Analysis of Seattle Collision Data

Miranda Childs

August 24th, 2020

1. Introduction and Business Understanding

1.1 Background

In Seattle, WA, from 2004 to present there have been 194,673 collisions reported by the Seattle Police Department (SPD) to the Seattle Department of Transportation (SDOT). 58,188 of those collisions resulted in an injury.

1.2 Problem

Through analysis of the collision data, I will discover which factors contribute most heavily to severe collisions. By determining and then focusing on those features, I will be able to create meaningful strategies to reduce the number of accidents, especially those with injuries, in order to increase the wellbeing and longevity of our community.

1.3 Interest

I will prepare a presentation for SDOT and the Vision Zero Network, "a collaborative campaign helping communities reach their goals of Vision Zero -- eliminating all traffic fatalities and severe injuries -- while increasing safe, healthy, equitable mobility for all." (Vision Zero Network). Through thorough analysis I will make recommendations for the next campaigns and strategies that Vision Zero can execute in collaboration with SDOT.

2. Data Understanding and Preparation

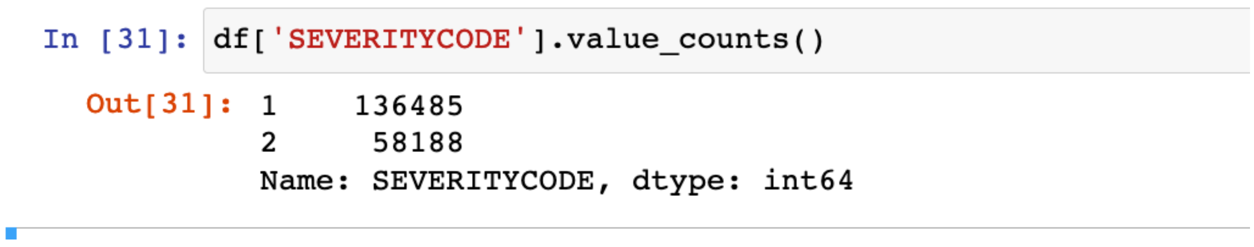
2.1 Acquisition of Data

I will be using the shared data set on collisions from 2004 to present, provided by the Traffic Records Group in conjunction with the Seattle Police Department and Seattle Department of Transportation. (Here are links to the data set and corresponding metadata.)

2.2 Data Preparation

I will be preparing the data using the following methods:

