This notebook will be mainly used for the capstone project

Table of contents

In [1]:

```
import itertools
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import NullFormatter
import pandas as pd
import numpy as np
import matplotlib.ticker as ticker
from sklearn import preprocessing
%matplotlib inline
```

In [2]:

```
print ('Hello Capstone Project Course')
```

```
Hello Capstone Project Course
```

Conclusion

In conclusion, based on the work done below, the data set cannot tell us the the possibility of getting into a car accident, but can tell us that if we are in an accident in certain conditions how severe it would be.

The decision tree model works 100% of the time on the test data after being trained.

Introduction

Discussion of the background:

To complete capstone, I will be working on a case study which is to predict the severity of an accident.

Say you are driving to another city for work or to visit some friends. It is rainy and windy, and on the way, you come across a terrible traffic jam on the other side of the highway. Long lines of cars barely moving. As you keep driving, police car start appearing from afar shutting down the highway. Oh, it is an accident and there's a helicopter transporting the ones involved in the crash to the nearest hospital. They must be in critical condition for all of this to be happening.

Problem:

Now, wouldn't it be great if there is something in place that could warn you, given the weather and the road conditions about the possibility of you getting into a car accident and how severe it would be, so that you would drive more carefully or even change your travel if you are able to. Well, this is exactly what you will be working on in this course.

Data

Description of Data

We will be using our shared data for Seattle city as an example of how to deal with the accidents data.

The first column is the labeled data. The remaining columns have different types of attributes. Some or all can be used to train the model.

The label for the data set is severity, which describes the fatality of an accident. The shared data has unbalanced labels. The data will be balanced, otherwise, you will create a biased ML model. We can do some feature engineering to improve the predictability of the model.

Attributes:

In total there are 37 attributes (columns). Some attribtes have misisng data, and includes both numerical and categorical types of data. Attributes inlcude Location, Weather condition, Car speeding, Light conditions, Road condition, Junction junction, number of people involved, number of vehicles involed.

Severity key 1 = Property Damage Only Collision (136485 collisions) Severity key 2 = Injury Collision (58188 collisions)

Methodology

The problem states that the road conditions and weather should be categories that indicate whether there is likelihood of getting into a car accident.

I have decided to use a **decision tree** because, looking at the categories, the weather and road conditions will effectively point to a damage only collision, or an injury collision. If a decision tree does not produce a high enough accuracy, I will try another method.

I will **clean the data** removing any blank or Unknown values from the Weather and Road Conditions columns.

In [60]:

```
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages/IPython/co
re/interactiveshell.py:3072: DtypeWarning: Columns (33) have mixed types.
Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [4]:

Out[4]:

	SEVERITYCODE	X	Y	OBJECTID	INCKEY	COLDETKEY	REPORTNO
0	2	-122.323148	47.703140	1	1307	1307	3502005
1	1	-122.347294	47.647172	2	52200	52200	2607959
2	1	-122.334540	47.607871	3	26700	26700	1482393
3	1	-122.334803	47.604803	4	1144	1144	3503937
4	2	-122.306426	47.545739	5	17700	17700	1807429

5 rows × 38 columns

Data Visualization and Analysis

Let’s see how many of each Severity is in our data set

In [5]:

Out[5]:

```
0         2
1         1
2         1
3         1
4         2
         ..
194668    2
194669    1
194670    2
194671    2
194672    1
Name: SEVERITYCODE, Length: 194673, dtype: int64
```

In [6]:

Out[6]:

```
Index(['SEVERITYCODE', 'X', 'Y', 'OBJECTID', 'INCKEY', 'COLDETKEY', 'REPO
RTNO',
       'STATUS', 'ADDRTYPE', 'INTKEY', 'LOCATION', 'EXCEPTRSNCODE',
       'EXCEPTRSNDESC', 'SEVERITYCODE.1', 'SEVERITYDESC', 'COLLISIONTYP
E',
       'PERSONCOUNT', 'PEDCOUNT', 'PEDCYLCOUNT', 'VEHCOUNT', 'INCDATE',
       'INCDTTM', 'JUNCTIONTYPE', 'SDOT_COLCODE', 'SDOT_COLDESC',
       'INATTENTIONIND', 'UNDERINFL', 'WEATHER', 'ROADCOND', 'LIGHTCOND',
       'PEDROWNOTGRNT', 'SDOTCOLNUM', 'SPEEDING', 'ST_COLCODE', 'ST_COLDE
SC',
       'SEGLANEKEY', 'CROSSWALKKEY', 'HITPARKEDCAR'],
      dtype='object')
```

In [8]:

Out[8]:

