In [1]:
```
'''
DSC630 - Milestone 4

Finalizing Results

Joel McMillin

October 30, 2022
'''
```

Out[1]: '\nDSC630 - Milestone 4\n\nFinalizing Results\n\nJoel McMillin\n\nOctober 30, 2022\n'

In [69]:
```
#Suppress warnings

import warnings
warnings.filterwarnings("ignore")
```

In [70]:
```
#Importing all libraries up-front:


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import math
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.decomposition import PCA
from sklearn import tree
from sklearn import datasets
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn import linear_model
from sklearn.model_selection import cross_val_predict
from sklearn.feature_selection import VarianceThreshold
from sklearn.feature_selection import chi2
from sklearn.feature_selection import SelectKBest
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.tree import export_text
```

In [71]:
```
#Loading our data

df = pd.read_csv('US_Accidents_Dec21_updated.csv')
```

In [72]:
```
df.head()
```

Out[72]:

	ID	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	Description	...	Roundabout	Station	Stop	Traffic_Calming	Traffic_Signal
0	A-1	3	2016-02-08 00:37:08	2016-02-08 06:37:08	40.108910	-83.092860	40.112060	-83.031870	3.230	Between Sawmill Rd/Exit 20 and OH-315/Olentang...	...	False	False	False	False	False
1	A-2	2	2016-02-08 05:56:20	2016-02-08 11:56:20	39.865420	-84.062800	39.865010	-84.048730	0.747	At OH-4/OH-235/Exit 41 - Accident.	...	False	False	False	False	False
2	A-3	2	2016-02-08 06:15:39	2016-02-08 12:15:39	39.102660	-84.524680	39.102090	-84.523960	0.055	At I-71/US-50/Exit 1 - Accident.	...	False	False	False	False	False
3	A-4	2	2016-02-08 06:51:45	2016-02-08 12:51:45	41.062130	-81.537840	41.062170	-81.535470	0.123	At Dart Ave/Exit 21 - Accident.	...	False	False	False	False	False
4	A-5	3	2016-02-08 07:53:43	2016-02-08 13:53:43	39.172393	-84.492792	39.170476	-84.501798	0.500	At Mitchell Ave/Exit 6 - Accident.	...	False	False	False	False	False

5 rows × 47 columns

In [73]:
```python
df.drop('ID', axis = 1, inplace = True)
df.drop('Start_Time', axis = 1, inplace = True)
df.drop('End_Time', axis = 1, inplace = True)
df.drop('End_Lat', axis = 1, inplace = True)
df.drop('End_Lng', axis = 1, inplace = True)
df.drop('Distance(mi)', axis = 1, inplace = True)
df.drop('Description', axis = 1, inplace = True)
df.drop('Number', axis = 1, inplace = True)
df.drop('Street', axis = 1, inplace = True)
df.drop('Side', axis = 1, inplace = True)
df.drop('City', axis = 1, inplace = True)
df.drop('County', axis = 1, inplace = True)
df.drop('State', axis = 1, inplace = True)
df.drop('Country', axis = 1, inplace = True)
df.drop('Timezone', axis = 1, inplace = True)
df.drop('Airport_Code', axis = 1, inplace = True)
df.drop('Weather_Timestamp', axis = 1, inplace = True)
df.drop('Wind_Direction', axis = 1, inplace = True)
df.drop('Amenity', axis = 1, inplace = True)
df.drop('Bump', axis = 1, inplace = True)
df.drop('Crossing', axis = 1, inplace = True)
df.drop('Give_Way', axis = 1, inplace = True)
df.drop('Junction', axis = 1, inplace = True)
df.drop('No_Exit', axis = 1, inplace = True)
df.drop('Railway', axis = 1, inplace = True)
df.drop('Roundabout', axis = 1, inplace = True)
df.drop('Station', axis = 1, inplace = True)
df.drop('Stop', axis = 1, inplace = True)
df.drop('Traffic_Calming', axis = 1, inplace = True)
df.drop('Traffic_Signal', axis = 1, inplace = True)
df.drop('Turning_Loop', axis = 1, inplace = True)
df.drop('Civil_Twilight', axis = 1, inplace = True)
df.drop('Nautical_Twilight', axis = 1, inplace = True)
df.drop('Astronomical_Twilight', axis = 1, inplace = True)
df.drop('Wind_Chill(F)', axis = 1, inplace = True)
df.drop('Wind_Speed(mph)', axis = 1, inplace = True)
df.drop('Precipitation(in)', axis = 1, inplace = True)
df.drop('Pressure(in)', axis = 1, inplace = True)
df.drop('Visibility(mi)', axis = 1, inplace = True)

#We are dropping variables that are either lacking in informative data (from an insurance perspective they are
# too far outside a consumer's control, or the variable themselves don't change significantly from one
# observation to the next)
```

