Data Science capstone report

Introduction/Business Problem:

Vehicular accidents are common on roads across the world. The type of accident varies in severity. They can simply range from property damage, such as minor fender-benders, to loss of life with one or more parties. It is an unfortunate commonplace. What if it was possible to predict the severity of an accident occurring given current conditions? Drivers across the world would benefit from this information. Decisions could be made about whether it was worth the risk of getting on the road or postponing the trip for a later time when conditions improve

Data

A dataset from the Seattle Department of Transportation (SDOT) will be used to create and train multiple models, which will be evaluated for a comparison of each model's accuracy. The SDOT data set includes entries for nearly 195,000 accidents from 2004 to the present. The severity of each accident is categorized with multiple features to choose from for modeling. A few examples of the features are as follows: • Collision • Address Type (Alley, Block, Intersection)

- Location
- Collision Type
- Number of people involved in the collision
- Number of pedestrians involved in the collision 1
- Number of cyclists involved in the collision
- Number of vehicles involved in the collision
- Number of fatalities
- Weather conditions
- Road conditions
- Lighting Conditions

The primary focus of this investigation are the environmental driving conditions at the time of the collision. Therefore, the following features will be investigated:

- Weather conditions (WEATHER)
- Roadconditions (ROADCOND)
- Light conditions (LIGHTCOND)

In [24]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn import preprocessing
from sklearn.metrics import jaccard_score
from sklearn.metrics import f1_score
```

In [2]:
```
import pandas as pd
df = pd.read_csv('D:\Data-Collisions.csv', usecols = ['WEATHER', 'ROADCOND', 'LIGHTCOND', 'SEVERITYCODE'])
df.head()
```

Out[2]:

	SEVERITYCODE	WEATHER	ROADCOND	LIGHTCOND
0	2	Overcast	Wet	Daylight
1	1	Raining	Wet	Dark - Street Lights On
2	1	Overcast	Dry	Daylight
3	1	Clear	Dry	Daylight
4	2	Raining	Wet	Daylight

In [3]:
```
df = df[['WEATHER', 'ROADCOND', 'LIGHTCOND', 'SEVERITYCODE']]
df.head()
```

Out[3]:

	WEATHER	ROADCOND	LIGHTCOND	SEVERITYCODE
0	Overcast	Wet	Daylight	2
1	Raining	Wet	Dark - Street Lights On	1
2	Overcast	Dry	Daylight	1
3	Clear	Dry	Daylight	1
4	Raining	Wet	Daylight	2

In [4]:
```
df.count()
```

Out[4]:
```
WEATHER         189592
ROADCOND        189661
LIGHTCOND       189503
SEVERITYCODE    194673
dtype: int64
```

In [5]:
```
new_df = df.dropna(axis = 0)
new_df.count()
```

Out[5]:
```
WEATHER         189337
ROADCOND        189337
```

jupyter capstone Last Checkpoint: 22 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

Out[6]:
```
Clear                       111008
Raining                      33117
Overcast                     27681
Unknown                      15039
Snowing                        901
Other                          824
Fog/Smog/Smoke                 569
Sleet/Hail/Freezing Rain       113
Blowing Sand/Dirt               55
Severe Crosswind                25
Partly Cloudy                    5
Name: WEATHER, dtype: int64
```

In [7]:
```
new_df['ROADCOND'].value_counts(sort=True)
```

Out[7]:
```
Dry               124300
Wet                47417
Unknown            15031
Ice                 1206
Snow/Slush           999
Other                131
Standing Water       115
Sand/Mud/Dirt         74
Oil                   64
Name: ROADCOND, dtype: int64
```

In [8]:
```
new_df['LIGHTCOND'].value_counts(sort=True)
```

Out[8]:
```
Daylight                    116077
Dark - Street Lights On      48440
Unknown                      13456
Dusk                          5889
Dawn                          2502
Dark - No Street Lights       1535
Dark - Street Lights Off      1192
Other                          235
Dark - Unknown Lighting         11
Name: LIGHTCOND, dtype: int64
```

In [9]:
```
new_df['SEVERITYCODE'].value_counts(sort=True)
```

Out[9]:
```
1    132285
2     57052
Name: SEVERITYCODE, dtype: int64
```

jupyter capstone Last Checkpoint: 23 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

```
Dusk            5648
Dawn            2413
Name: LIGHTCOND, dtype: int64
```

In [14]:
```
weather_numeric = [*range(0, 9, 1)]
roadcond_numeric = [*range(0, 7, 1)]
lightcond_numeric = [*range(0, 4, 1)]

weather_list = ['Clear', 'Raining', 'Overcast', 'Snowing', 'Fog/Smog/Smoke', 'Sleet/Hail/Freezing Rain', 'Blowing Sand/Dirt', 'Se
roadcond_list = ['Dry', 'Wet', 'Ice', 'Snow/Slush', 'Standing Water', 'Sand/Mud/Dirt', 'Oil']
lightcond_list = ['Daylight', 'Dark', 'Dusk', 'Dawn']

