Hannah Kane

Intro to Machine Learning

Dr. Abukmail

April 24, 2023

Logistic Regression with Gradient Descent on Mushroom Dataset

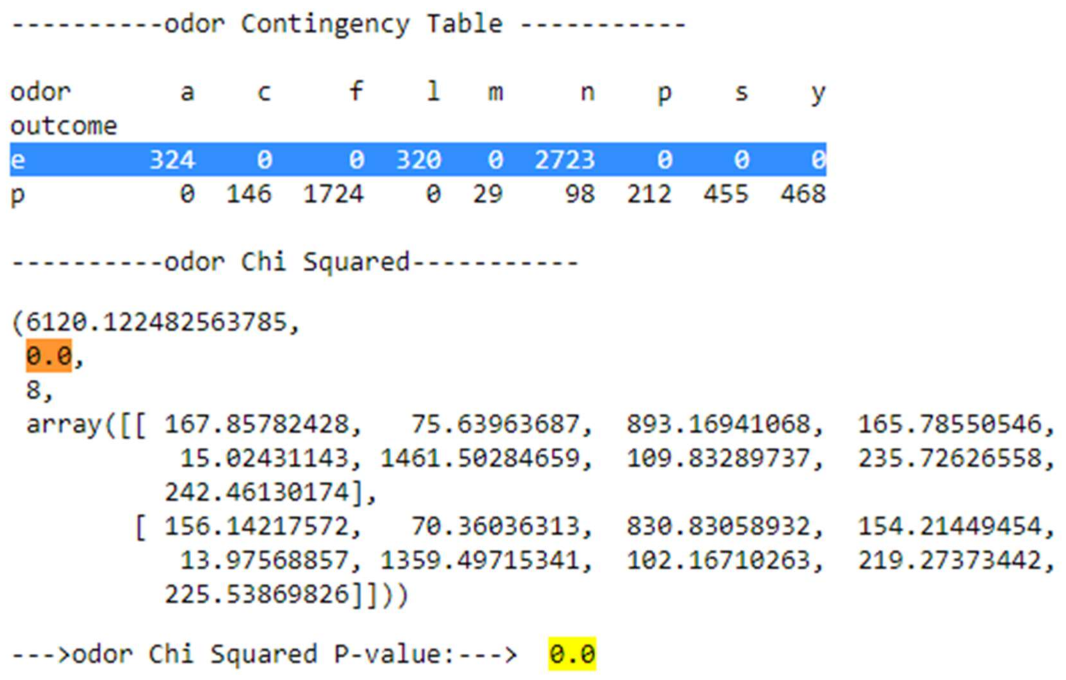
I. Introduction

This project uses the UCI Mushroom data set to implement the gradient descent algorithm. After using gradient descent to produce the weights that will minimize in-sample error a logistic regression function runs to estimate percentage likelihood that the mushroom is edible and categorizes each sample as either edible or poisonous based on a 75% likelihood threshold.

II. Methodology

First the data was analyzed on a Data Frame to get an overview of the features and their values. This revealed that the entire dataset used character values instead of numeric ones. Next the data was further analyzed in frequency tables to get a better idea of which features might be important to include. A few features could be removed immediately because they output the same value 100% of the time.

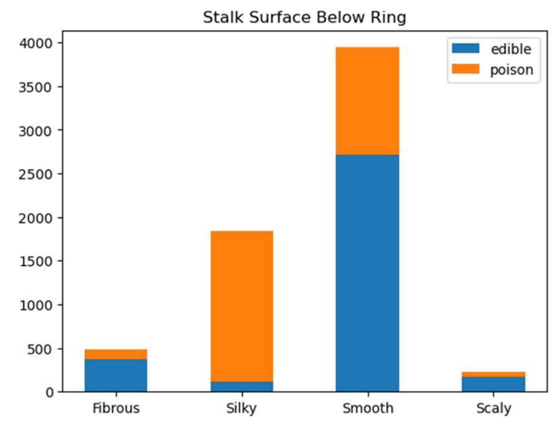
The chi-squared independence test was used next to further reduce unhelpful features. The chi-squared p-value can be used to identify if individual features have a correlation to the outcome values. A p-value near zero indicates that the feature is a good candidate to be included so all features that did not have p-values of zero were eliminated. Additionally, this test revealed that some feature values were only listed under a single outcome—either all edible or all poisonous—meaning they would indicate the outcome 100% of the time as illustrated in the odor contingency table screenshot below:

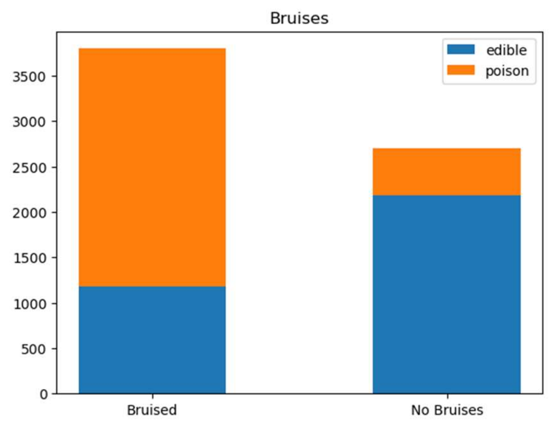


```
----------odor Contingency Table -----------

odor        a    c     f    l    m     n    p    s    y
outcome
e         324    0     0  320    0  2723    0    0    0
p           0  146  1724    0   29    98  212  455  468

----------odor Chi Squared-----------

(6120.122482563785,
 0.0,
 8,
 array([[ 167.85782428,   75.63963687,  893.16941068,  165.78550546,
           15.02431143, 1461.50284659,  109.83289737,  235.72626558,
          242.46130174],
        [ 156.14217572,   70.36036313,  830.83058932,  154.21449454,
           13.97568857, 1359.49715341,  102.16710263,  219.27373442,
          225.53869826]]))

--->odor Chi Squared P-value:--->  0.0
```

Those features were also eliminated because they would lead to an infinite descent loop. After that only six features were left to work with. Four of them were binary so the data frame was updated accordingly to replace their single character values with 0s and 1s. Finally for the two features that had multiple possible values dummy columns were added so that each possible value has its own column. When exporting the data from Jupyter notebooks for Java implementation only if a feature has k dummy columns only k – 1 of those was exported—in this case 3 of the 4 dummy columns for each feature—to avoid multicollinearity.

Bar charts were used to further visualize the data in Jupyter notebooks but no further features were removed based on this step.





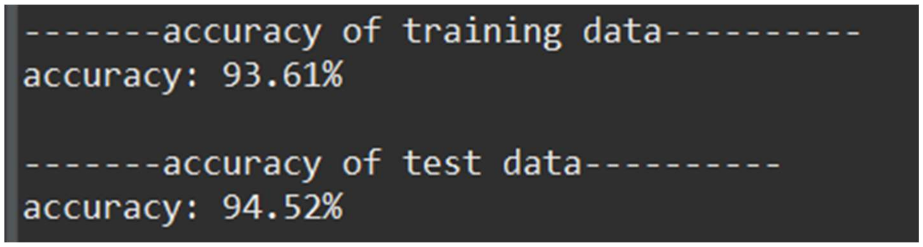
The Java implementation includes the option for users to enter the information about a single mushroom and receive the prediction based on the weights the training data provides to the logistic regression function.

The user will need to

III. Results

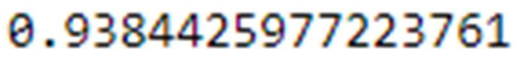
After implementing the gradient descent and logistic regression in Java the accuracy nearly matched that of SciKit Learn as seen below:

Java implementation accuracy:



```
-------accuracy of training data----------
accuracy: 93.61%

-------accuracy of test data----------
accuracy: 94.52%
```

Scikit Learn implementation accuracy:



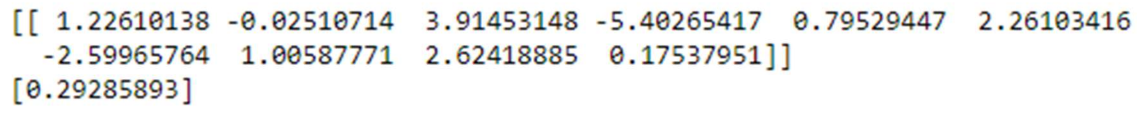
```
0.9384425977223761
```

The weights from Scikit Learn are not close to the weights in the Java implementation.

Java implementation weights:

```
Weights: [-1.5601286788066526, 0.9550112696077162, 0.9582438675414489, -
2.793454078515897, 2.8353771897566276, 0.2832940283821576, -
0.498736487150146, 1.7226095925902989, 0.5787719002553057, -
0.9133097376216112, 0.9206425127243127]
```

Scikit Learn implementation weights:



```
[[ 1.22610138 -0.02510714  3.91453148 -5.40265417  0.79529447  2.26103416
  -2.59965764  1.00587771  2.62418885  0.17537951]]
[0.29285893]
```

This is likely because several fits will produce a similar accuracy using this data. Unfortunately increasing the prediction threshold in Java does reduce the accuracy quite a bit.

IV. Conclusion

Overall, the project was a success in both knowledge gained during the successful implementation of the gradient descent and logistic regression as well as the results roughly matching that of Scikit learn. In general, the model does not have a practical application because the mushroom dataset does not include a large enough variety of different kinds of mushrooms to be useful to someone trying to make a prediction based off field data. Despite that it was a fun dataset to implement the gradient descent and logistic regression on and the implementation could be modified to accommodate a larger dataset with more real life applications.

Note to TA: Please contact me at kaneh6777@uhcl.edu if you have any issue running the program. It should run without error as long as the .csv files are in the same directory as the .java file. I am happy to meet with you and troubleshoot any error that may arise from running it on a different device. Unfortunately, my laptop is very old and not set up to run it but if needed I can attempt to set that up, I primarily work on my desktop. Please let me know ASAP if it seems like that will be necessary.