In [74]:
```python
#Since we are keeping weather conditions here, we do want to get rid of the NaN weather condition values
# as they only account for 2% of the total data


df['Weather_Condition'].isna().sum()  # ---> 70636, which is 2% of total weather conditions; However, also looking at
#the number of weather conditions present within that variable, it's over 130, and many of them are redundant or
#insufficiently descriptive - We will remove this now and then depending on future modeling we can decide if we want
#to also model with it included
#df.shape  ---> (2845342,9)
```

Out[74]: 70636

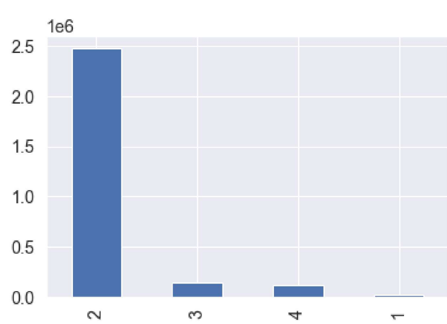
In [75]:
```python
df = df[df['Weather_Condition'].notna()]
```

In [76]:
```python
df['Weather_Condition'].isna().sum()


#Confirmed we removed the NaN values for weather condition
```

Out[76]: 0

In [77]:
```python
#Before creating any dummy variables, we will start to visualize the data that we have to look for
# trends or other outstanding observations


#Severity:

df['Severity'].value_counts().plot(kind='bar')

#the overwhelming majority of accidents are low_mid severity on a scale of low, low_mid, mid-high and high
```

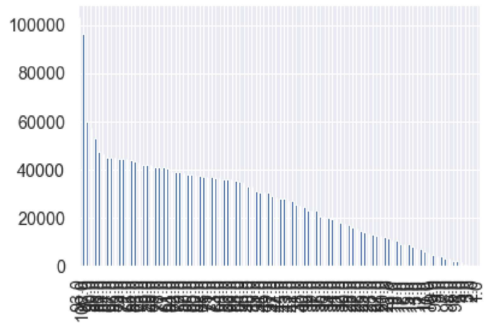
Out[77]: <AxesSubplot:>



In [78]:

```
df['Humidity(%)'].value_counts().plot(kind='bar')

#This doesn't share much with us beyond the fact that there is a typical range of humidity, but this
# might still be helpful as we move forward
```

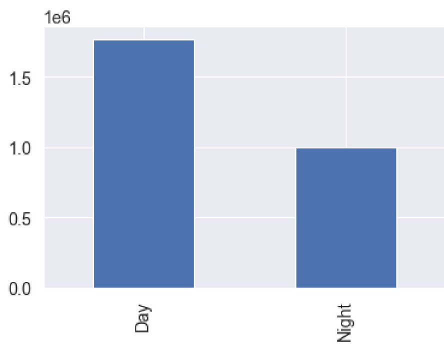
Out[78]: <AxesSubplot:>



In [79]:

```
df['Sunrise_Sunset'].value_counts().plot(kind='bar')

#More accidents happen during the day than at night, which makes sense since more driving happens
# during daylight hours - But is there any connection with severity and day vs night?
```

Out[79]: <AxesSubplot:>



In [80]:

```
df['Temperature(F)'].value_counts().plot(kind='bar')

#Much like humidity, temperature might just indicate that there's a certain window of temperatures
# that lend themselves more to people being out driving
```

Out[80]: <AxesSubplot:>

In [81]:

```
df.corr()

#This removes values for categorical variables, which are important for this project, though we do see
# a lot of variation in output below, although there aren't any strong correlations
```

