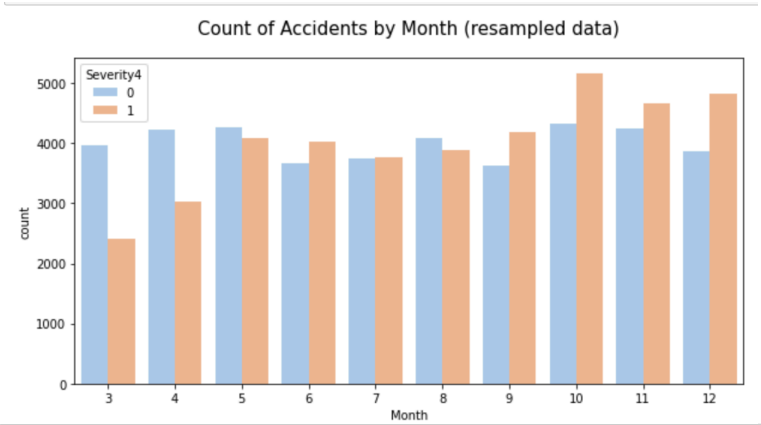
In [12]:
```
df['Clear'] = np.where(df['Weather_Condition'].str.contains('Clear', case=False, na = False), 1, 0)
df['Cloud'] = np.where(df['Weather_Condition'].str.contains('Cloud|Overcast', case=False, na = False), 1, 0)
df['Rain'] = np.where(df['Weather_Condition'].str.contains('Rain|storm', case=False, na = False), 1, 0)
df['Heavy_Rain'] = np.where(df['Weather_Condition'].str.contains('Heavy Rain|Rain Shower|Heavy T-Storm|Heavy Thunder
df['Snow'] = np.where(df['Weather_Condition'].str.contains('Snow|Sleet|Ice', case=False, na = False), 1, 0)
df['Heavy_Snow'] = np.where(df['Weather_Condition'].str.contains('Heavy Snow|Heavy Sleet|Heavy Ice Pellets|Snow Show
df['Fog'] = np.where(df['Weather_Condition'].str.contains('Fog', case=False, na = False), 1, 0)

# Assign NA to created weather features where 'Weather_Condition' is null.

weather = ['Clear','Cloud','Rain','Heavy_Rain','Snow','Heavy_Snow','Fog']
for i in weather:
  df.loc[df['Weather_Condition'].isnull(),i] = df.loc[df['Weather_Condition'].isnull(),'Weather_Condition']
```

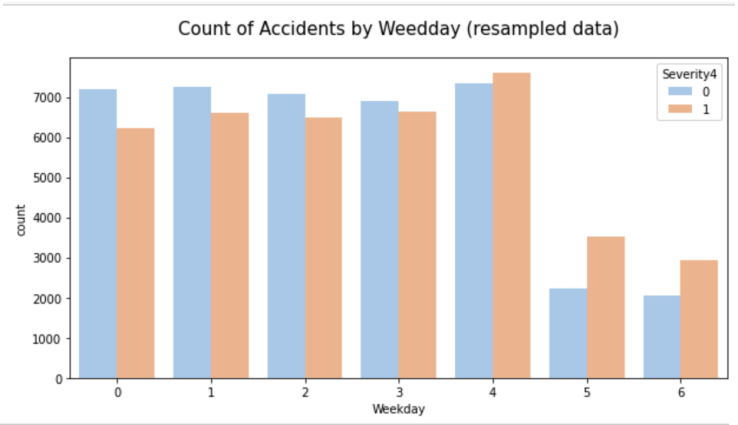
As we can see, we simplify the 2 variables this prevent the noise in the future or weird outputs, after this we just have to fix the date, and drop all the missing values that we can't change for and average of mean, at the end of this part it will be a table of all the attributes dropped and why it is dropped, the last but no less is fixing the part of the time, for this i use the function pd.to_datetime, so now it's in the right format.