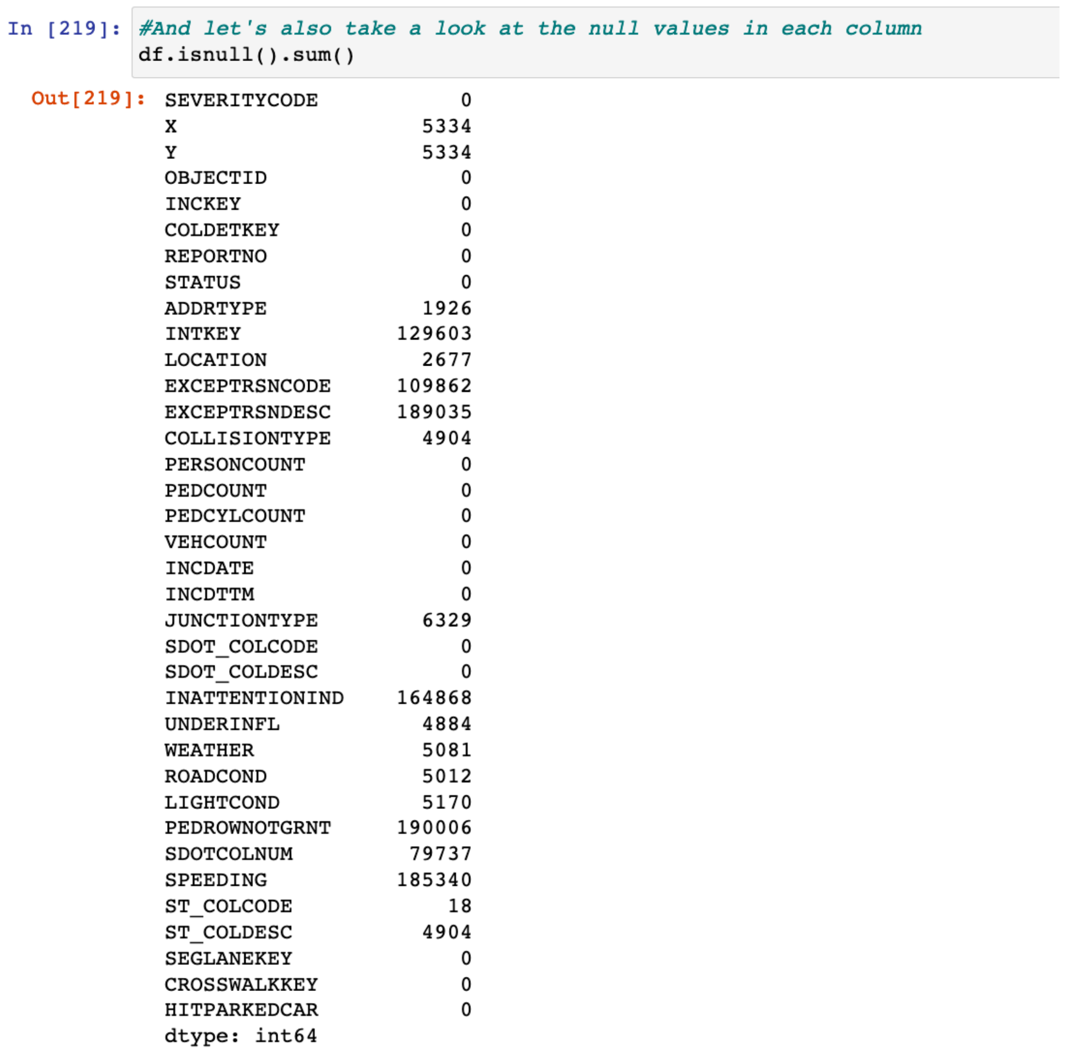
Balancing the labeled data: as we can see the labeled data is imbalanced, with 194,673 type 1



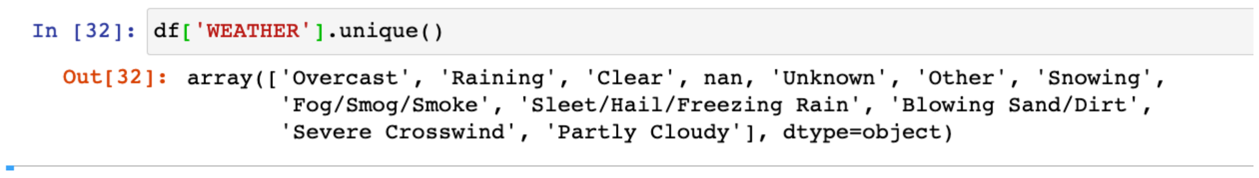
```
In [31]: df['SEVERITYCODE'].value_counts()

Out[31]: 1    136485
         2     58188
         Name: SEVERITYCODE, dtype: int64
```

entries (“property damage only”), and 58,188 type 2 entries (“injury”). We must balance the data so that we can use machine learning algorithms most effectively. As we have a fairly large dataset, we will achieve this by undersampling, i.e. removing type 1 entries.

Removing and replacing missing data: exploring the data, we have many null values which must be remedied.