Out[81]:

	Severity	Start_Lat	Start_Lng	Temperature(F)	Humidity(%)
Severity	1.000000	0.090110	0.112458	-0.045177	0.037888
Start_Lat	0.090110	1.000000	-0.157846	-0.475342	0.005714
Start_Lng	0.112458	-0.157846	1.000000	0.032446	0.171060
Temperature(F)	-0.045177	-0.475342	0.032446	1.000000	-0.366342
Humidity(%)	0.037888	0.005714	0.171060	-0.366342	1.000000

In [82]:
```python
#We can scale the data, but first we will need to set up dummy variables ...

#Encoding categorical data

df = pd.concat((df, pd.get_dummies(df.Sunrise_Sunset)),1)

df.drop(['Sunrise_Sunset'], axis=1, inplace=True)
```

In [83]:
```python
df.head()
```

Out[83]:

	Severity	Start_Lat	Start_Lng	Zipcode	Temperature(F)	Humidity(%)	Weather_Condition	Day	Night
0	3	40.108910	-83.092860	43017	42.1	58.0	Light Rain	0	1
1	2	39.865420	-84.062800	45424	36.9	91.0	Light Rain	0	1
2	2	39.102660	-84.524680	45203	36.0	97.0	Overcast	0	1
3	2	41.062130	-81.537840	44311	39.0	55.0	Overcast	0	1
4	3	39.172393	-84.492792	45217	37.0	93.0	Light Rain	1	0

In [84]:
```python
#Also do same for weather conditions...

df = pd.concat((df, pd.get_dummies(df.Weather_Condition)), 1)
df.drop(['Weather_Condition'], axis = 1, inplace = True)
```

In [85]:
```python
#For further standardization, I am removing anything beyond the main 5 digits of zipcodes

df['Zipcode'] = df['Zipcode'].str.split('-').str[0]
```

In [86]:
```python
#Have we gotten rid of all NaN values?

#26,595 NaN values out of 2774706 observations is less than 1%, so we will delete all NaN values

df.isnull().sum().sum()
```

Out[86]: 26595

In [87]:
```python
df = df.dropna()
```

In [88]:
```python
df.isnull().sum().sum()
```

Out[88]: 0

In [89]:
```python
df2 = df

#We are saving a copy of df as df2 since it is cleaned except the operations being completed next
# This will serve as a placeholder in case we have issues with scaling/absolute values.
```

In [90]:
```python
#We are now going to get absolute values on any variable that has a negative value in the df
# While scaling will fix this to an extent, it does not help when it comes time for using the
# chi-squared metric for feature selection. To reduce re-work, we will proceed as below:

df2['Start_Lat'] = df2['Start_Lat'].abs()
df2['Start_Lng'] = df2['Start_Lng'].abs()
df2['Temperature(F)'] = df2['Temperature(F)'].abs()
df2['Humidity(%)'] = df2['Humidity(%)'].abs()
```

In [91]:
```python
df2.head()
```

Out[91]:

	Severity	Start_Lat	Start_Lng	Zipcode	Temperature(F)	Humidity(%)	Day	Night	Blowing Dust	Blowing Dust / Windy	...	Thunder and Hail / Windy	Thunder in the Vicinity	Thunderstorm	Thunderstorms and Rain	Tornado	Volcanic Ash
0	3	40.108910	83.092860	43017	42.1	58.0	0	1	0	0	...	0	0	0	0	0	0
1	2	39.865420	84.062800	45424	36.9	91.0	0	1	0	0	...	0	0	0	0	0	0
2	2	39.102660	84.524680	45203	36.0	97.0	0	1	0	0	...	0	0	0	0	0	0
3	2	41.062130	81.537840	44311	39.0	55.0	0	1	0	0	...	0	0	0	0	0	0
4	3	39.172393	84.492792	45217	37.0	93.0	1	0	0	0	...	0	0	0	0	0	0

5 rows × 135 columns

In [92]:
```python
#Splitting our data into training and test sets -

X = df2.loc[ : , df2.columns != 'Severity'] # Features
y = df2.Severity # Target variable


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1) # 80% training and 20% test
```