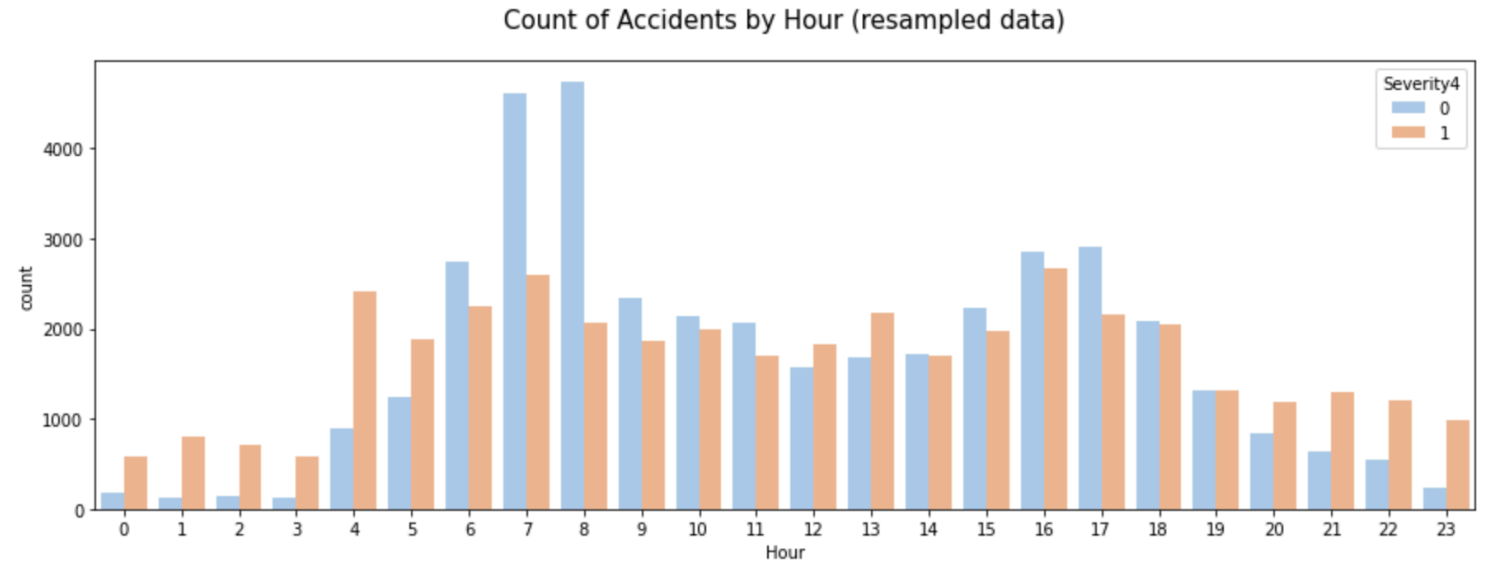
In the first plot we made is all about the accidents and the year that happend, in 2019 and 2020 we can clearly see that the severity is bigger than in years before, this is not specified but maybe the MapQuest change their metric for this accidents, so we need to change this, because the data can have outliers that make our predictions worst, so i drop those 2 years, to avoid problems, since the data set is still big enough for make predictions.



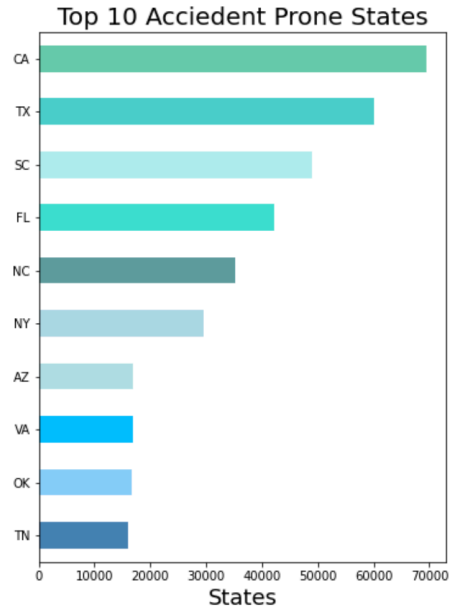
Now on the side of the month, day and hour, as we can see that the months with more accidents are the 3 last months, and the less are the first 3, in week day we have that between the week are more accidents, but the severity if high in the weekends, and with the hour, most accidents happen in the morning, when the people is going to work or the school, but the severity is high after the 8 P.M. to 4 P.M. i guess when the streets are emptier the people tend to go faster than the average, plus the darkness of the night the people is proner to have a worst accident.