```
Clear                         111135
Raining                        33145
Overcast                       27714
Unknown                        15091
Snowing                          907
Other                            832
Fog/Smog/Smoke                   569
Sleet/Hail/Freezing Rain         113
Blowing Sand/Dirt                 56
Severe Crosswind                  25
Partly Cloudy                      5
Name: WEATHER, dtype: int64
```

In [22]:

Out[22]:

	SEVERITYCODE	WEATHER	ROADCOND
0	2	Overcast	Wet
1	1	Raining	Wet
2	1	Overcast	Dry
3	1	Clear	Dry
4	2	Raining	Wet
...	...	...	...
194668	2	Clear	Dry
194669	1	Raining	Wet
194670	2	Clear	Dry
194671	2	Clear	Dry
194672	1	Clear	Wet

194673 rows × 3 columns

I want to see where we have null values, then remove them

Below we can see that we have null values in WEATHER and READCOND columns

In [23]:

```
SEVERITYCODE
False    194673
Name: SEVERITYCODE, dtype: int64

WEATHER
False    189592
True       5081
Name: WEATHER, dtype: int64

ROADCOND
False    189661
True       5012
Name: ROADCOND, dtype: int64
```

In [25]:

```
SEVERITYCODE
False    173006
Name: SEVERITYCODE, dtype: int64

WEATHER
False    173006
Name: WEATHER, dtype: int64

ROADCOND
False    173006
Name: ROADCOND, dtype: int64
```

We now have data without any null values and any 'Unkown' values

In [26]:

Out[26]:

```
SEVERITYCODE     int64
WEATHER         object
ROADCOND        object
dtype: object
```

In [27]:

Out[27]:

	SEVERITYCODE	WEATHER	ROADCOND
0	2	Overcast	Wet
1	1	Raining	Wet
2	1	Overcast	Dry
3	1	Clear	Dry
4	2	Raining	Wet
...	...	...	...
106	1	Clear	Dry
109	1	Clear	Dry
110	1	Clear	Dry
111	1	Overcast	Dry
112	1	Overcast	Dry

100 rows × 3 columns

In [44]:

Out[44]:

```
array([[2, 'Overcast', 'Wet'],
       [1, 'Raining', 'Wet'],
       [1, 'Overcast', 'Dry'],
       [1, 'Clear', 'Dry'],
       [2, 'Raining', 'Wet']], dtype=object)
```

Some features in this dataset are categorical such as **WEATHER** or **ROADCOUNT**. Sklearn Decision Trees do not handle categorical variables. But still we can convert these features to numerical values.

In [46]:

Out[46]:

```
array([[2, 4, 7],
       [1, 6, 7],
       [1, 4, 0],
       [1, 1, 0],
       [2, 6, 7]], dtype=object)
```

In [47]:

Out[47]:

```
0    2
1    1
2    1
3    1
4    2
Name: SEVERITYCODE, dtype: int64
```

Setting up the Decision Tree

We will be using **train/test split** on our **decision tree**. Let's import **train_test_split** from **sklearn.cross_validation**.

In [49]:

In [50]:

Practice

Print the shape of X_trainset and y_trainset. Ensure that the dimensions match

In [51]:

```
Train set: (121104, 3) (121104,)
```

In [52]:

```
Test set: (51902, 3) (51902,)
```

Modeling

We will first create an instance of the **DecisionTreeClassifier** called **drugTree**.
Inside of the classifier, specify *criterion="entropy"* so we can see the information gain of each node.

In [53]:

Out[53]:

```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=
4,
            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=Non
e,
            splitter='best')
```

In [54]:

Out[54]:

```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=
4,
            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=Non
e,
            splitter='best')
```

Prediction

Let's make some **predictions** on the testing dataset and store it into a variable called **predaccTree**.

In [55]:

In [56]:

```
[1 1 2 1 1]
122146    1
144594    1
11905     2
105489    1
51948     1
Name: SEVERITYCODE, dtype: int64
```

Results

When we test the model, we see that it accurately predicts the possibility of an accident 100% of the time

```
In [57]:
```

```
DecisionTrees's Accuracy:  1.0
```

Discussion

The problem with the data is that it only tells us of those journeys where there was an accident. It does not help tell us whether a journey will definitely result in an accident or not.

```
In [ ]:
```