Capstone project

Introduction

In this project we will predict the severity of getting into an accident according to a traffic dataset. The objective of this project is to predict if there is a possibility of getting into a car accident and how sever it would be, so a driver would be more carefully or change the travel.

Problem

The goal is to predict the possibility of an accident based on the weather, road condition, vehicle count and number of persons. Which impacts on peoples' life and a potential of a fatality accident.

1. How much financial cost can be reduced by avoiding accidents?
2. What is the rate of prevented accidents using a predictive model?

Data

Data acquisition

A dataset of all collisions provided by SPD and recorded by traffic records will be used. This is an extensive data set from the Seattle Police Department, with over 190000 observations collected over the last 15 years. To accurately build a model to prevent future accidents and/or reduce their severity, we will use the following attributes — ADDRTYPE, WEATHER, ROADCOND, VEHCOUNT, PERSONCOUNT.

1. ADDRTYPE: collision address type
2. WEATHER: A description of the weather conditions during the time of the collision.
3. ROADCOND: The condition of the road during the collision.
4. VEHCOUNT: The number of vehicles involved in the collision
5. PERSONCOUNT:The number of persons involved in the collision

Exploratory data analysis

I use waffle chart to explore how does the weather affects collisions the weather consisted in the following scenarios:

```
WEATHER
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
```

For this I used the matplotlib library, then compute the proportion of each category with respect to the total

```
Clear: 0.5808520312965489
Raining: 0.17323381992463321
Overcast: 0.14484845634005988
Unknown: 0.07887378417506834
Snowing: 0.004740475929148962
Other: 0.004348485086055056
Snowing: 0.004740475929148962
Other: 0.004348485086055056
Fog/Smog/Smoke: 0.002973903862939095
Sleet/Hail/Freezing Rain: 0.0005905995369281507
Blowing Sand/Dirt: 0.00029268649617678267
Severe Crosswind: 0.0001306636143646351
Partly Cloudy: 2.613272287292702e-05
```

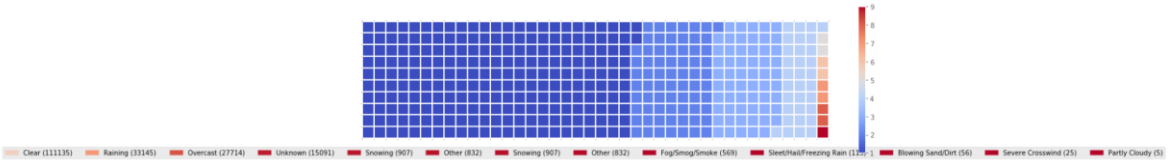
From this, I created the waffle chart and compute the number of weathe rs for each category

```
Clear: 232
Raining: 69
Overcast: 58
Unknown: 32
Snowing: 2
Other: 2
Snowing: 2
Other: 2
Fog/Smog/Smoke: 1
Sleet/Hail/Freezing Rain: 0
Blowing Sand/Dirt: 0
Severe Crosswind: 0
Partly Cloudy: 0
```

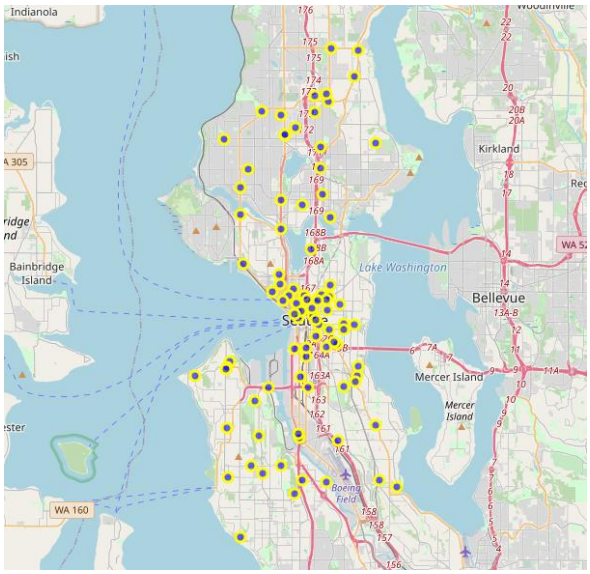
I initialize the waffle chart as an empty matrix and load the data int
o it:

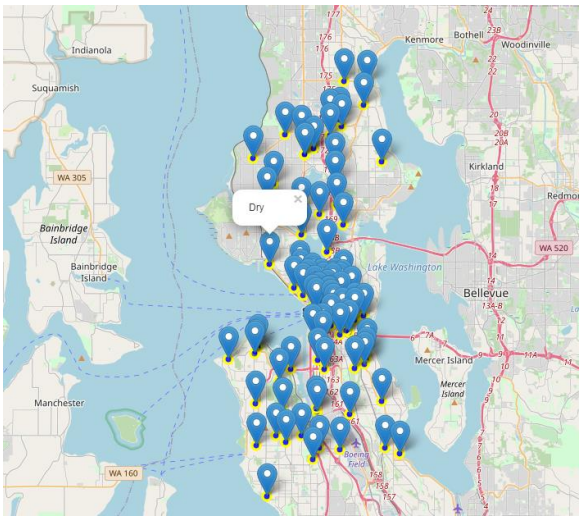
```
# initialize the waffle chart as an empty matrix
waffle_chart = np.zeros((height, width))

# define indices to loop through waffle chart
category_index = 0
tile_index = 0

# populate the waffle chart
for col in range(width):
    for row in range(height):
        tile_index += 1

        # if the number of tiles populated for the current category is equal to its corresponding all
        if tile_index > sum(tiles_per_category[0:category_index]):
            # ...proceed to the next category
            category_index += 1

        # set the class value to an integer, which increases with class
        waffle_chart[row, col] = category_index

print ('Waffle chart populated!')
```

I instantiate a new figure object, use mathshow to display the waffle chart, get the axis, set minor ticks, added gridlines based on minor ticks and created legends.



Then I use seabron and folium to create locations markers of the accidents in seattle, for this I reduce to 200 the dataset to use a sample





Then to classify the rate of the severity accident [1,2] I used a tree classifier to predict if the variables affects to the severity of the accident

```
from sklearn import preprocessing

le_WEATHER = preprocessing.LabelEncoder()
le_WEATHER.fit(['Clear','Raining','Overcast','Unknown','Snowing','Other','Fog/Smog/Smoke',
                'Sleet/Hail/Freezing Rain', 'Blowing Sand/Dirt', 'Severe Crosswind','Partly Cloudy'])
X[:,0] = le_WEATHER.transform(X[:,0])

le_ROADCOND = preprocessing.LabelEncoder()
le_ROADCOND.fit(['Dry','Wet','Unknown','Ice','Snow/Slush','Other','Standing Water','Sand/Mut/Dirt', 'Oil'])
X[:,1] = le_ROADCOND.transform(X[:,1])

le_ADDRTYPE = preprocessing.LabelEncoder()
le_ADDRTYPE.fit(['Block','Intersection','Alley'])
X[:,2] = le_ADDRTYPE.transform(X[:,2])

X[0:10]
```

```
X_trainset, X_testset, y_trainset, y_testset = train_test_split(X, y, test_size=0.2, random_state=2)
```