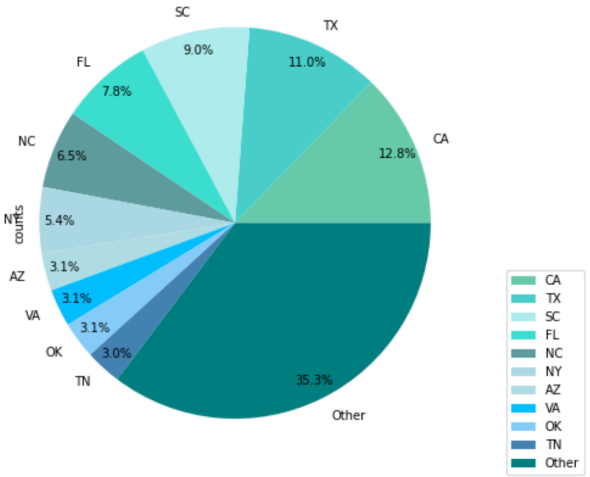




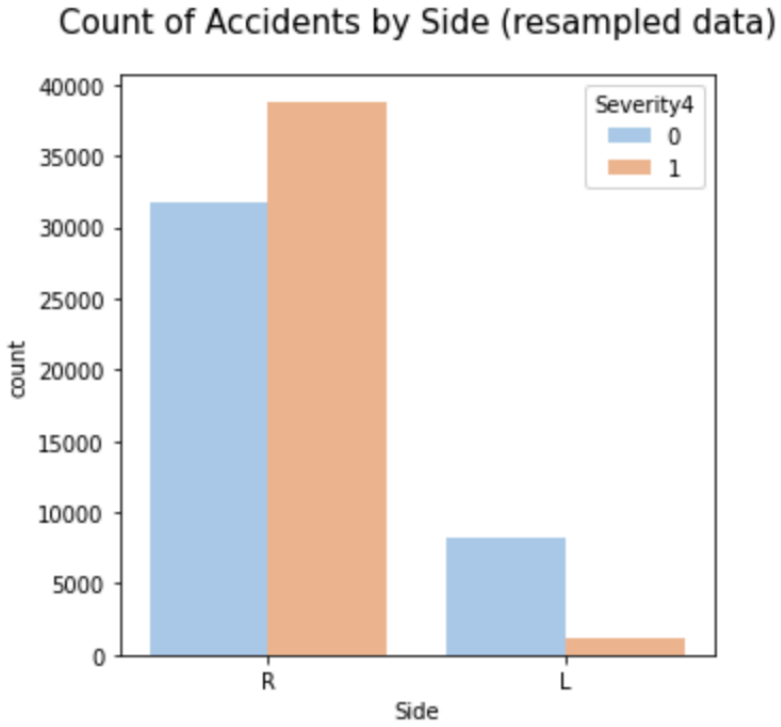


Now we can see the top 10 acciedent prone states, i haven't been in all the cities but i think this mostly happened because the design of the city, and the population.

