DS 3001
Homework 2

Carlos Barcelos
April 4, 2018

Task 1

Data Cleaning

The first task was to collect the tables of Circle, Triangle, and Fireball shaped UFOs and combine them into one file. This was accomplished using a simple HTML to CSV converter online and combining the three CSV files into one in Microsoft excel. The next step was to remove missing files using Pandas's dropna() function. This reduced the dataset from 31,385 to 28,472 rows, an acceptable 9% decrease in the dataset. In dropping rows with missing values, I also dropped any datetime that was incorrectly formatted. This involved using the Pandas to_datetime() function with the errors attribute to convert improper datetimes to null values; this operation eliminated 25 dirty rows from the dataset. Finally, it was necessary to restrict our dataset to consider only sightings made between January 1, 2005 and September 22, 2016. This reduced the dataset greatly but increased the reliability of the data by using only the most recent sightings.

The next consideration was converting the **Duration** column into a more usable form. This required removing all instance of text only entries (e.x. 'Frequently' is not a usable duration). Then I was able to collect the upper bound of the duration and convert the time to seconds. This was accomplished using regular expressions and a function which would convert whatever time the duration was recorded in (ex. 'minutes', 'hours', 'years', etc.) into seconds. This process removed 3,332 rows of data.

In summary, the initial dataset of 31,385 rows was reduced to 16,706 clean rows of usable data. This is a 46.7% reduction in data that is necessary to get a better, numerical analysis of UFO sightings. The majority of the cleaning came from limiting the range of sightings.

Data Analysis

The first step to creating a boxplot for each of the UFO sightings was to remove outliers. This was accomplished by finding the interquartile range of each UFO shape and removing values according to the following formula:

$$Outlier = Q1 - (1.5 * IQR)$$

$$Outlier = Q3 + (1.5 * IQR)$$

With no outliers, the boxplot in Figure 1 could be created. This shows the five-figure analysis of the duration of sightings according to UFO shape.

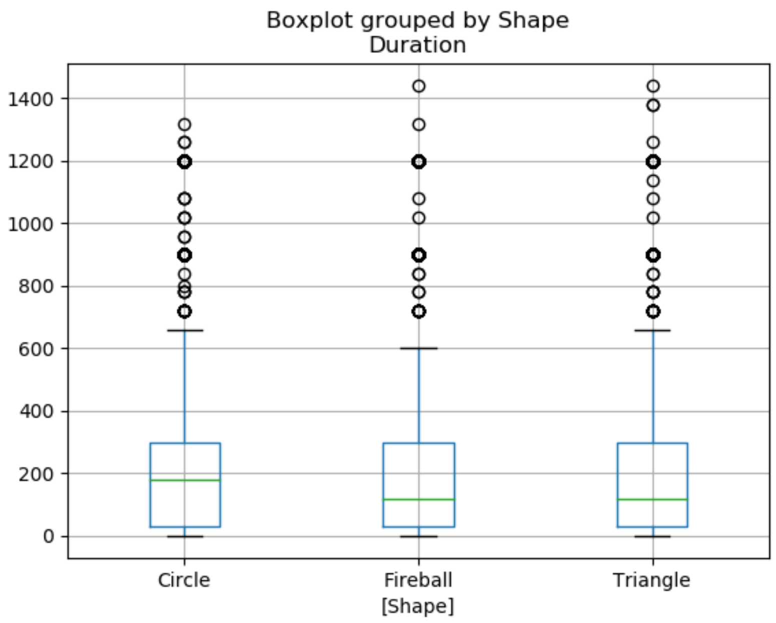


Figure 1 Duration of UFO sightings by shape of UFO

In order to create a time series figure I first created a unique data frame with the **shape** and **year** of the sighting. I was then able to create a pivot table indexed over the year of the sighting, sorted by the shape and aggregated by the count. This was then turned into an easy to digest time series figure as seen in Figure 2.