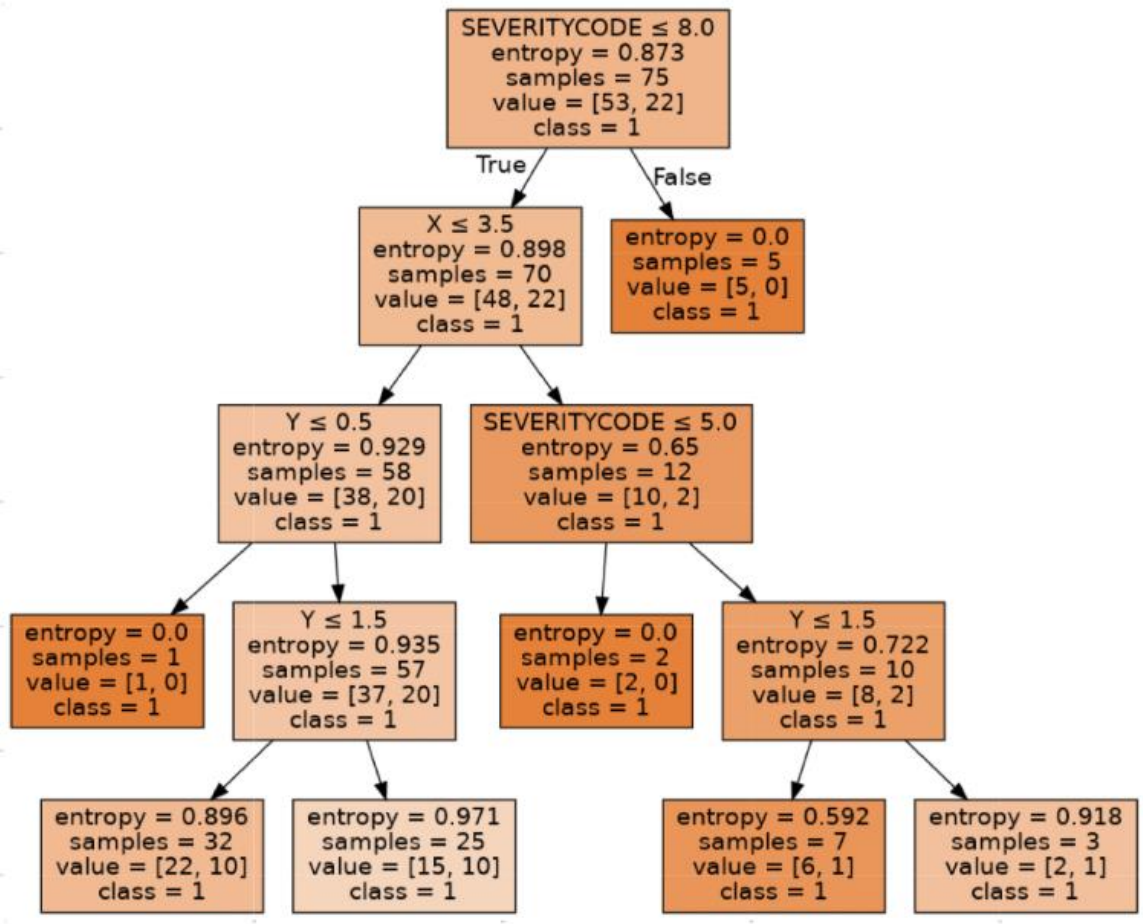
```
Tree = DecisionTreeClassifier(criterion="entropy", max_depth = 4)
Tree
```

```
]: 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=None,
                splitter='best')
```

Overall accuracy on prediction:

```
from sklearn import metrics
import matplotlib.pyplot as plt
print("DecisionTrees's Accuracy: ", metrics.accuracy_score(y_testset, predTree))
```

```
    DecisionTrees's Accuracy:  0.7368421052631579
```


SEVERITYCODE ≤ 8.0
entropy = 0.873
samples = 75
value = [53, 22]
class = 1
True
False
X ≤ 3.5
entropy = 0.898
samples = 70
value = [48, 22]
class = 1
entropy = 0.0
samples = 5
value = [5, 0]
class = 1
Y ≤ 0.5
entropy = 0.929
samples = 58
value = [38, 20]
class = 1
SEVERITYCODE ≤ 5.0
entropy = 0.65
samples = 12
value = [10, 2]
class = 1
entropy = 0.0
samples = 1
value = [1, 0]
class = 1
Y ≤ 1.5
entropy = 0.935
samples = 57
value = [37, 20]
class = 1
entropy = 0.0
samples = 2
value = [2, 0]
class = 1
Y ≤ 1.5
entropy = 0.722
samples = 10
value = [8, 2]
class = 1
entropy = 0.896
samples = 32
value = [22, 10]
class = 1
entropy = 0.971
samples = 25
value = [15, 10]
class = 1
entropy = 0.592
samples = 7
value = [6, 1]
class = 1
entropy = 0.918
samples = 3
value = [2, 1]
class = 1