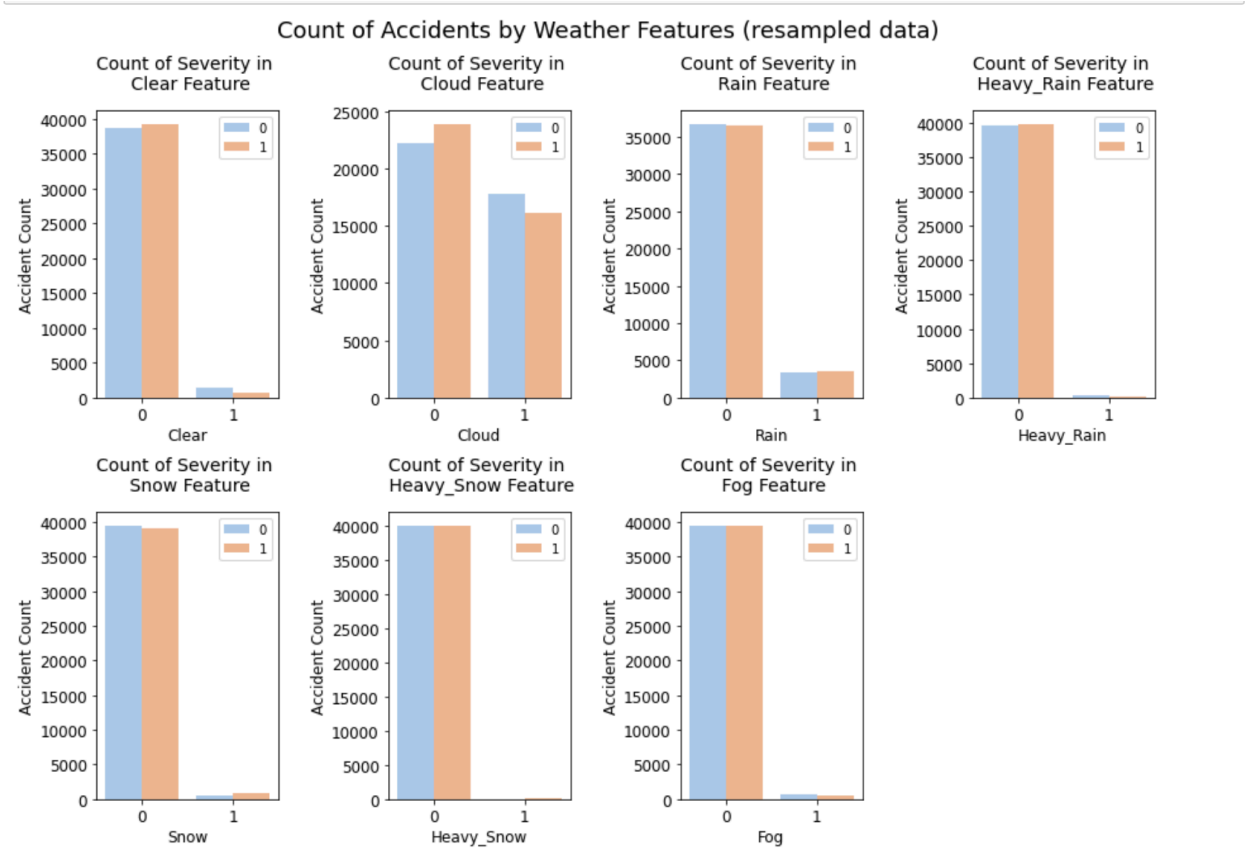




Now on the part of the side we can see that the right side is by far the most dangerous.



After normalize the data about weather, lets see the last plot that i considered important to understand the data and it's the weather.



As we can see mostly happen in clear, heavy rain, heavy snow, snow and fog so this give us info like, maybe the weather doesn't matter that much when an accident happen.
And finally the table of drop features and why:

Feature	Data type	Why
Bump	Object	It can be an outlier
Give_Way	Object	It can be an outlier
No_Exit	Object	It can be an outlier
Roundabout	Object	It can be an outlier
Traffic_Calming	Object	It can be an outlier
Population_County_log	Object	It can be an outlier
City_Freq	Object	It can be an outlier

Feature	Data type	Why
Side	Object	Since almost all the accidents happen in the right side let's asume it will happen there.
Sunrise_Sunset	Object	Doesn't give us much info.
Civil_Twilight	Object	Doesn't give us much info.
Nautical_Twilight	Object	Doesn't give us much info.
Astronomical_Twilight	Object	Doesn't give us much info.

3. Modeling

Now let go with the modeling, since its labeled data we have 2 options:

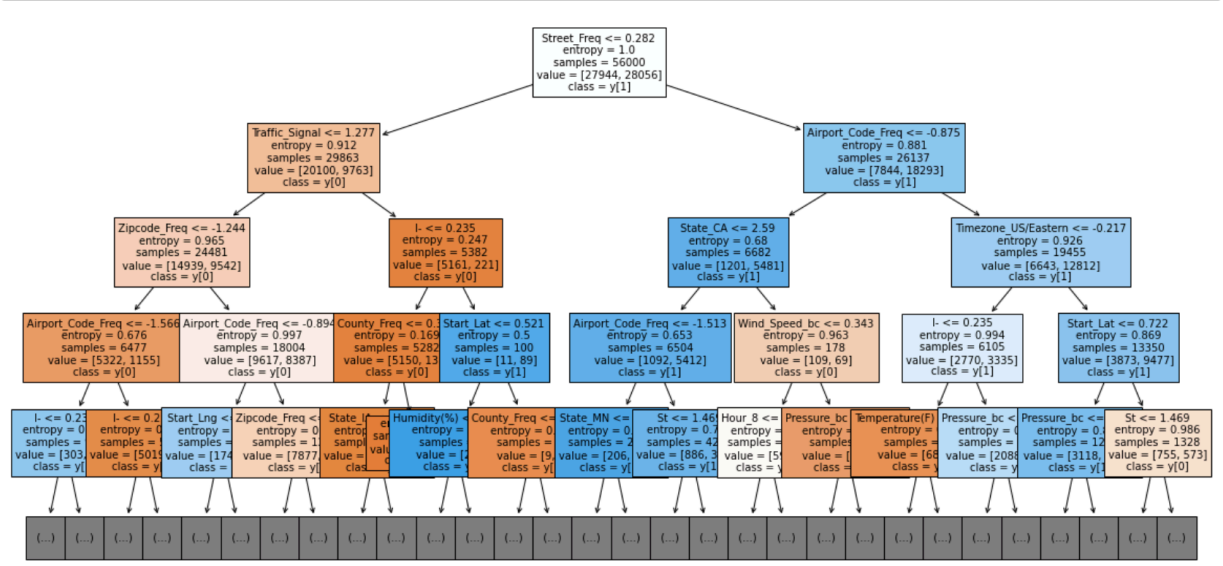
- Classification: predict a class or a discrete value
- Regression: predict a continuous value

So lets choose classification, since severity is a discrete value, and we have 3 algorithms to do it KNN, logistic regression and decision tree, but KNN don't work with this kind of values and converting entries like the street to int isn't the best practice of all, so with this in mind lets start with splitting the data.

3.1 Decision Tree

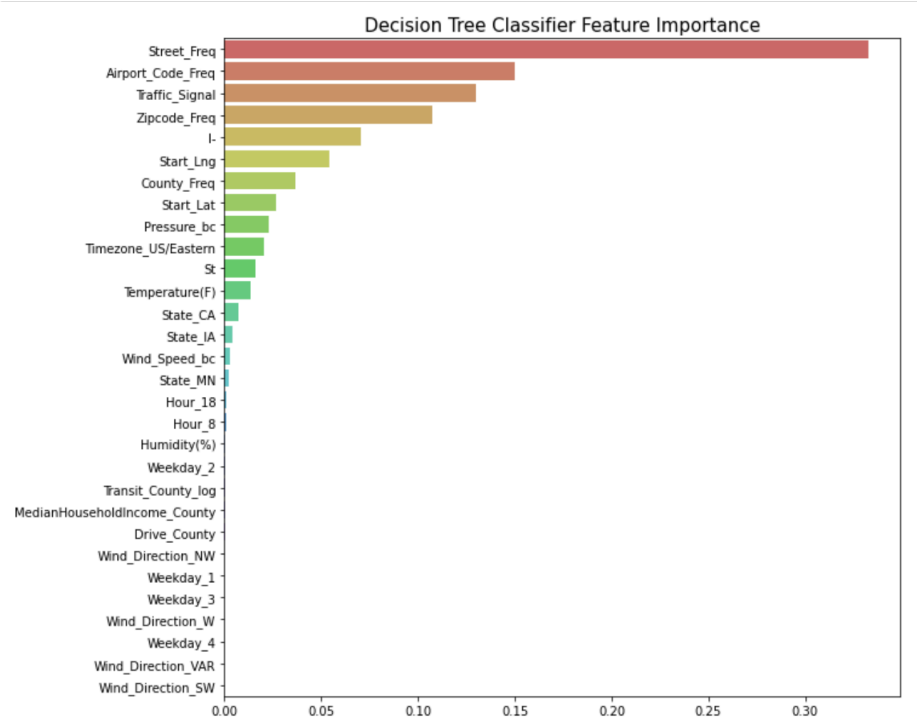
The results of implement this algorithms where this:

	precision	recall	f1-score	support
0	0.77	0.73	0.75	12056
1	0.74	0.78	0.76	11944
accuracy			0.75	24000
macro avg	0.75	0.75	0.75	24000
weighted avg	0.75	0.75	0.75	24000