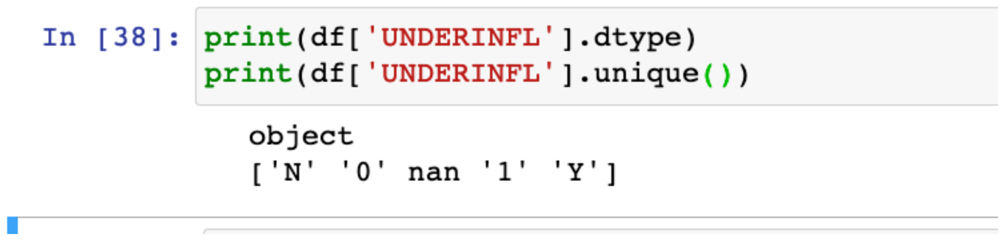
```
In [219]: #And let's also take a look at the null values in each column
          df.isnull().sum()

Out[219]: SEVERITYCODE          0
          X                  5334
          Y                  5334
          OBJECTID              0
          INCKEY                0
          COLDETKEY             0
          REPORTNO              0
          STATUS                0
          ADDRTYPE           1926
          INTKEY           129603
          LOCATION           2677
          EXCEPTRSNCODE    109862
          EXCEPTRSNDESC    189035
          COLLISIONTYPE      4904
          PERSONCOUNT           0
          PEDCOUNT              0
          PEDCYLCOUNT           0
          VEHCOUNT              0
          INCDATE               0
          INCDTTM               0
          JUNCTIONTYPE       6329
          SDOT_COLCODE          0
          SDOT_COLDESC          0
          INATTENTIONIND   164868
          UNDERINFL          4884
          WEATHER            5081
          ROADCOND           5012
          LIGHTCOND          5170
          PEDROWNOTGRNT    190006
          SDOTCOLNUM        79737
          SPEEDING         185340
          ST_COLCODE           18
          ST_COLDESC         4904
          SEGLANEKEY            0
          CROSSWALKKEY          0
          HITPARKEDCAR          0
          dtype: int64
```



```
In [32]: df['WEATHER'].unique()

Out[32]: array(['Overcast', 'Raining', 'Clear', nan, 'Unknown', 'Other', 'Snowing',
                'Fog/Smog/Smoke', 'Sleet/Hail/Freezing Rain', 'Blowing Sand/Dirt',
                'Severe Crosswind', 'Partly Cloudy'], dtype=object)
```

Furthermore, attributes such as weather have both NaN values as well as the values ‘Unknown’ and ‘Other’.

Transformation: for example, the attribute UNDERINFL is type object but can easily be transformed to integers, converting N to 0 and Y to 1.

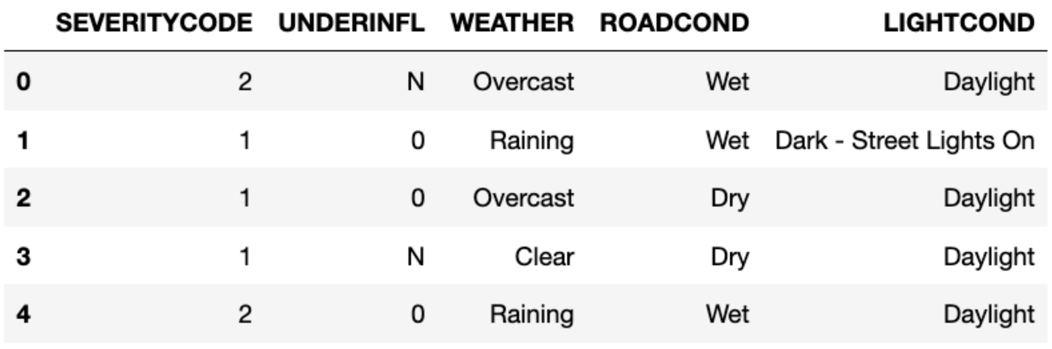


```
In [38]: print(df['UNDERINFL'].dtype)
         print(df['UNDERINFL'].unique())

            object
            ['N' '0' nan '1' 'Y']
```

Cleaning the dataset: we will drop unnecessary columns, especially those with mostly null values, and rename others for ease of reference.