# create final dataframe and change the string values to integers
final_df = new_df.copy()
final_df['WEATHER'].replace(to_replace = weather_list, value = weather_numeric, inplace = True)
final_df['ROADCOND'].replace(to_replace = roadcond_list, value = roadcond_numeric, inplace = True)
final_df['LIGHTCOND'].replace(to_replace = lightcond_list, value = lightcond_numeric, inplace = True)

final_df.head()
```

Out[14]:

	WEATHER	ROADCOND	LIGHTCOND	SEVERITYCODE
0	2	1	0	2
1	1	1	1	1
2	2	0	0	1
3	0	0	0	1
4	1	1	0	2

In [15]:
```
X = final_df[['WEATHER', 'ROADCOND', 'LIGHTCOND']]
X[0:5]
```

Out[15]:

	WEATHER	ROADCOND	LIGHTCOND
0	2	1	0
1	1	1	1
2	2	0	0
3	0	0	0
4	1	1	0

In [16]:
```
y = final_df['SEVERITYCODE'].values
y[0:5]
```

Out[16]: array([2, 1, 1, 1, 2], dtype=int64)

jupyter capstone Last Checkpoint: 22 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

In [10]:
```
index_values_to_drop = new_df[((new_df['WEATHER'] == 'Unknown') | (new_df['WEATHER'] == 'Other'))
                              | ((new_df['ROADCOND'] == 'Unknown') | (new_df['ROADCOND'] == 'Other'))
                              | ((new_df['LIGHTCOND'] == 'Unknown') | (new_df['LIGHTCOND'] == 'Other'))].index
new_df = new_df.drop(index_values_to_drop) #used this notation to prevent warnings from setting inplace = True
new_df.count()
```

Out[10]:
```
WEATHER         169957
ROADCOND        169957
LIGHTCOND       169957
SEVERITYCODE    169957
dtype: int64
```

In [11]:
```
new_df['WEATHER'].value_counts(sort=True)
```

Out[11]:
```
Clear                       108825
Raining                      32648
Overcast                     26923
Snowing                        825
Fog/Smog/Smoke                 553
Sleet/Hail/Freezing Rain       107
Blowing Sand/Dirt               46
Severe Crosswind                25
Partly Cloudy                    5
Name: WEATHER, dtype: int64
```

In [12]:
```
new_df['ROADCOND'].value_counts(sort=True)
```

Out[12]:
```
Dry               121490
Wet                46324
Ice                 1080
Snow/Slush           833
Standing Water       105
Sand/Mud/Dirt         65
Oil                   60
Name: ROADCOND, dtype: int64
```

In [13]:
```
new_df['LIGHTCOND'].replace(['Dark - Street Lights On','Dark - No Street Lights', 'Dark - Street Lights Off', 'Dark - Unknown Lig
new_df['LIGHTCOND'].value_counts(sort=True)
```

Out[13]:
```
Daylight    112618
Dark         49278
Dusk          5648
Dawn          2413
Name: LIGHTCOND, dtype: int64
```

In [14]:
```
weather_numeric = [*range(0, 9, 1)]
roadcond_numeric = [*range(0, 7, 1)]
lightcond_numeric = [*range(0, 4, 1)]
```

jupyter capstone Last Checkpoint: 23 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run Code

In [16]:
```
y = final_df['SEVERITYCODE'].values
y[0:5]
```

Out[16]:
```
array([2, 1, 1, 1, 2], dtype=int64)
```

In [17]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 4)
X_train = preprocessing.StandardScaler().fit(X_train).transform(X_train)
X_train[0:5]
```

Out[17]:
```
array([[-0.66539872, -0.57845596, -0.63726064],
       [-0.66539872, -0.57845596, -0.63726064],
       [-0.66539872, -0.57845596, -0.63726064],
       [-0.66539872, -0.57845596,  2.55426532],
       [-0.66539872, -0.57845596,  0.95850234]])
```

In [18]:
```
X_test = preprocessing.StandardScaler().fit(X_test).transform(X_test)
X_test[0:5]
```

Out[18]:
```
array([[-0.67312293, -0.58065715,  0.96901319],
       [-0.67312293, -0.58065715, -0.63963578],
       [ 1.77506825, -0.58065715,  0.96901319],
       [-0.67312293,  1.26882728, -0.63963578],
       [-0.67312293, -0.58065715, -0.63963578]])
```

In [22]:
```
#decision tree
dt = DecisionTreeClassifier(criterion = 'entropy', max_depth = 4)
dt.fit(X_train, y_train)
y_pred = dt.predict(X_test)
print(jaccard_score(y_test, y_pred))
print(f1_score(y_test, y_pred))
```

```
0.6733054836432102
0.8047609838428946
```

In [23]:
```
lr = LogisticRegression(C = 0.01)
lr.fit(X_train, y_train)
y_pred1 = lr.predict(X_test)
print(jaccard_score(y_test, y_pred1))
print(f1_score(y_test, y_pred1))
```

```
0.6733054836432102
0.8047609838428946
```

In [ ]: