Written Task: Introduction

Due to an unforeseen tragedy, we have lost all metadata associated with our dataset. Luckily, we still have the labels category in the ‘label’ column. In this project, I am constructing a trinary classifier that successfully maps an input vector to an output of 0, 1, or 2. Our input data contains 12 different input columns. Most of these columns will be converted to float data types while a few of them will be converted to type string. I initially propose using logistic regression, a random forest classifier, and K-nearest neighbors. Both logistic regression and K-nearest neighbors are simple to implement and outputs that are simple to interpret. Easily interpretable outputs will be beneficial in a dataset that lost its metadata. Conversely, a random forest classifier is a more complex model with harder to interpret data. The more complex model is a test to see if the results improve with increased model complexity.

Written Task: Data Cleaning

Originally, the vast majority of our data is of type “object.” Obviously, this is incorrect as there are quantitative (numeric) and qualitative (string) data. Initially, I convert the object columns that are supposed to be numeric data by using a generalized regular expression. The generalized regular expression works well to catch all the variations of numbers on the first try, since they could be large numbers or small decimals. Unsurprisingly, some holes exist within my numeric columns. If my regular expression is unable to capture a number, I simply replace the data with a 0. With more information about the data, another representation of missing data might have been a better option. I do not alter any potential outliers in my data as I do not understand enough about the data to warrant any outlier smoothing. For the columns containing strings, I used one-hot encoding to convert the strings to numeric data. In my data, there are overlapping terms in columns 02 and 03. For example, computer science appears in both of these columns. I decided to differentiate between finding computer science in column 02 or 03. So, my one-hot encodings are column sensitive. However, the one-hot encodings are not case sensitive which standardizes the seemingly complex data into much more manageable data. Whenever my one-hot encoding algorithm comes across missing words, such as null or n/a, the algorithm ignores these words and does not encode a “null” or “n/a” column. After finding numbers through regular expressions or

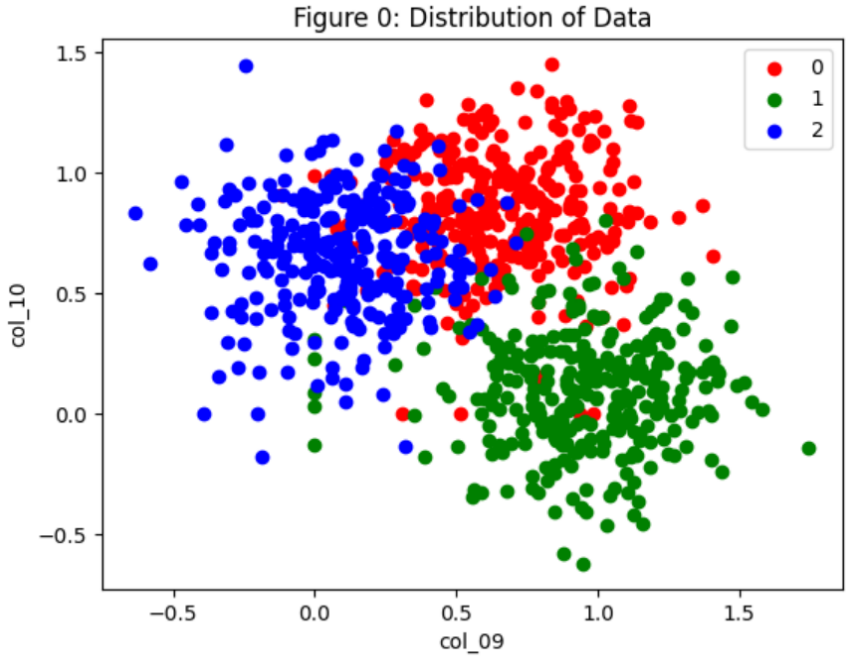
one-hot encoding the data, all columns were of type integer or float. Since there were many possible keywords in the string columns, the data went from having 12 dirty columns to 53 clean, numeric columns. One drawback of this approach is that the data frame requires a large amount of storage. Perhaps in the future I can look to remove unnecessary data columns.