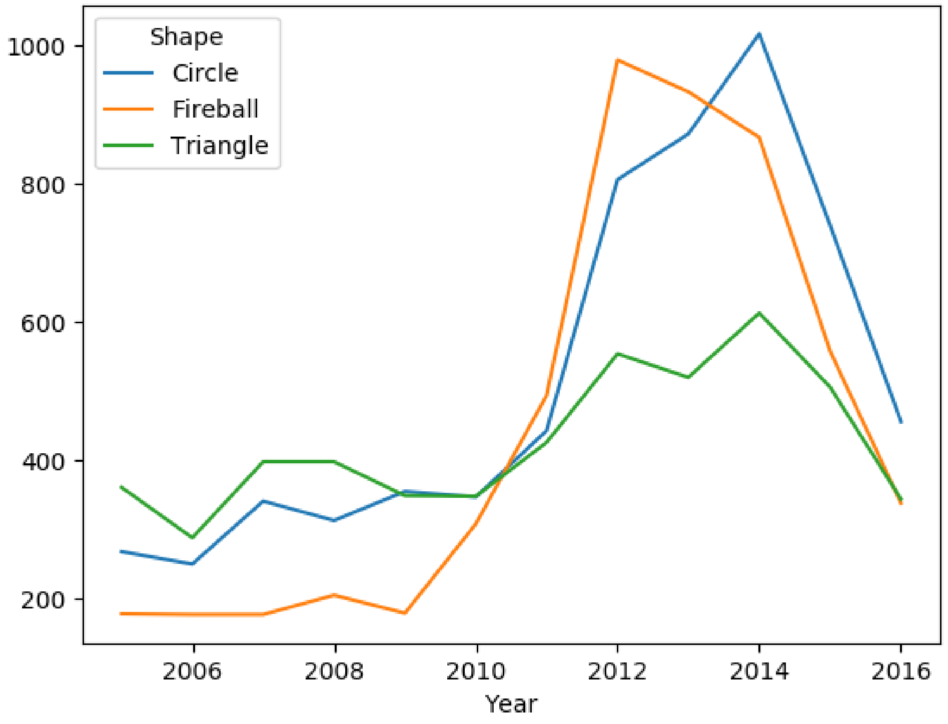


Figure 2 Time series figure of number of sightings per year

A bar chart listing sightings by state is another useful visualization of the data. Figure 3 displays the number of UFO sightings by state.

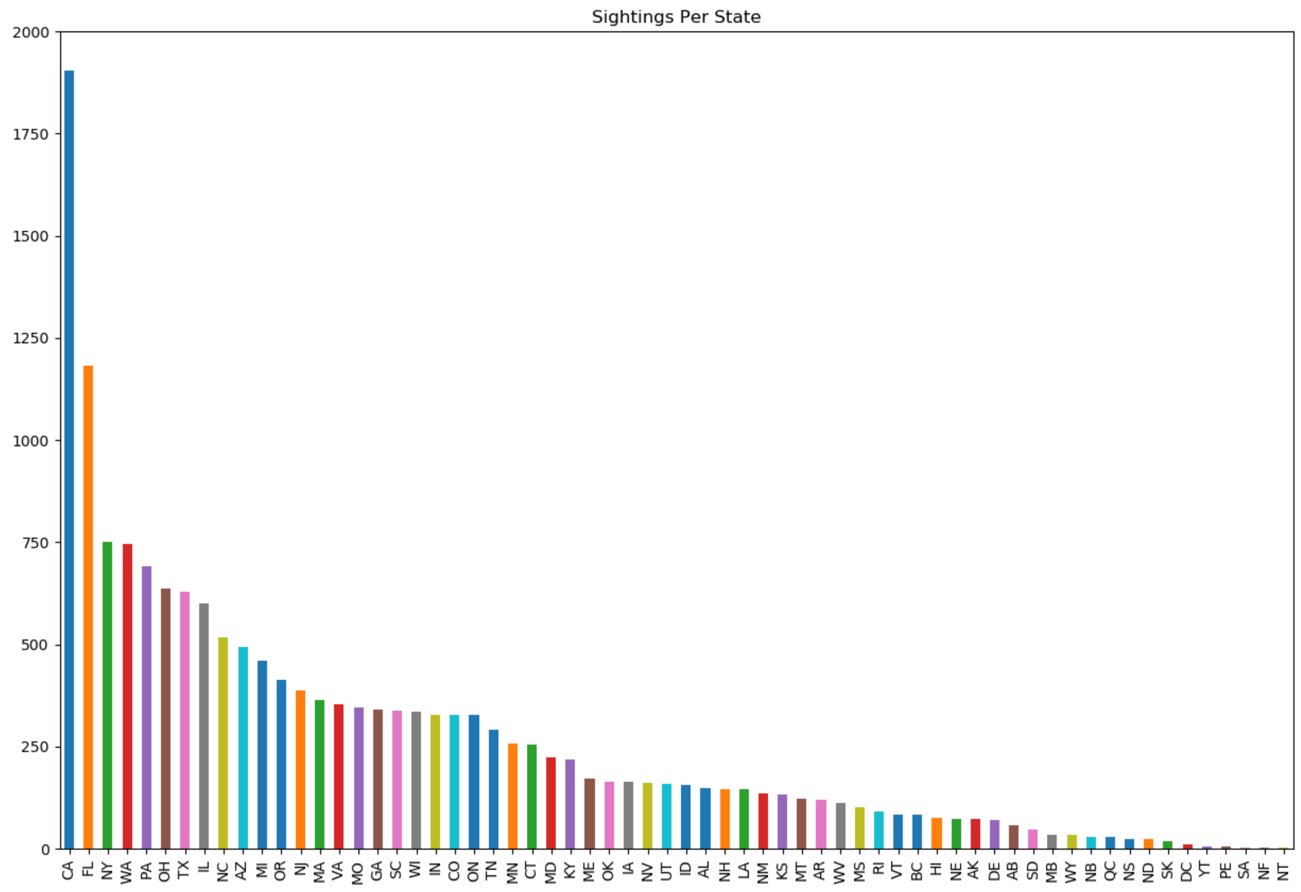


Figure 3 UFO sightings by state

To normalize the sighting by state population the first step was the extract the information of sightings by state. With this CSV file I was able to remove entries that were not states in the United States of America (there was several misprints on state names, 13 to be exact). I then found the populations of each of the states. Finally, I followed the formula below to find the number of UFO sightings per state per 100,000 citizens.

$$Normalized = \frac{count}{population} * 100{,}00$$

The result of this normalization is seen in Figure 4. It is interesting to note that by normalizing the sightings by state population, the states with the most number of sightings are no longer on top. In fact, the states that had the most number of sightings in Figure 3 seem to be those states with the largest population. This leads me to believe that the number of sightings is not related to any actual UFO sighting but the number of people living in the state. The more people there were, the more people claimed to see a UFO.

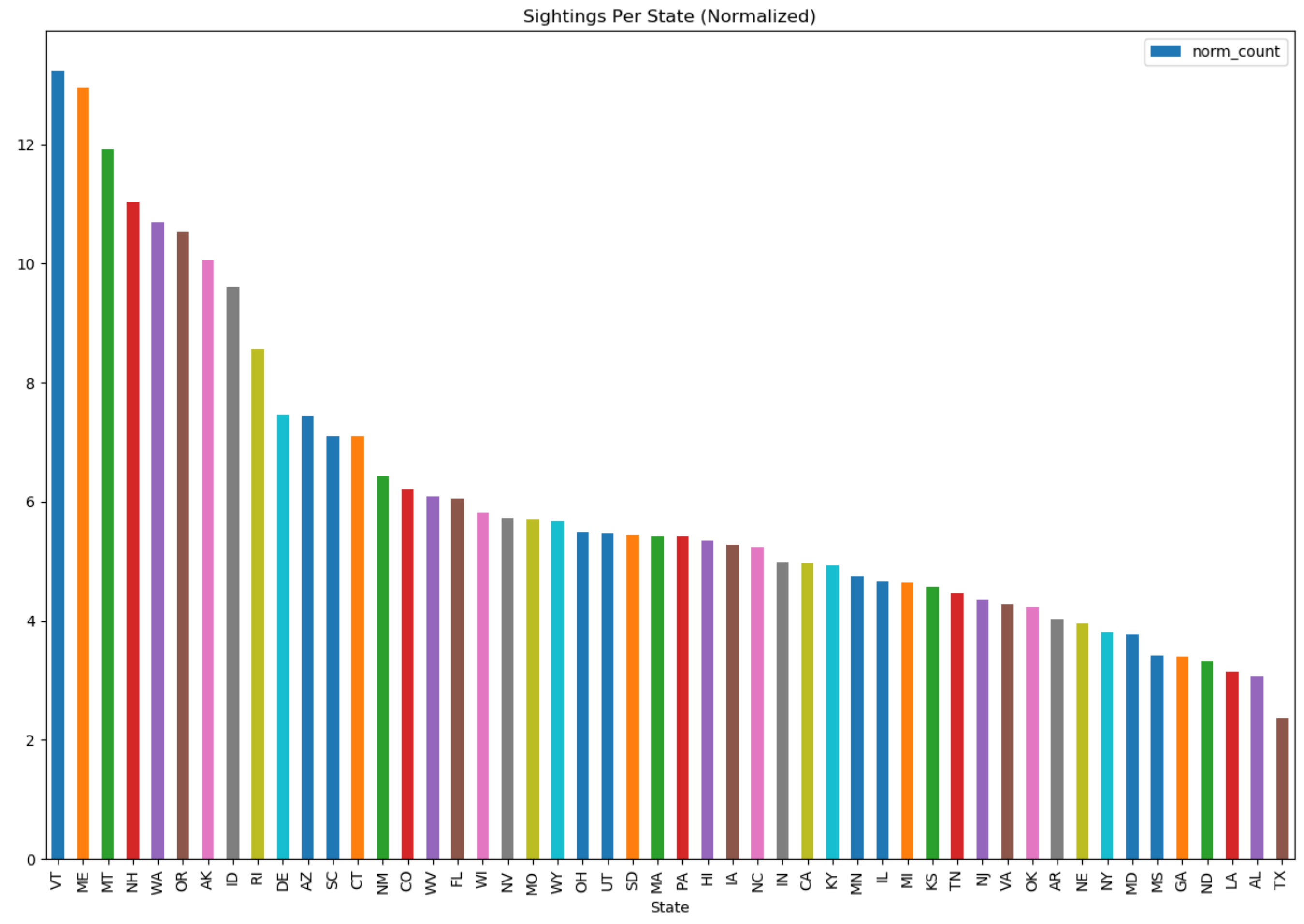


Figure 4 UFO sightings by state normalized by population

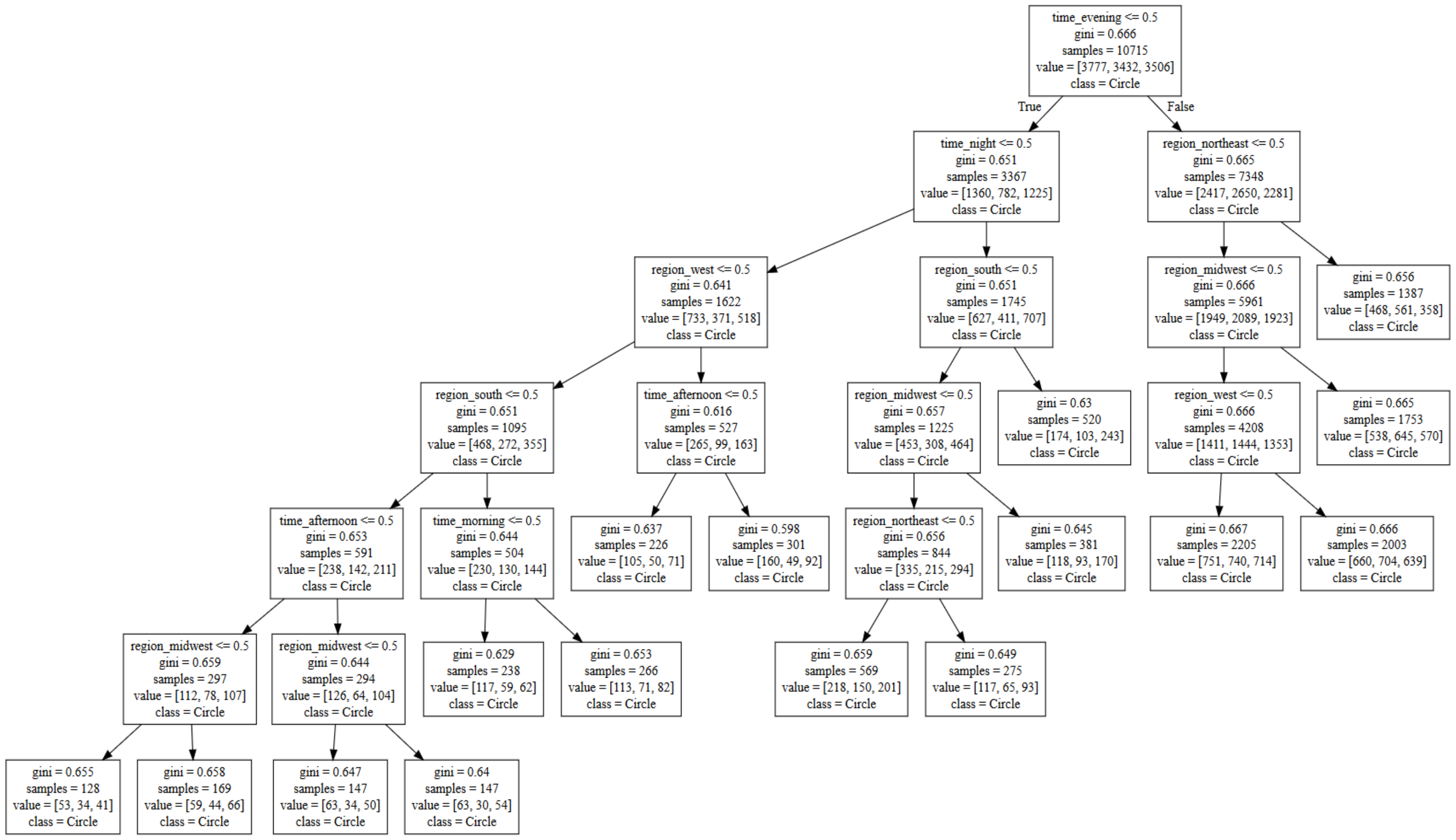
Task 2

The first step to predicting UFO shapes was to create categories in which to organize the data. The two categories were **country region** and **time of day** of the sightings. Creating each category involved binning the various ‘State’ and ‘Time’ columns into four pieces. This was accomplished within Pandas using lambda expressions and a few levels of if statements.

Then the new data frame was to separate the cleaned UFO sightings into a training and testing set. This was accomplished by dividing the initial data frame according to a *split_date* (December 31, 2013). Using Pandas datetime objects it was easy to create a simple conditional; training data have dates below or on the *split_date* and testing data points have dates above the *split_date*.

After splitting the data, the first step was to create the decision tree classifier. This was accomplished using sklearn’s DecisionTreeClassifier() method. Importantly, this method of creating a decision tree classifier uses GINI impurity to determine the best feature at each step This method required numerical data for the feature values and strings as class labels. The class labels could easily be retrieved from the *Shape* column in the dataframe. In order to turn the categorical *region* and *time* data into numerical data I utilized the Pandas get_dummies() function. This method uses one hot encoding to turn one row of categorical data with n categories into n-columns of binary values.