3. Methodology

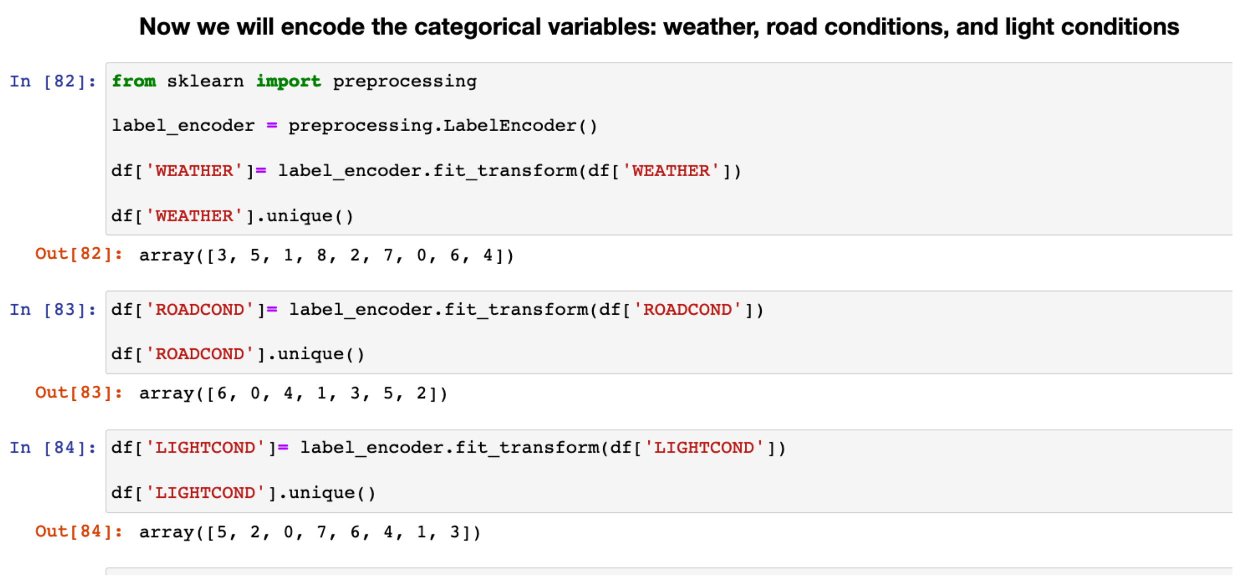
During the preparation phase, I encountered countless obstacles, but was able to have some success with my methods. Based on the number of null values in many of the feature columns, it was clear which features would be used to predict our target variable of severity (SEVERITYCODE).



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

It was then necessary to transform the values regarding whether a driver had been under the influence (UNDERINFL) so that they would all be numerical instead of a messy combination of Ys, Ns, 0s, and 1s.

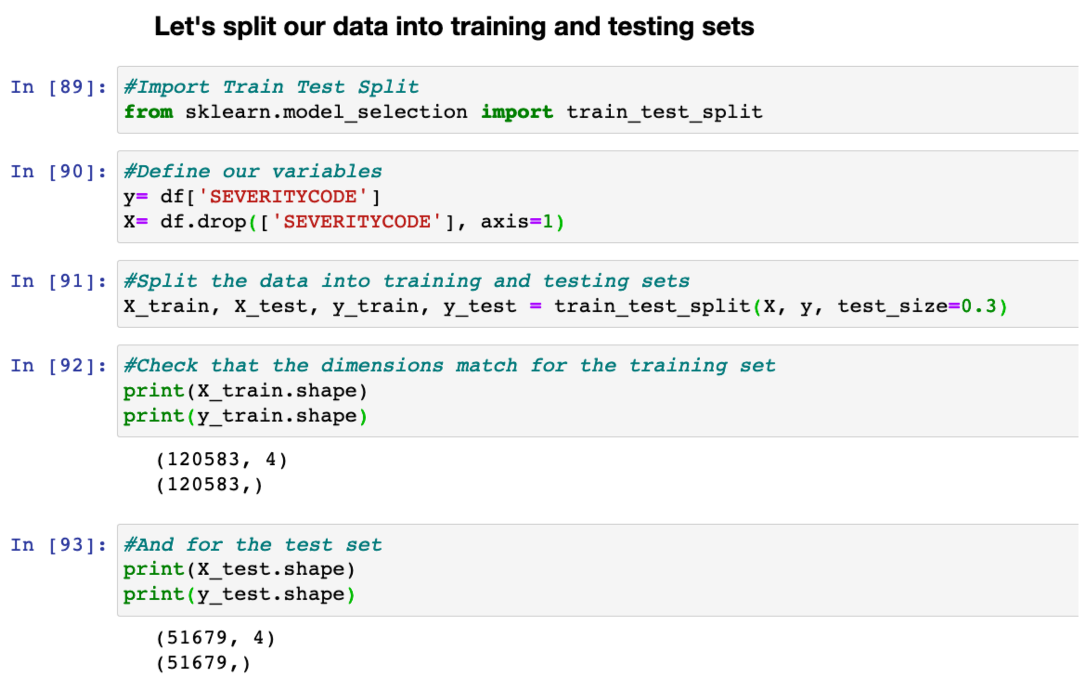
Next, the task was to encode the categorical variables so that they could work with machine learning. I initially tried One-Hot Encoding, but it created an error in the next step (creating a decision tree) and I was unable to understand the root of the error. I circled back to the preparation stage at this point and employed a different encoding technique, with which I successfully transformed the variables into numerical arrays.



Now we will encode the categorical variables: weather, road conditions, and light conditions

```
In [82]: from sklearn import preprocessing

         label_encoder = preprocessing.LabelEncoder()

         df['WEATHER']= label_encoder.fit_transform(df['WEATHER'])

         df['WEATHER'].unique()

  Out[82]: array([3, 5, 1, 8, 2, 7, 0, 6, 4])

In [83]: df['ROADCOND']= label_encoder.fit_transform(df['ROADCOND'])

         df['ROADCOND'].unique()

  Out[83]: array([6, 0, 4, 1, 3, 5, 2])

In [84]: df['LIGHTCOND']= label_encoder.fit_transform(df['LIGHTCOND'])

         df['LIGHTCOND'].unique()

  Out[84]: array([5, 2, 0, 7, 6, 4, 1, 3])
```