In [93]:
```
#Test Set

xt = X_test.values
xt.shape

#feature matrix
```
Out[93]: (551905, 134)

In [94]:
```
#Train Set

xs = X_train.values
xs.shape

#feature matrix
```
Out[94]: (2207618, 134)

In [95]:
```
xr = df2.values
xr.shape

#feature matrix for full dataset of df2
```
Out[95]: (2759523, 135)

In [96]:
```
#Now we will scale the data -

scaler = StandardScaler()
```

In [97]:
```
scaler.fit(xs)
```
Out[97]: StandardScaler()

In [98]:
```
xs_scaled = scaler.transform(xs)

#Training values scaled
```

In [99]:
```
pca = PCA(n_components = 0.99, whiten = True)

#Using PCA to investigate relevant features
```

In [100]:
```
features_pca = pca.fit_transform(xs)
```

In [101]:
```
features_pca.shape

#This shows a reduction from 135 to 1 when done on our training set
#We can also check for its impact on the test and the full sets...
```
Out[101]: (2207618, 1)

In [102]:
```
#Test fit

scaler.fit(xt)
```
Out[102]: StandardScaler()

In [103]:
```
features_pca_test = pca.fit_transform(xt)
```

In [104]:
```
features_pca_test.shape

#This reduced features significantly - 135 down to 1
```
Out[104]: (551905, 1)

In [105]:
```
#For the full set

scaler.fit(xr)
```
Out[105]: StandardScaler()

In [106]:
```
xr_scaled = scaler.transform(xr)
```

In [107]:
```
features_pca_full = pca.fit_transform(xr)
```

In [108]:
```
features_pca_full.shape

#Same reduction compared to the original - down to 1 from 135
```
Out[108]: (2759523, 1)

In [109]:
```
'''

Using transformed PCA data to again get R^2 score and RMSE

'''
```
Out[109]: '\n\nUsing transformed PCA data to again get R^2 score and RMSE\n\n'

In [110]:
```
regression = linear_model.LinearRegression()

#While I am using a Decision Tree Model, as I build up to that, I am also exploring any linear relationship
# that could exist in the data - We will quickly see there is not
```

In [111]:
```
regression.fit(features_pca, y_train)

#Fitting the model
```

Out[111]:
```
LinearRegression()
```

In [112]:
```
y_c = regression.predict(features_pca)

#Using the model for predictions
```

In [113]:
```
y_cv = cross_val_predict(regression, features_pca, y_train, cv = 10)

#Cross validating
```

In [114]:
```
score_c = r2_score(y_train, y_c)
score_cv = r2_score(y_train, y_cv)

#Obtaining evaluation metrics - R^2
```

In [115]:
```
mse_c = mean_squared_error(y_train, y_c)
mse_cv = mean_squared_error(y_train, y_cv)

#Obtaining evaluation metrics - MSE
```

In [116]:
```
score_c

# 0.8 % - R^2 score
# This should indicate our model is not a good fit - A Linear Model is not our best option
```

Out[116]:
```
0.008610448812355287
```

In [117]:
```
mse_c

#MSE
```

Out[117]:
```
0.22602142266055894
```

In [118]:
```
'''
At this point, we will move forward with the Decision Tree Model
'''
```

Out[118]:
```
'\nAt this point, we will move forward with the Decision Tree Model\n'
```

In [119]:
```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

# Predict the response for test dataset
y_pred = clf.predict(X_test)
```

In [120]:
```
#Now we will report the accuracy of the model and create a confusion matrix for
# the model prediction on the test set

y_pred = clf.predict(X_test)

print("Model Accuracy with criterion gini index of {0:0.4f}". format(accuracy_score(y_test, y_pred)))

print(confusion_matrix(y_test, y_pred))

#Accuracy is 87% which seems a bit high, but perhaps due to already disproportionately high
# instances of losses of 2nd level Severity?
```

```
Model Accuracy with criterion gini index of 0.8723
[[  2222   2523    231    130]
 [  2975 457978  17815  12819]
 [   273  16841  10622   2317]
 [   137  12299   2097  10626]]
```