I was then able to export the decision tree classifier to a dot file using export_graphviz(). The dot file was difficult to read as is so I used an online resource to translate the text into an easy to understand graph. The final decision tree classifier is pictured below.


time_evening <= 0.5 gini = 0.666 samples = 10715 value = [3777, 3432, 3506] class = Circle
True
False
time_night <= 0.5 gini = 0.651 samples = 3367 value = [1360, 782, 1225] class = Circle
region_northeast <= 0.5 gini = 0.665 samples = 7348 value = [2417, 2650, 2281] class = Circle
region_west <= 0.5 gini = 0.641 samples = 1622 value = [733, 371, 518] class = Circle
region_south <= 0.5 gini = 0.651 samples = 1745 value = [627, 411, 707] class = Circle
region_midwest <= 0.5 gini = 0.666 samples = 5961 value = [1949, 2089, 1923] class = Circle
gini = 0.656 samples = 1387 value = [468, 561, 358] class = Circle
region_south <= 0.5 gini = 0.651 samples = 1095 value = [468, 272, 355] class = Circle
time_afternoon <= 0.5 gini = 0.616 samples = 527 value = [265, 99, 163] class = Circle
region_midwest <= 0.5 gini = 0.657 samples = 1225 value = [453, 308, 464] class = Circle
gini = 0.63 samples = 520 value = [174, 103, 243] class = Circle
region_west <= 0.5 gini = 0.666 samples = 4208 value = [1411, 1444, 1353] class = Circle
gini = 0.665 samples = 1753 value = [538, 645, 570] class = Circle
time_afternoon <= 0.5 gini = 0.653 samples = 591 value = [238, 142, 211] class = Circle
time_morning <= 0.5 gini = 0.644 samples = 504 value = [230, 130, 144] class = Circle
gini = 0.637 samples = 226 value = [105, 50, 71] class = Circle
gini = 0.598 samples = 301 value = [160, 49, 92] class = Circle
region_northeast <= 0.5 gini = 0.656 samples = 844 value = [335, 215, 294] class = Circle
gini = 0.645 samples = 381 value = [118, 93, 170] class = Circle
gini = 0.667 samples = 2205 value = [751, 740, 714] class = Circle
gini = 0.666 samples = 2003 value = [660, 704, 639] class = Circle
region_midwest <= 0.5 gini = 0.659 samples = 297 value = [112, 78, 107] class = Circle
region_midwest <= 0.5 gini = 0.644 samples = 294 value = [126, 64, 104] class = Circle
gini = 0.629 samples = 238 value = [117, 59, 62] class = Circle
gini = 0.653 samples = 266 value = [113, 71, 82] class = Circle
gini = 0.659 samples = 569 value = [218, 150, 201] class = Circle
gini = 0.649 samples = 275 value = [117, 65, 93] class = Circle
gini = 0.655 samples = 128 value = [53, 34, 41] class = Circle
gini = 0.658 samples = 169 value = [59, 44, 66] class = Circle
gini = 0.647 samples = 147 value = [63, 34, 50] class = Circle
gini = 0.64 samples = 147 value = [63, 30, 54] class = Circle

After completing the training process it was time to test the accuracy of the decision tree. I used the test data that I separated earlier and applied the same one hot encoding on it that I used for the training data. In order to report the classification accuracy, I used the build in method predict() on the decision tree classifier object. This created a list of predicted classes which I could compare to the true list of values contained in the test data using the classification_report() function.

The result of this process, and the classification accuracy of the decision tree using the test data is listed below. Overall, the accuracy of the decision tree classifier was not good. Looking at the data, there seemed to be many 'Circle' objects which may have skewed the data. The accuracy of the 'Circle' objects are much higher (43% precision) than the next most accurate 'Fireball' object (36% precision). I believe that with more data points in favor of the 'Fireball' and 'Triangle' shape, or by reducing the number of 'Circle' samples, it may be possible to obtain a more accurate classification report.

	Precision	**Recall**	**FL-score**	**Support**
Circle	0.43	0.46	0.44	2127
Fireball	0.36	0.54	0.43	1700
Triangle	0.33	0.11	0.17	1400
Avg./Total	0.38	0.39	0.37	5227