Phil Liu
2034233
CSE 40 Spring 2024
6/2/2024
Hands-On 5 Report

1. Introduction.

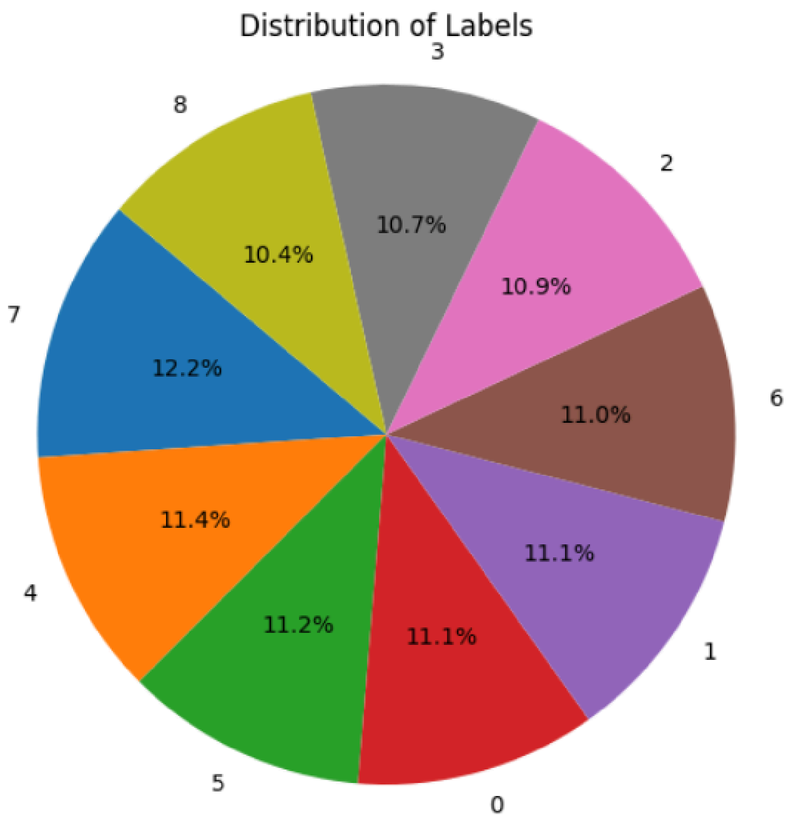
My unique dataset contained 1200 rows and 14 columns and is given in tsv(tab separated values) format. The first column in the data contains the labels for each data entry while all other columns are unlabelled and contain different data. Besides the label column, there are 10 columns containing numerical data and 3 columns containing categorical data. The numerical columns have varying units or none at all, while the categorical data is split with 2 of the columns seemingly containing animals, while the last one includes sports. My goal for the project is to utilize the columns beside the labels to classify the true label for the data entry. Some models that might be utilized are Logistic Regression, KNN classification, and the Decision Tree classifier. These are all generally more simple classifiers that don't overfit as much as more complicated classifiers such as NN. In addition, these are much easier to implement which allows me to get results much faster than training a neural network for example.