Written Task: Data Visualization

Table 0

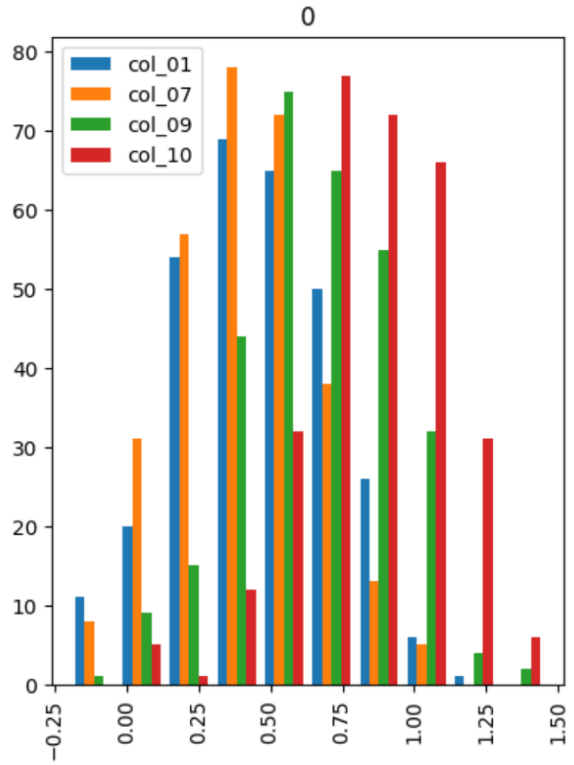
Column	P Values
col_11	1.13E-171
col_10	3.07E-48
col_09	5.65E-37
col_08	1.35E-161
col_07	3.11E-59
col_06	3.12E-273
col_04	8.58E-223
col_01	5.48E-78
col_00	3.25E-273

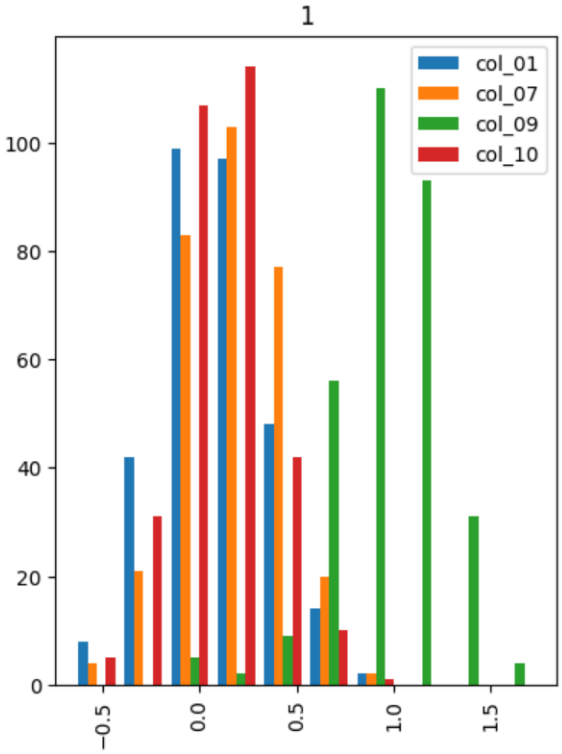
In Table 0, we can see some different columns and their correlation to the ‘label’ column. The P values come from a t test. The larger the value, the more correlated there is between the column and the ‘label’ column. For my data visualization, I can only display 2 of the many dimensions of this vector space. Since the one hot encoded data is sparse, it is not very correlated with the label. For this reason, all columns that were one hot encoded are not displayed in Table 0. The two columns that are most correlated to the ‘label’ column are ‘col_09’ and ‘col_10’. Since these two columns are the most correlated, they will be used to represent the x and y axis of the feature space to help us best visualize the data.

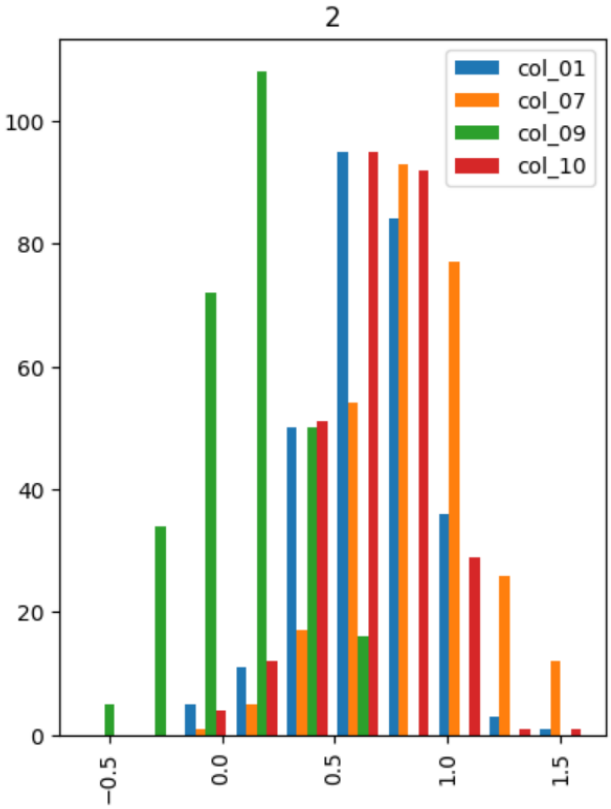


In the scatter plot shown in Figure 0, some minor trends can be seen. Since there are over 50 feature vectors, I had to choose the two best columns to make a scatter plot. Col_09 and col_10 have the highest correlation to the output label, so they were the clear choice to represent a large dimensional space in 2-Dimensions. The x-axis represents the value in col_09 and the y-axis represents the value in col_10. The red, green, and blue dots represent labels of 0, 1, and 2, respectively. With this simple reduction, we see that there are clear trends between the output labels and the two columns.

Histograms 0, 1, 2


0
col_01
col_07
col_09
col_10
80
70
60
50
40
30
20
10
0
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50

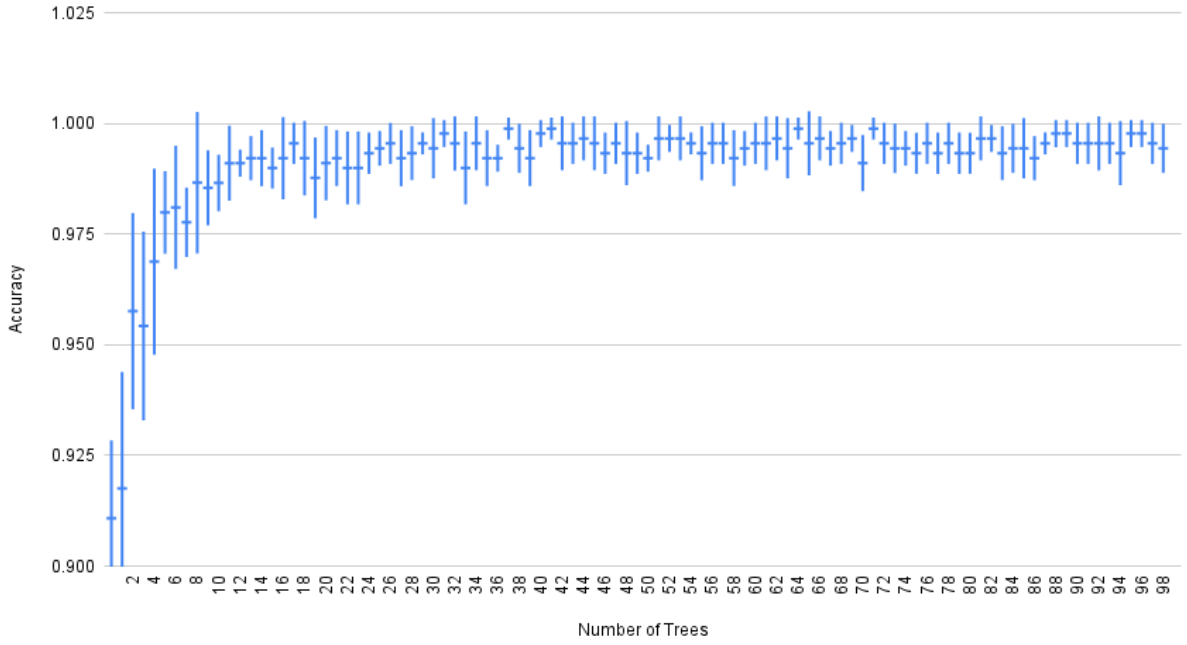

1
col_01
col_07
col_09
col_10
100
80
60
40
20
0
−0.5
0.0
0.5
1.0
1.5


2
col_01
col_07
col_09
col_10
100
80
60
40
20
0
−0.5
0.0
0.5
1.0
1.5

As seen in the above histograms, I split the data based on the ‘label’. Each histogram represents the data found across four columns as seen in the legend. These four columns have the highest correlation to the data, as seen in the p value chart, Table 0. The x axis represents the value of the particular column. The y axis represents the frequency of the value occurring. From this simple view of the data, we notice some simple trends in the data. Looking at ‘col_09’, the green columns, we see that label 1 has more positive values, label 2 has more negative values, and label 0 has a more normal distribution around 0. Additional trends can be seen with the other 4 columns. Sadly, since the label and columns have lost their metadata, we cannot extrapolate real world information out of the trends we visualized in these histograms.

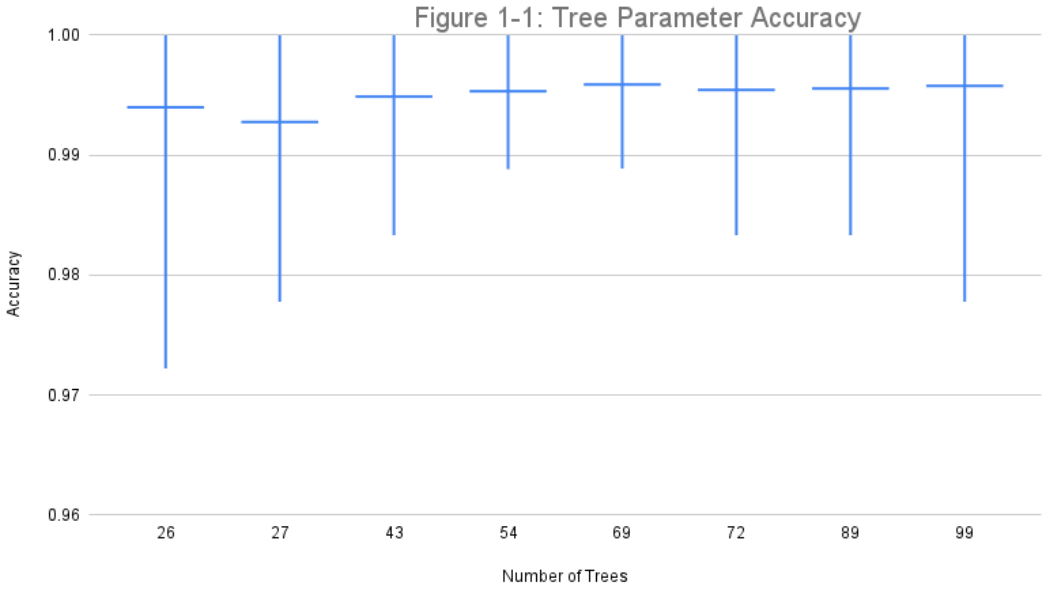
Written Task: Modeling

Figure 1: Random Forest Accuracy

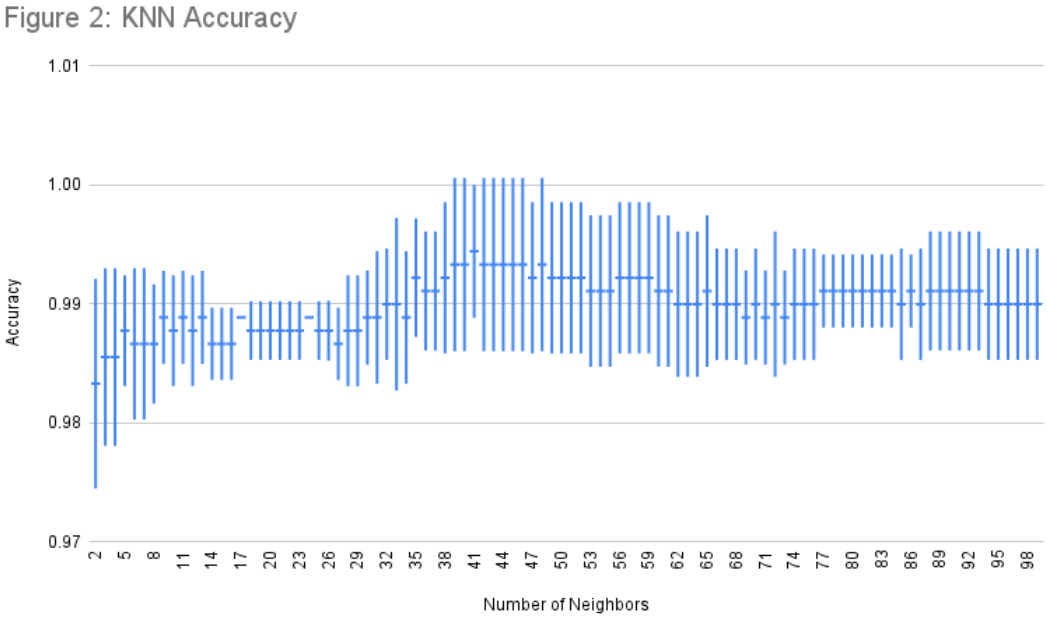


In Figure 1, I am testing the effects of changing the number of trees parameter that is passed to the random forest algorithm. As we can see in Figure 1, many different values for the number of trees yield good results. Additionally, these statistics change based on the way our data is split by KV fold split. So, I ran the above visualization 10 times and recorded the best, highest average score each time. The ten best averages were [43, 26, 72, 54, 21, 69, 27, 99, 69, 89]. The winners of these trials confirm the trends we see

in Figure 1. Many different numbers of trees above a certain threshold perform well. But, there is no clear pattern or winner. So, I took these 10 winning values and investigated further.

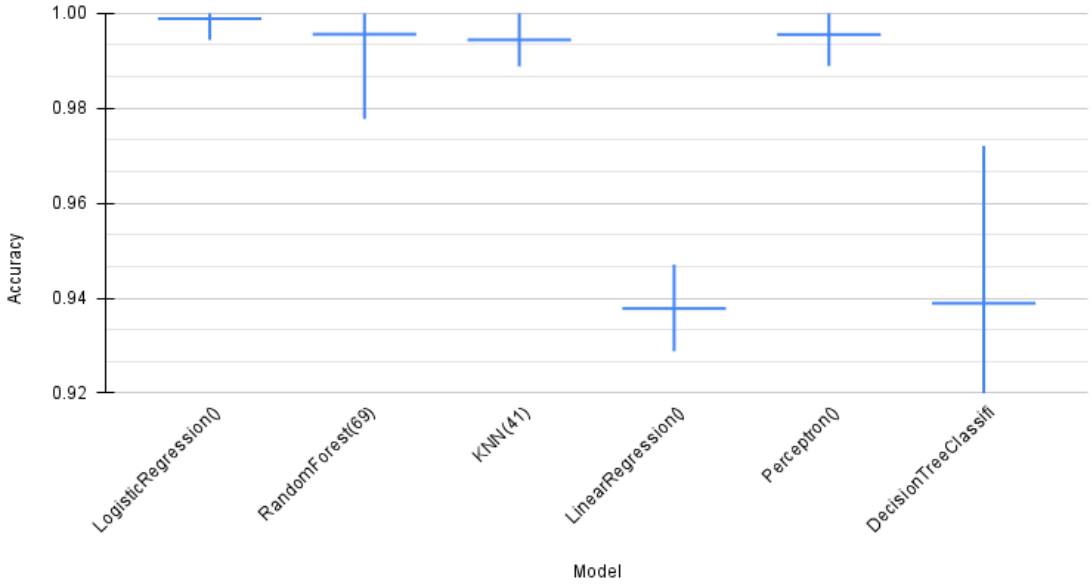


In Figure 1-1, we see that all ten of these perform well in our additional trials. All trees scored a high of 100 percent accuracy, but the average and minimum values changed slightly. After these additional tests, the random forest with a parameter of 69 trees had the best average and minimum value. So, when comparing the random forest algorithm, my program will use the best parameter, which is 69 trees.



In Figure 2, I am testing the effects of altering the number of neighbors, the parameter, passed to the KNN algorithm. As we can see, there is a general trend that shows we do not want too few or too many neighbors to produce good results. Luckily, the number of neighbors algorithm is deterministic. So, when testing the best parameter, 41 neighbors is always the best choice for my data set. Using 41 neighbors as the parameter for the KNN model produces the best average accuracy on my dataset. For future comparisons between algorithms, the KNN algorithms will use a parameter of 41 neighbors.

Figure 3: Model Accuracy



In Figure 3, I am testing the accuracy against a KV fold using six different machine learning modeling algorithms. For the random forest and KNN models, I am using the best parameters, as described above. The middle line represents the average performance of the model, with the top and bottom being the maximum and minimum performance, respectively. All of these models performed well. The top four models in this test are the Logistic Regression, Random Forest with 69 trees, KNN with 41 neighbors, and the Perceptron algorithm.

Table 1

Model	Mean Accuracy	Standard Deviation
LogisticRegression()	0.9988826816	0.002236874866
RandomForest(69)	0.995603414	0.00452699373
KNN(41)	0.9944134078	0.005001804257
LinearRegression()	0.9378069544	0.007049785688
Perceptron()	0.9955431409	0.004164908503
DecisionTreeClassifier()	0.9389240844	0.01610489493

The statistics we see in Table 1 confirm the visualization shown in Figure 3. All of these algorithms have high accuracy and relatively low standard deviation. The best algorithm, Logistic Regression, has both the highest mean accuracy and the lowest standard deviation. Thus, Logistic Regression is clearly the best algorithm for predicting.

```
LogisticRegression vs RandomForestClassifier: False
LogisticRegression vs KNeighborsClassifier: False
LogisticRegression vs LinearRegression: True
LogisticRegression vs Perceptron: False
LogisticRegression vs DecisionTreeClassifier: True
```

I compare the best algorithm, Logistic Regression, with the other classification algorithms to test for a statistically significant performance difference. Similar to the conclusions above, the top four models, Logistic Regression, Random Forest, KNN, and Perceptron are not significantly different from one another. However, there is a statistically significant difference between the Linear Regression and Decision Tree classification algorithms when compared to Logistic Regression. For this reason, I recommend against using either of these two models.

Written Task: Analysis

As discussed above, there are correlations between the various columns and the labels. Many of these correlations are weak. But by combining the correlations by considering all of the columns at once, the machine learning models achieved a high accuracy. I

recommend using Logistic Regression as discussed in the Modeling section. In that section I show that other models perform as well as Logistic Regression while others fall behind. Since the accuracy of the model is calculated on a KV fold, there is a high confidence that the algorithm will perform well in the real world. As shown in the Data Visualization section, there are some features that have much higher correlation to the label than others.

Written Task: Conclusion

The incoming data was dirty, but not irreparable. By extracting numeric values and one hot encoding the regularized versions of the strings, I was able to make many numeric feature vectors. However, there were difficulties in determining the correlation between the columns and the output label, due to the lack of information surrounding the input data and output labels. Instead of relying on intuition, I used statistical analysis to find the correlation between the data. A possible point of further improvement could be feature trimming. I did not test the effects of removing features with extremely low correlation to the output data. Through extensive parameter testing, I found the best parameters for certain models. By thoroughly testing the accuracy of six different models, I was able to find a model that achieved extremely high accuracy with low standard deviation on data from a KV fold. I was able to compute the statistical significance between the performance of these models. I found that Logistic Regression performed provably better than some other models. For this reason, I recommend using a Logistic Regression model to accurately predict labels for new, unseen data.

Written Task: References

I used the tools from all of the previous Hands On Assignments. I did not reference any published papers or studies when writing my report.