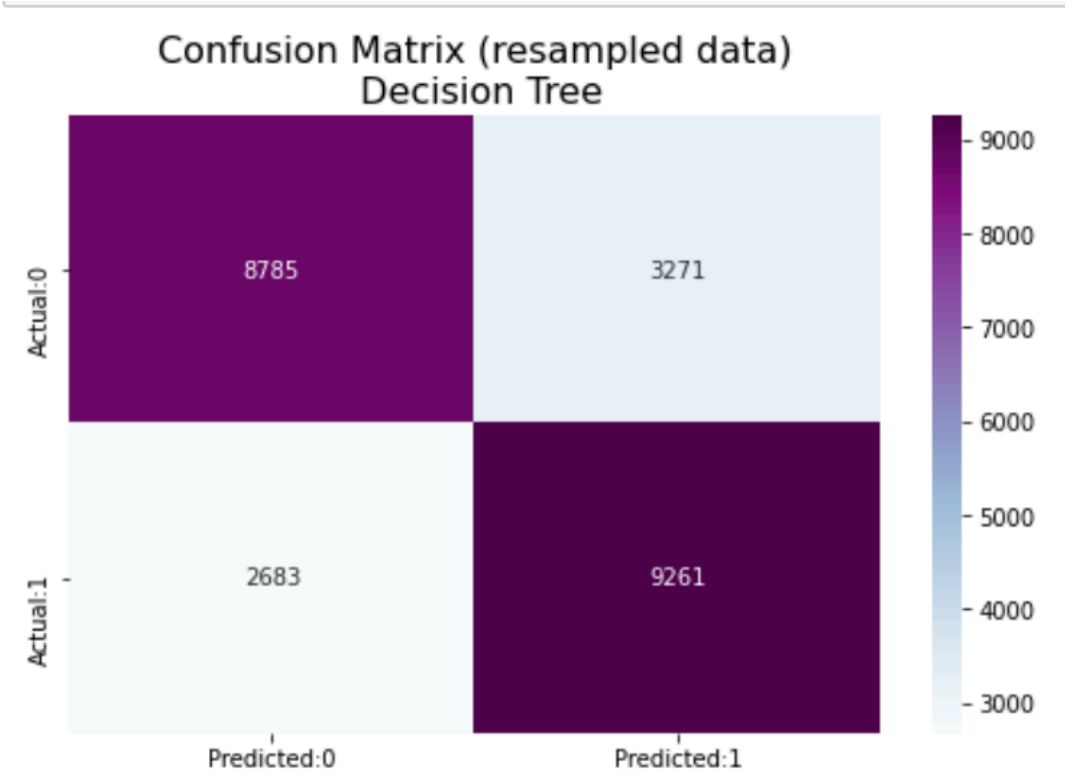


With this plot we can see that Street Frequency is the one that divide the trees this can be explained in another plot, so we can see the features that affect the most in the behavior of the tree.

Let's take the top 5 features here and see that the frequency of a street is the most importan feature, and the kind of street is a interstate which we can do some affirmations like if an interstate has a lot of traffic the probability of an accident increment.



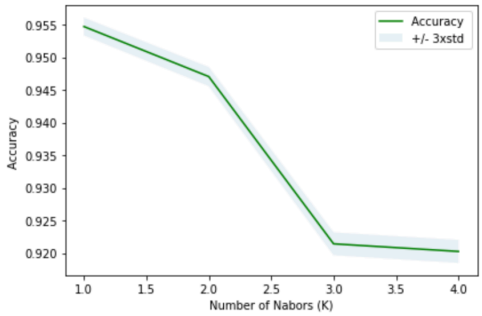
Now in the final part the matrix of confusion, to reaffirm the precision.



3.2 KNN

Using this algorithm i considere is a bad idea since it's based on distances and basically we just have 2, that's the why on the plot there's a fall on 2 to 3, also the process is too slow.

In [74]:
```
plt.plot(range(1,Ks),mean_acc,'g')
plt.fill_between(range(1,Ks),mean_acc - 1 * std_acc,mean_acc + 1 * std_acc, alpha=0.10)
plt.legend(('Accuracy ', '+/- 3xstd'))
plt.ylabel('Accuracy ')
plt.xlabel('Number of Nabors (K)')
plt.tight_layout()
plt.show()
```



In [70]:
```
Ks = 5
mean_acc = np.zeros((Ks-1))
std_acc = np.zeros((Ks-1))
ConfustionMx = [];
for n in range(1,Ks):

    #Train Model and Predict
    neigh = KNeighborsClassifier(n_neighbors = n).fit(X_train,y_train)
    yhat=neigh.predict(X_test)
    mean_acc[n-1] = metrics.accuracy_score(y_test, yhat)


    std_acc[n-1]=np.std(yhat==y_test)/np.sqrt(yhat.shape[0])

mean_acc
```

Out[70]:
```
array([0.95470833, 0.94704167, 0.92141667, 0.92025   ])
```

So in this part i use KNN and happen 2 things, the first thing is that is not the best to predict since the best K is 1 and is not really very accurate, the same we can see in the first plot that 1 have the highest accuracy, and the second thing that happens is that is really really slow, at least in my computer calculating whit 5 K's the wait was like 1 hour aprox. so i discard this method by convenience.

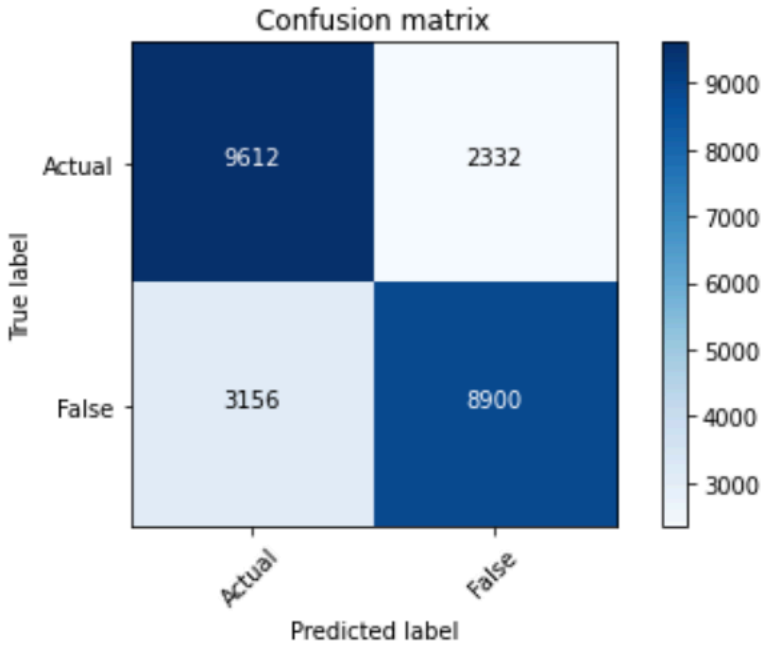
3.3 Logistic Regression

Finally the results of the logistic regression is almost the same than the decision tree so let's see why

	precision	recall	f1-score	support
0	0.79	0.74	0.76	12056
1	0.75	0.80	0.78	11944
accuracy			0.77	24000
macro avg	0.77	0.77	0.77	24000
weighted avg	0.77	0.77	0.77	24000

As we can see the difference between the result are .02 and lets see the confusion matrix. Now in the part of results let's explain why i choose decision tree even when the logistic seem more accurate.

```
Confusion matrix, without normalization
[[9612 2332]
 [3156 8900]]
```



4. Results

As we can see both os them are almost the same, but in this process i will choose decision tree, because it give me more information such as the top 5 features with most impact of an accident like:

* The street frequency
* Airport code
* Traffic signal
* Zipcode
* The kind of street in this case I

So for now, i decide stay with decision tree algorithm, the logistic regression was algo a good algorithm but the information that i can plot about it is less graphic, and the result was almost the same as the decision tree, now on the KNN i already said that i don't think is a good algorithm for this case because even when we normalize the data the result about the best K is still 1, this and the time to process the algorithm give reasons to avoid it.

5. Conclusions

For now i'm going to choose the decision tree for the prediction, and with such info about it, we can implement in a real time system that can prevent this accidents or at least their severity, based on the most important variables, or based on the plots we can say that they happen more often in the morning when people go to work, but the severity of an accident is worst in the night, this information can help also in the logistic of transit police
so they can give more importance to busy streets, at certain times and near certain places, however this project by itself can get better, so maybe improve the relations of the variables will be too a future work and we can predict better based on it.