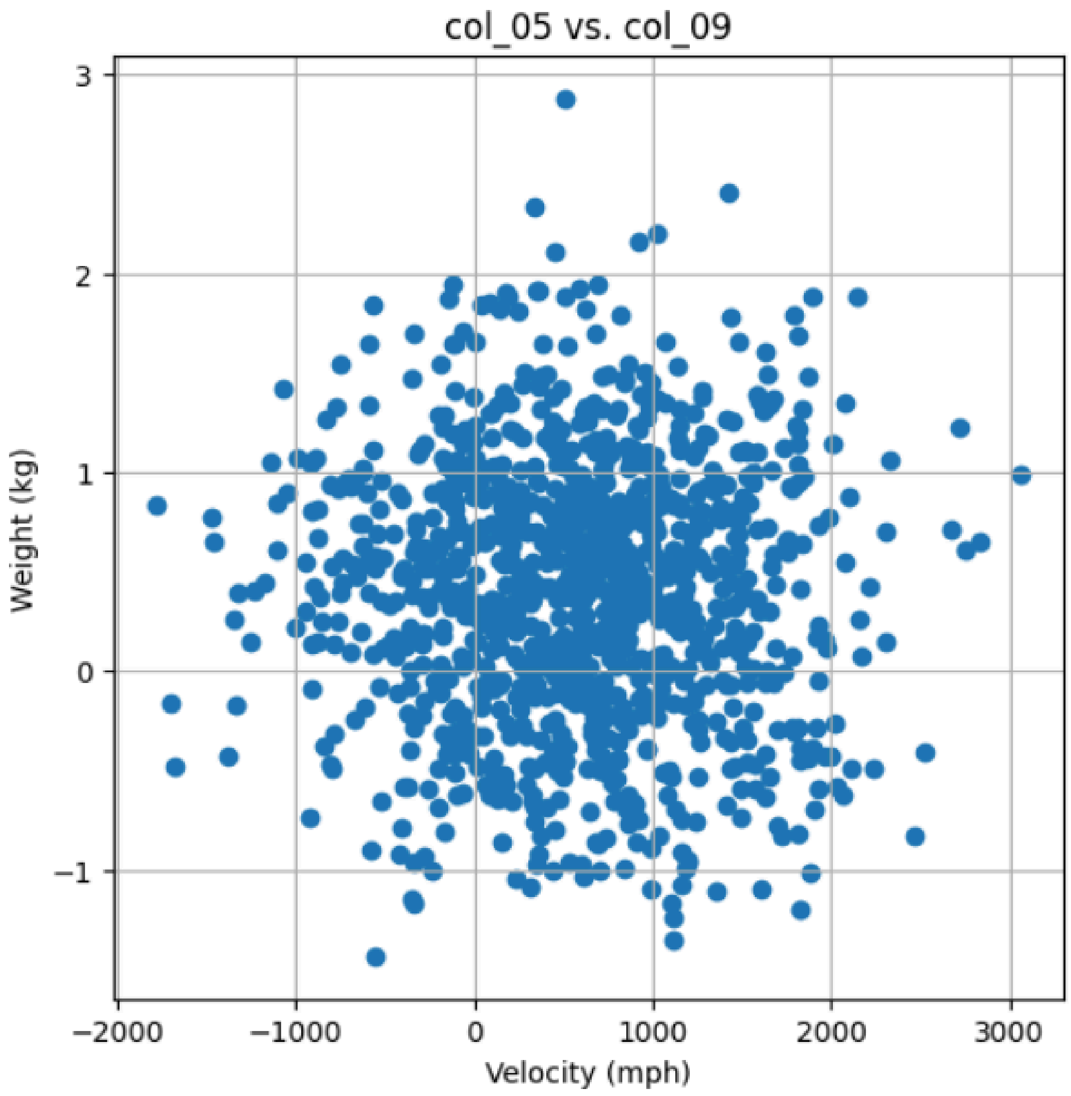
2. Data Cleaning

My first step in the data cleaning process was to simply just drop the rows for any entries that had null values in any of the columns. I saw that I still had 1077 rows of data from the original 1200 (≈90%), so I didn't expect that 10% of data to drastically change my results and just ignored those row entries. Next, I decided that I needed to separate my categorical columns from my numerical data, so I explicitly defined the categorical data and the label column to be ignored by the numerical cleaning – we'll get to those later. As a side note, I don't want to change my label column in any way so I just defined my cleaning functions to skip over the label column whenever it was encountered. To clean up my numerical data, I first defined a regular expression that would pick up and isolate any numerical values found in the data, if they existed. Then I would cast that numerical data as a float if there was a decimal, an integer if there was not. I would add all this data to a list, and replace the existing column with the list data. To set the datatype of the column, I would try and call to_numeric() on the column, and if there was an error, I would then just skip over. If no errors, the numeric columns will be classified correctly as numeric columns. For the categorical data, I decided to one-hot encode using the pandas get_dummies() function which makes the one-hot encoding easy, although certain edge cases could be improved within the function. For example, for the animal columns, the entries GUINEA PIG and guinea pig are treated as separate encodings even though we know as humans that they represent the same thing. In addition, for the animal columns, even though they both list different types of animals, the encodings are separated by columns which may or may not be beneficial depending on the lost original data format. We don't know whether this is going to be an issue or not, so I just left it for now. For encoding the sports, the encoding considered lists of sports as a different entry instead of encoding ones for both of the sports, which is an issue but given the timeline for working on this assignment, was just not feasible for me. Something to look into fixing if I were to continue working on cleaning this data. Still, for the majority of the data, using get_dummies is a convenient way to one-hot encode quickly and accurately. After this, as expected, the vast majority of the data frame is just the one-hot encoded columns, with the rest being numerical columns.