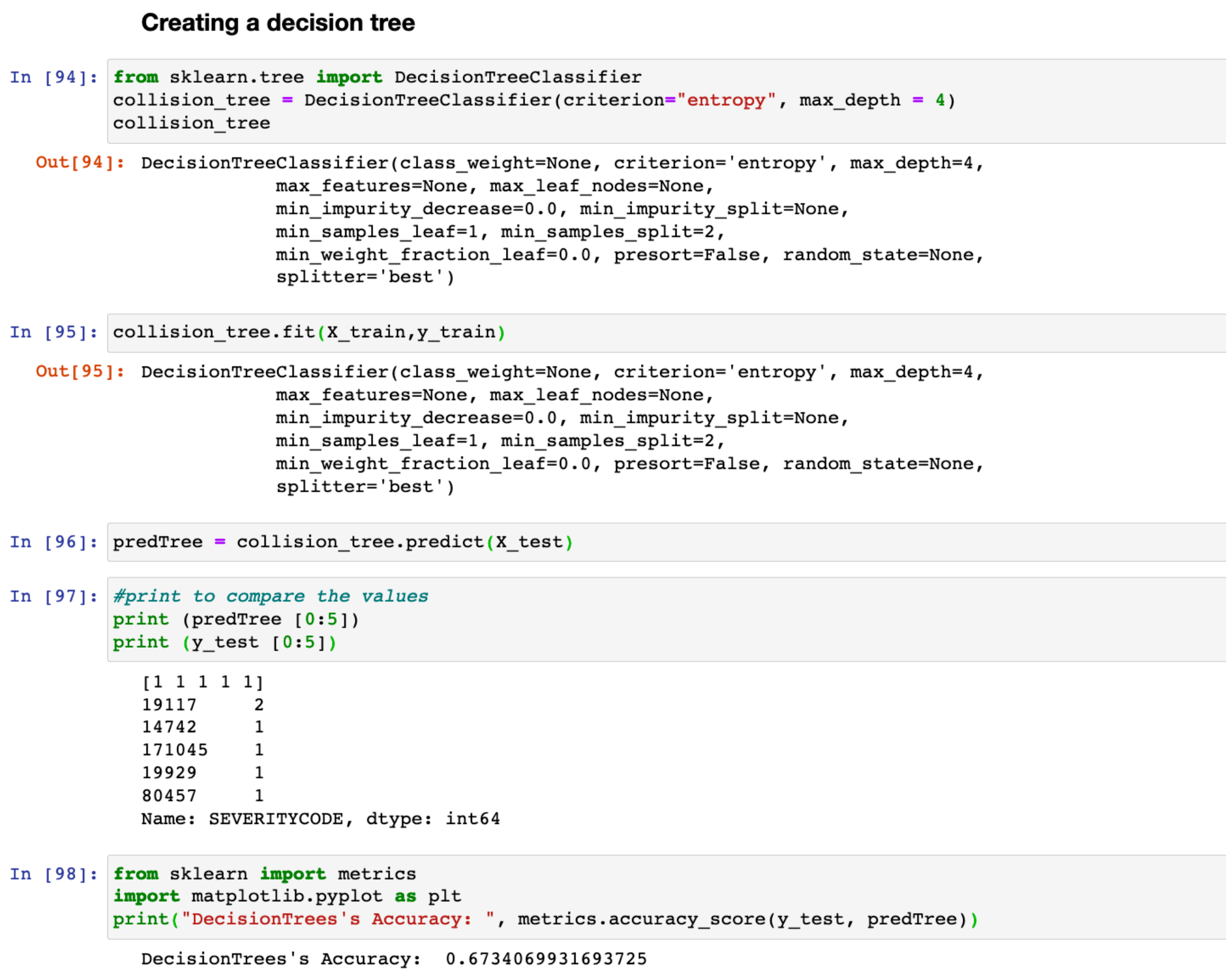
I was then able to split the data into training and testing sets.



Let's split our data into training and testing sets

```
In [89]: #Import Train Test Split
         from sklearn.model_selection import train_test_split

In [90]: #Define our variables
         y= df['SEVERITYCODE']
         X= df.drop(['SEVERITYCODE'], axis=1)

In [91]: #Split the data into training and testing sets
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

In [92]: #Check that the dimensions match for the training set
         print(X_train.shape)
         print(y_train.shape)

           (120583, 4)
           (120583,)

In [93]: #And for the test set
         print(X_test.shape)
         print(y_test.shape)

           (51679, 4)
           (51679,)
```

My plan was to use classification techniques, since the goal was to develop a model to predict the **category** of the collision: either severe (resulting in injury), or less severe (property damage only). A decision tree was a good algorithm to try as it is a classification algorithm and because it can work well with imbalanced data.



Creating a decision tree

```
In [94]: from sklearn.tree import DecisionTreeClassifier
         collision_tree = DecisionTreeClassifier(criterion="entropy", max_depth = 4)
         collision_tree

Out[94]: 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')

In [95]: collision_tree.fit(X_train,y_train)

Out[95]: 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')

In [96]: predTree = collision_tree.predict(X_test)

In [97]: #print to compare the values
         print (predTree [0:5])
         print (y_test [0:5])

           [1 1 1 1 1]
           19117     2
           14742     1
           171045    1
           19929     1
           80457     1
           Name: SEVERITYCODE, dtype: int64

In [98]: from sklearn import metrics
         import matplotlib.pyplot as plt
         print("DecisionTrees's Accuracy: ", metrics.accuracy_score(y_test, predTree))

           DecisionTrees's Accuracy:  0.6734069931693725
```

After creating the decision tree, the goal was to check the accuracy and then visualize it using Matplotlib.