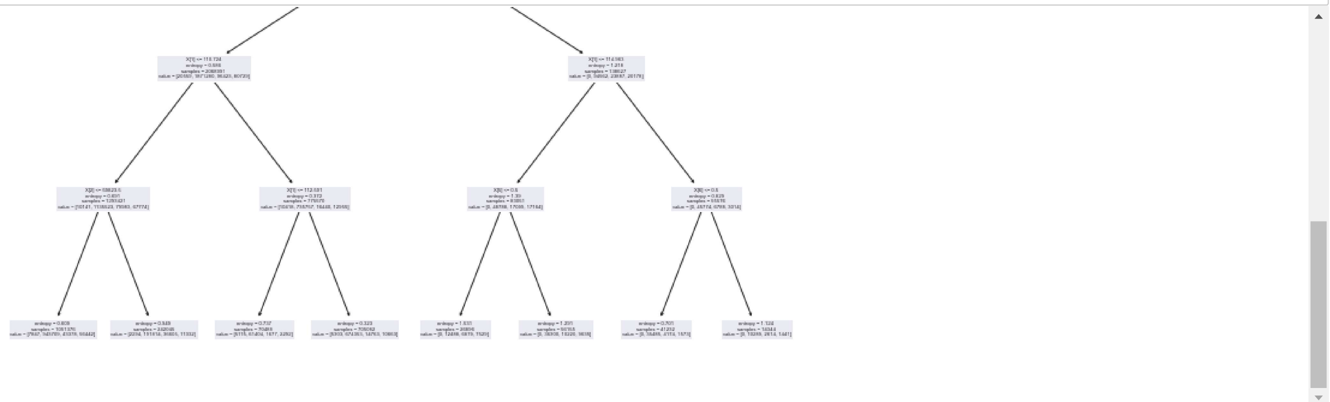
In [121]:
```
#We can also optimize this tree:

# Create Decision Tree classifer object
clf = DecisionTreeClassifier(criterion="entropy", max_depth=3)

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test data
y_pred = clf.predict(X_test)

# Model Accuracy
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

#We see here 89% which is a slight improvement over our initial iteration of the model
```

```
Accuracy: 0.8907094518078292
```

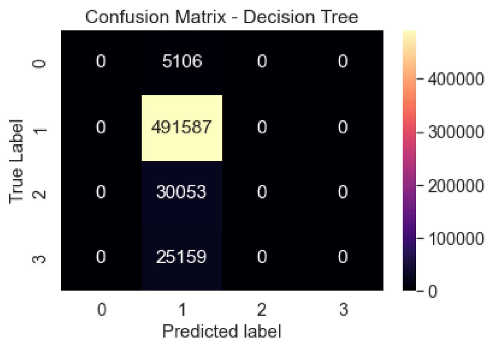
In [122]:
```
# Making Predictions with Our Model

predictions = clf.predict(X_test)
print(predictions[:5])

#5 predictions indicating the predicted severity of 5 losses
# We see all 2s, which may go back to the original concern that there are such a high proportion of
# losses rated a 2 in severity that a default assumption of 2 level severity is as good a guess as
# what the model shows
```

```
[2 2 2 2 2]
```

In [123]:
```
#We will now plot and visualize the Decision Tree


plt.figure(figsize = (12, 8))

from sklearn import tree

tree.plot_tree(clf.fit(X_train, y_train))
```



In [124]:
```
#Visualizing with Confusion Matrix

#import the relevant packages
from sklearn import metrics
import seaborn as sns
import matplotlib.pyplot as plt#get the confusion matrix
confusion_matrix = metrics.confusion_matrix(y_test,
                                            y_pred)#turn this into a dataframe
matrix_df = pd.DataFrame(confusion_matrix)#plot the result
ax = plt.axes()
sns.set(font_scale=1.3)
plt.figure(figsize=(10,7))
sns.heatmap(matrix_df, annot=True, fmt="g", ax=ax, cmap="magma")#set axis titles
ax.set_title('Confusion Matrix - Decision Tree')
ax.set_xlabel("Predicted label", fontsize =15)
ax.set_ylabel("True Label", fontsize=15)
plt.show()
```




```
<Figure size 720x504 with 0 Axes>
```