3. Data Visualization

There are a lot of different visualization libraries, but I decided to just use matplotlib. The data is a little odd to visualize since it's all unlabelled and we don't really know what the numbers and units are representing. Nevertheless, I tried to find some relationships between some of the columns. For my first pie chart, I decided to investigate whether there was any oddity in the distribution of labels themselves. I created a list with counts of all the different labels in the column, but it seems that after plotting, the labels were all distributed somewhat evenly, so there wasn't any abnormality there. After that, I decided to try and look at the relationship between two of the columns with units: col_05 with units in mph and col_09 with units in kg. I thought potentially there could be some sort of correlation between the speed and the weight of an entry, but after plotting a scatter plot, it seems like there isn't any correlation at all between the two columns despite my hopes. It seemed to me after this that there wasn't going to be any obvious correlation between any two specific columns that would make any sense so I just continued onto the modeling portion of the assignment.





4. Modeling

The classifiers that I have chosen were the logistic regression, k-nearest neighbors, and decision tree. I just used the default paramaters for the untrained classifers, and ran the k-fold validation to train each classifier. Here are some tables that summarize my classifier data.

Model	Accuracies:				
Logistic Regression	0.9907407407407407	0.9768518518518519	0.9534883720930233	0.9813953488372092	0.9534883720930233

K-Neighbors Classifier	0.2037037037037037	0.22685185185185186	0.19534883720930232	0.19534883720930232	0.1813953488372093
Decision Tree	0.9027777777777778	0.8981481481481481	0.8883720930232558	0.9116279069767442	0.9209302325581395

Taking a look at the accuracies of each test, it's clear without further analysis that Logistic Regression performs the best out of the classifiers we chose. The following table summarizes the data further:

Model	Mean Accuracy	Standard Deviation
Logistic Regression	0.97119293712317	0.015133620801208
K-Neighbors Classifier	0.20052971576227	0.014984899402765
Decision Tree	0.90437123169681	0.011172270567662

Judging based off the standard deviations, the models all have roughly around the same variance, so it's not like they perform much better on a specific subset on the data. On the t-test all of my results returned that they were statistically significant.

5. Analysis

The logistic classifier seems to work best for this data because it can easily identify the relationship between the columns even though it's much harder and more time-consuming for a human to try and match the relationships between all the columns. The decision tree also works decently because it provides a logical structure to decide and classify each data entry. I think the KNN performs so poorly because of how the data is cleaned. I hypothesize that the one-hot encoding done in the data cleaning process has somehow affected the KNN such that data points that are supposed to be closer together are much further apart, drastically decreasing the accuracy and leading to more inaccurate predictions. I think evaluating the models based on vanilla accuracy should be fine, after all, we are trying to see if using these models to classify new unseen data would be successful or not. The statistical significance between the classifiers makes sense: each of their accuracies is very far from each other and the standard deviations are generally pretty small too. I think for the Logistic Regression classifier, I could increase the number of maximum iterations to increase performance as I am still getting convergence warnings when I run my classifiers. Other than that, I could also tweak the amount of neighbors that the KNN actually checks.

6. Conclusion

In conclusion, the Logistic Regression model best predicts the labels of my unique dataset, getting up to even 99% accuracy in one of the trials. It’s interesting to note that all of the accuracies don’t have too much variance between the different folds, which means that each model is generally consistent for each of their results. I think the main point of investigation is checking and understanding why the KNN classifier did so poorly in comparison to the other models, and what can be done to the data to try and make it perform better.

7. References

(1) Pandas. (n.d.). Pandas documentation. Retrieved June 3, 2024, from https://pandas.pydata.org/docs/

(2) SciPy. (n.d.). SciPy.stats documentation. Retrieved June 3, 2024, from https://docs.scipy.org/doc/scipy/reference/stats.html

(3) Matplotlib. (n.d.). Matplotlib documentation. Retrieved June 3, 2024, from https://matplotlib.org/stable/index.html

(4) Wikipedia contributors. (2024, May 21). Student's t-test. In Wikipedia, The Free Encyclopedia. Retrieved June 3, 2024, from https://en.wikipedia.org/wiki/Student%27s_t-test