Unfortunately, every step of this process was near impossible for me, despite the countless hours I put into it. I revisited the machine learning course as well as other courses from the certification in attempt to learn the things I apparently had not mastered well enough to complete this project.

I knew that the preparation stage would take a long time, and indeed it did, but after about 36 hours of attempts to properly prepare my data for machine learning, I was stumped again when attempting to use the machine learning algorithms that had proved so challenging just a week or two beforehand.

In addition to trying my hand at old class labs I read numerous data science blogs, hoping for some help. These efforts all proved to yield some momentary success. I also posted several questions in the discussion forums, hoping for some help and clarification. Although I received some very helpful feedback and even links to more data science blog posts from my fellow students, funnily, even after 72 hours I did not receive a single response to a question I posed from an instructor or teaching assistant. Although those responses have been historically obtuse, I was still hoping from some guidance from a professor.

Despite having spent nearly twice the projected time so far, I would be happy to put in more work if I had new information or feedback that could yield better results.

4. Results

I was able to finally create a decision tree, with an accuracy score of 67%. However, my numerous attempts at visualizing the decision tree were not successful. Despite having just completed the machine learning course, none of this was fresh in my memory. In fact, I struggled quite a bit with the machine learning course as well, which I'm sure was part of the problem in achieving results here.

If I had had better luck with my decision tree, and especially the visualization thereof, I would have continued on to other classification methods, in the hopes of finding something with a higher accuracy score. I also intended to balance the data, but as that was not covered in the certificate program, I theorized that perhaps we weren't supposed to do so.

I am wondering if another issue was my choice to use the shared dataset. I would love any feedback here. Again, my best case scenario would be that I am able to more fully understand the problems I am having in my notebook, and be able to change and complete what I had originally intended.

5. Discussion

If you're available and open to talking about the topics presented in these courses, and/or would be interested in forming a study group please email me at mirandacchilds@gmail.com. I would also love any feedback about the work I attempted to do in my notebook.

I would still like to learn these subjects despite having put so much time into this inadequate program, all for naught. I will definitely be looking for other programs, people to study with, and hopefully a mentor or two. I would like to say I would be interested in a tutor, but that would not be fiscally attainable for me at this time.

6. Conclusion

I hope that my failure here is somehow helpful for someone (perhaps even for me!). Alternately, if anyone has constructive feedback for me, tips, or other resources, I would be really happy to be able to re-visit this code, correct my errors and create a true finish to the course. It doesn't make sense for me to put in more hours at this point, as I am not creating better results. I am truly just spinning my wheels.

This program has been an interesting journey for me. Initially data science seemed like a perfect fit for me. I think it was the SQL course where I first started seriously doubting whether I was capable of learning these new skills. However, I have since learned that many people loathe SQL, and that gave me hope. Despite struggling quite a bit, I have still managed to pull it together and do well on the final project for most of these courses. I suppose I was expecting that would be the case for this course as well. I do not think it is impossible for me to learn, but perhaps it is at this moment with the resources I currently have.

And despite this experience, I don't think that online learning is a bad idea. I like the independence and flexibility of it. I found that other students were very helpful and that if that community had been fostered, it could have been an even more a valuable element to the program. However, I think that instructor feedback and office hours are even more essential than I would have assumed. Getting an unsatisfactory response from a teaching assistant, 24-36 hours after posing a question, could not be less sufficient. The next program I look for will be one with present and accessible instructors.

I hope to continue with my studies and look back on this experience as a challenging yet laughable marker of how inexperienced I once was— and if you agree, I wish that for you as well!