In [125]:
```
#Now we can use the Chi-Squared metric to try to select the top 5 features:

sel5 = SelectKBest(score_func = chi2, k = 5)
sel5.fit(df2.fillna(0), y)
df2.columns[sel5.get_support()].to_numpy()
```

Out[125]:
```
array(['Severity', 'Start_Lng', 'Zipcode', 'Humidity(%)', 'Clear'],
      dtype=object)
```

In [126]:
```
#Another feature selection option:

#Create a feature list from our dataframe

feature_list = list(df2.columns)

#Getting numerical feature importance:
importances = list(clf.feature_importances_)

# List of tuples with variables and importance
feature_importances = [(feature, round(importance, 2)) for feature, importance in zip(feature_list, importances)]

# Sorting the feature importance in descending order by importance
feature_importances = sorted(feature_importances, key = lambda x: x[1], reverse = True)
# Printing the feature and importances
[print('Variable: {:20} Importance: {}'.format(*pair)) for pair in feature_importances];


#Note - This is different from the previous list of top features, and this actually
# accounts for the fact that Severity itself should be removed from the calculation.
#This helps us see that latitude, snow, longitude, humidity and day vs. night do play a part
```

```
Variable: Severity             Importance: 0.0
Variable: Zipcode              Importance: 0.0
Variable: Temperature(F)       Importance: 0.0
Variable: Night                Importance: 0.0
Variable: Blowing Dust         Importance: 0.0
Variable: Blowing Dust / Windy Importance: 0.0
Variable: Blowing Sand         Importance: 0.0
Variable: Blowing Snow         Importance: 0.0
Variable: Blowing Snow / Windy Importance: 0.0
Variable: Clear                Importance: 0.0
Variable: Cloudy               Importance: 0.0
Variable: Cloudy / Windy       Importance: 0.0
Variable: Drifting Snow        Importance: 0.0
Variable: Drizzle              Importance: 0.0
Variable: Drizzle / Windy      Importance: 0.0
Variable: Drizzle and Fog      Importance: 0.0
Variable: Dust Whirls          Importance: 0.0
Variable: Duststorm            Importance: 0.0
Variable: Fair                 Importance: 0.0
Variable: Fair / Windy         Importance: 0.0
```

In [127]:
```
#Now we will use these top 5 features to do what we did above -
# We will fit a decision tree classifier on training set and then report the accuracy
# and create a confusion matrix for the model prediction on test sets

top5 = sel5.transform(df2)
top5 = pd.DataFrame(top5)
```

In [128]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(top5, y, test_size = 0.2)

clf = DecisionTreeClassifier()

clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
print("Model Accuracy with index of {0:0.4f}". format(accuracy_score(y_test, y_pred)))

from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, y_pred))


#Here we see that when we limit to 5 variables we get an index score of 1.0 which means
# that there is no relationship between the variables - that is the results are possibly
# totally random using the model as-is
```

```
Model Accuracy with index of 1.0000
[[  5160      0      0      0]
 [     0 491672      0      0]
 [     0      0  30131      0]
 [     0      0      0  24942]]
```

In [ ]:

In [ ]:

In [ ]:

In [129]:
```
'''
Another model I explored was a totally different approach that looked at the text descriptions of the
accidents to see if any helpful information could be gleaned from that data
'''
```

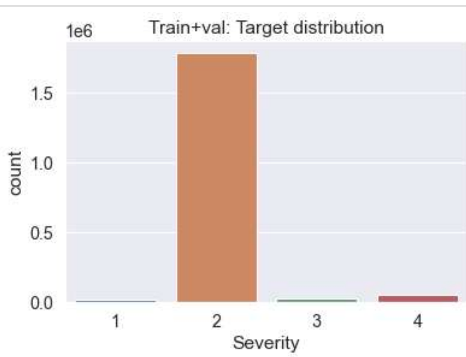
Out[129]:
```
'\nAnother model I explored was a totally different approach that looked at the text descriptions of the\naccidents to see if any helpful infor
mation could be gleaned from that data\n'
```

In [130]:
```
#Importing libraries

import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score, StratifiedKFold
from sklearn.feature_extraction.text import TfidfVectorizer
from scipy.sparse import hstack
from matplotlib import pyplot as plt
import seaborn as sns
import eli5
```

In [131]:
```
#Loading data

train = pd.read_csv('US_Accidents_Dec21_updated.csv', index_col='ID').dropna()
valid = pd.read_csv('US_Accidents_Dec21_updated.csv', index_col='ID').dropna()
test = pd.read_csv('US_Accidents_Dec21_updated.csv', index_col='ID').dropna()
```

In [132]:
```
#Concatenating training and validation sets

train_val = pd.concat([train, valid])
```

In [133]:
```
#Finding distribution of loss severity (our target) across training and validation data sets

sns.countplot(train_val['Severity']);
plt.title('Train+val: Target distribution');
```



In [134]:
```
#Setting up text transformer

text_transformer = TfidfVectorizer(stop_words='english', ngram_range=(1, 2), lowercase=True, max_features=150000)
```

In [136]:
```
#Transforming training and testing text

X_train_text = text_transformer.fit_transform(train_val['Description'])
X_test_text = text_transformer.transform(test['Description'])
```

In [137]:
```
#Data dimensions

X_train_text.shape, X_test_text.shape
```

Out[137]:
```
((1886636, 150000), (943318, 150000))
```

In [138]:
```
#Log Reg for our text data

logit = LogisticRegression(C=5e1, solver='lbfgs', multi_class='multinomial', random_state=17, n_jobs=4)
```

In [139]:
```
#Cross Validation

skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=17)
```

In [140]:
```
cv_results = cross_val_score(logit, X_train_text, train_val['Severity'], cv=skf, scoring='f1_micro')
```

In [141]:
```
cv_results, cv_results.mean()
```

Out[141]:
```
(array([0.96953579, 0.96909842, 0.96921238, 0.96954101, 0.96922298]),
 0.9693221160766188)
```

In [142]:
```
#Fitting the model

logit.fit(X_train_text, train_val['Severity'])
```

Out[142]:
```
LogisticRegression(C=50.0, multi_class='multinomial', n_jobs=4, random_state=17)
```

In [143]:

```python
#Finding top text features

eli5.show_weights(estimator=logit,
                  feature_names= list(text_transformer.get_feature_names()),
                 top=(50, 5))
```

Out[143]:

y=1 top features		**y=2** top features		**y=3** top features		**y=4** top features	
Weight?	**Feature**	**Weight?**	**Feature**	**Weight?**	**Feature**	**Weight?**	**Feature**
+8.733	road accident	+39.874	traffic	+8.010	accident	+42.264	closed
+8.662	rd accident	+29.886	stationary traffic	+6.329	lane closed	+20.873	closed accident
+7.662	ave accident	+29.886	stationary	+5.776	il	+15.983	closed road
+6.674	60 grand	+26.114	slow traffic	+5.028	belt	+11.668	road closed
+6.038	accident lanes	+26.066	slow	+4.159	fullerton ave	+10.790	blocked
+5.559	st accident	+23.379	incident	+4.002	accident lanes	+8.117	rd
+5.389	earlier accident	+21.533	near	+3.885	rd accident	+6.554	expect delays
+5.334	83rd ave	+20.670	caution	+3.810	dr accident	+6.085	road
+5.183	drexel rd	+19.237	drive caution	+3.773	cedar lake	+5.590	expect
+4.970	earlier	+17.695	drive	+3.735	traffic problem	+5.520	closed cr
+4.880	az	+14.965	crash	+3.688	hwy accident	+5.249	closure
+4.817	magee rd	+10.724	shoulder closed	+3.650	county farm	+5.135	delays
+4.781	dr accident	+10.430	fl	+3.643	pulaski	+5.058	mp expect
+4.714	magee	+10.303	veh	+3.611	lane traffic	+4.668	lanes blocked
+4.654	kolb rd	+10.061	traffic fl	+3.551	army	+4.591	lane blocked
+4.645	kolb	+9.825	alternate route	+3.489	accident traffic	+4.536	crash investigation
+4.627	thornydale	+9.136	conndot	+3.449	gary ave	+4.474	milemarker
+4.627	thornydale rd	+8.920	closed alternate	+3.417	overturned	+4.438	fl
+4.390	rd earlier	+8.618	right shoulder	+3.373	overturned vehicle	+4.395	summit rd
+4.350	tn	+8.437	shoulder	+3.370	20 lake	+4.141	county
+4.293	sw	+8.256	use	+3.349	sherman dr	+4.010	investigation
+4.288	tangerine rd	+7.918	1039	+3.347	kedzie	+3.992	ny
+4.285	craycroft rd	+7.818	accident	+3.315	pulaski rd	+3.647	summit
+4.275	craycroft	+7.796	alternate	+3.282	lane	+3.632	blocked overturned
+4.264	drexel	+7.630	road	+3.261	accident lane	+3.593	95 accident
+4.259	blvd accident	+7.472	lanes closed	+3.233	army trail	+3.520	motorists
+4.201	59th ave	+7.446	right	+3.223	lanes blocked	+3.514	st
+4.161	tatum blvd	+7.167	lanes	+3.217	alternate lane	+3.416	twp lanes
+4.160	mcdowell rd	+7.132	rd drive	+3.213	single alternate	+3.249	nb near
+4.097	tangerine	+6.943	ave	+3.121	bartlett rd	+3.172	126 closed
+4.094	prince rd	+6.904	route	+3.104	accident left	+3.158	directions
+4.070	ne	+6.778	lane closed	+3.052	ryan	+3.127	rt
+3.918	grant rd	+6.546	rd	+3.041	blocked right	+3.076	construction ca
+3.877	pike accident	+6.391	mn	+3.038	st accident	+3.043	route
+3.861	speedway blvd	+6.357	directions road	+3.033	single	+3.042	closed fl
+3.847	35th ave	+6.173	closed incident	+3.014	mcculloch blvd	+3.024	sb near
+3.838	cave creek	+6.074	use alternate	+2.961	ave exit	+3.018	nj
+3.813	dunlap ave	+5.968	lane lanes	+2.957	problem	+2.962	126
+3.800	accident	+5.903	chp	+2.941	blocked ahead	+2.959	95
+3.737	91st ave	+5.893	rd near	+2.928	purcell blvd	+2.915	motorists expect
+3.719	litchfield rd	+5.749	ca	+2.925	butterfield rd	+2.881	traffic affected
+3.711	swan rd	+5.651	accident road	+2.890	riverwoods rd	+2.801	road 126
+3.708	tatum	+5.571	spun	+2.886	ahead	+2.796	eb near
+3.687	valencia rd	+5.534	vehicle spun	+2.878	midlothian rd	+2.765	dr
+3.684	mcclintock dr	+5.273	st drive	+2.859	exit	+2.751	22 closed
+3.683	pima	+5.262	vs	+2.828	lombard accident	+2.748	st directions
+3.656	harrison rd	+5.231	lane	+2.824	tollway	+2.746	wb near
+3.634	litchfield	+5.191	restriction	+2.805	accident blocked	+2.705	rd road
+3.633	mcclintock	+5.160	near house	+2.786	roosevelt rd	+2.645	mn
+3.621	rural rd	+4.998	tc	+2.776	gary	+2.637	md
… 6742 more positive …		*… 119927 more positive …*		*… 11223 more positive …*		*… 25905 more positive …*	
… 143204 more negative …		*… 30019 more negative …*		*… 138723 more negative …*		*… 124041 more negative …*	
-9.903	near	-6.290	accident right	-10.837	slow	-8.239	conndot
-10.768	stationary traffic	-6.622	blocked	-11.449	closed	-8.492	rd accident
-10.768	stationary	-12.608	closed road	-12.502	stationary	-8.999	shoulder closed
-10.774	rd	-17.022	closed accident	-12.502	stationary traffic	-15.620	lane closed
-16.535	traffic	-24.230	closed	-15.728	traffic	-19.628	accident

In [144]:

```python
test_preds = logit.predict(X_test_text)
```

In [145]:

```python
pd.DataFrame(test_preds, columns=['Severity']).head()
```

Out[145]:

	Severity
0	4
1	4
2	4
3	2
4	2

In [ ]: