Luke Scott

COSC 311

Project Report

Dr. Wang

Project Report

Task 1: For parameters for the KNN model, I experimented with the n_neighbors parameter, and found that using the value 4 provided the highest score for the model. For the DT model, I set the criterion parameter to entropy, then conducted a standard self-test on the data. The DT model scored perfectly compared to the KNN model that scored 99%, this is because the data wasn't split.

KNN MODEL:

```
Confusion matrix and classification report for knn
model:
 [[500   0   0   0]
 [  0 496   4   0]
 [  3   3 493   1]
 [  2   0   1 497]]
              precision    recall  f1-score   support

           1       0.99      1.00      1.00       500
           2       0.99      0.99      0.99       500
           3       0.99      0.99      0.99       500
           4       1.00      0.99      1.00       500

    accuracy                           0.99      2000
   macro avg       0.99      0.99      0.99      2000
weighted avg       0.99      0.99      0.99      2000
```

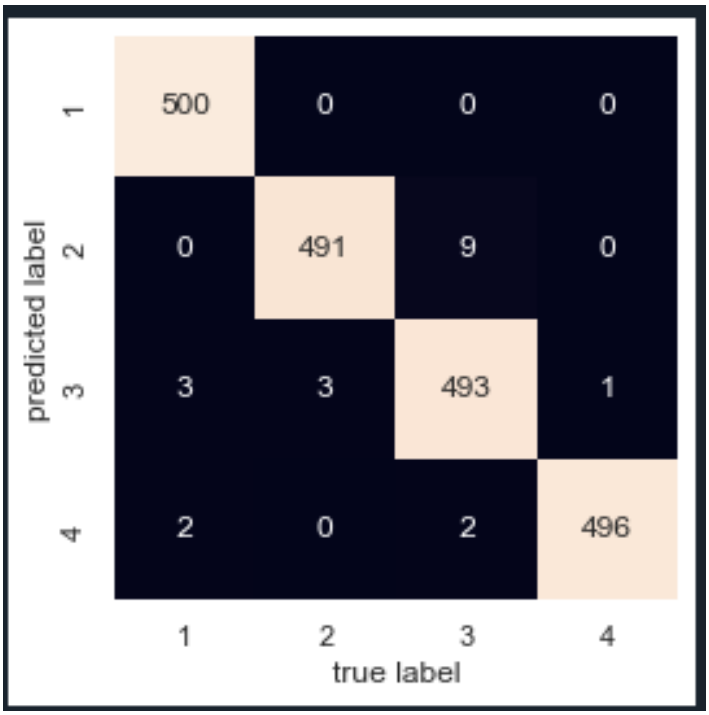
DT MODEL:

```
Confusion matrix and classification report for dt
model:
 [[500   0   0   0]
 [  0 500   0   0]
 [  0   0 500   0]
 [  0   0   0 500]]
              precision    recall  f1-score   support

           1       1.00      1.00      1.00       500
           2       1.00      1.00      1.00       500
           3       1.00      1.00      1.00       500
           4       1.00      1.00      1.00       500

    accuracy                           1.00      2000
   macro avg       1.00      1.00      1.00      2000
weighted avg       1.00      1.00      1.00      2000
```

Task 2: After the split, I kept the KNN parameters the same, however this model performed slightly worse compared to the KNN model performed before the split. I changed the parameters on the DT model as I wanted to change the max_depth to 4 in order to give a proper number of classifiers for the data. The model performed worse when going above or below 4 for this value.

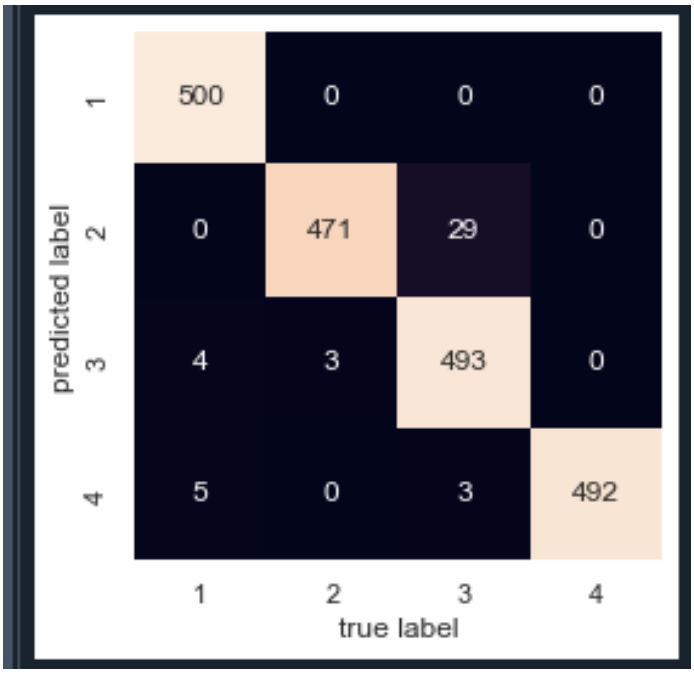
KNN SPLIT MODEL:

```
Confusion matrix and classification report for split
knn model:
 [[500   0   0   0]
 [  0 491   9   0]
 [  3   3 493   1]
 [  2   0   2 496]]
              precision    recall  f1-score   support

           1       0.99      1.00      1.00       500
           2       0.99      0.98      0.99       500
           3       0.98      0.99      0.98       500
           4       1.00      0.99      0.99       500

    accuracy                           0.99      2000
   macro avg       0.99      0.99      0.99      2000
weighted avg       0.99      0.99      0.99      2000
```



DT SPLIT MODEL:

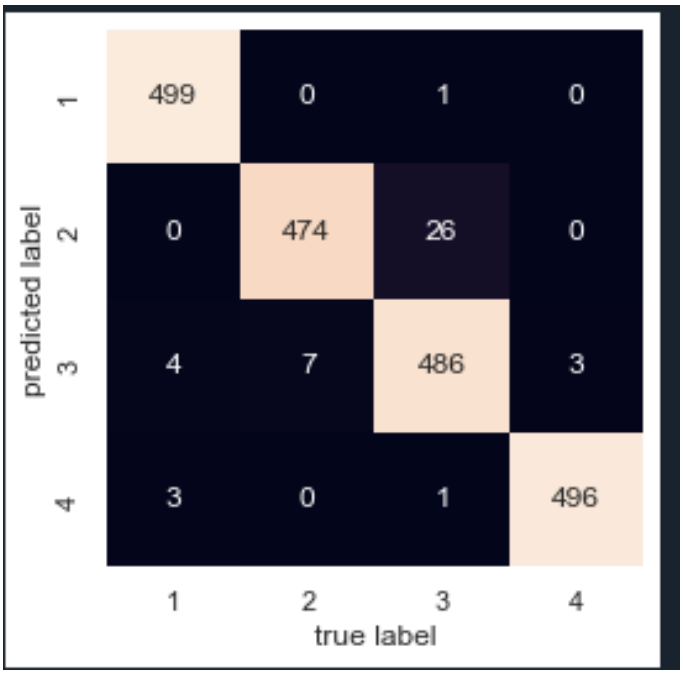
```
Confusion matrix and classification report for dt split
model:
 [[500   0   0   0]
 [  0 471  29   0]
 [  4   3 493   0]
 [  5   0   3 492]]
              precision    recall  f1-score   support

           1       0.98      1.00      0.99       500
           2       0.99      0.94      0.97       500
           3       0.94      0.99      0.96       500
           4       1.00      0.98      0.99       500

    accuracy                           0.98      2000
   macro avg       0.98      0.98      0.98      2000
weighted avg       0.98      0.98      0.98      2000
```



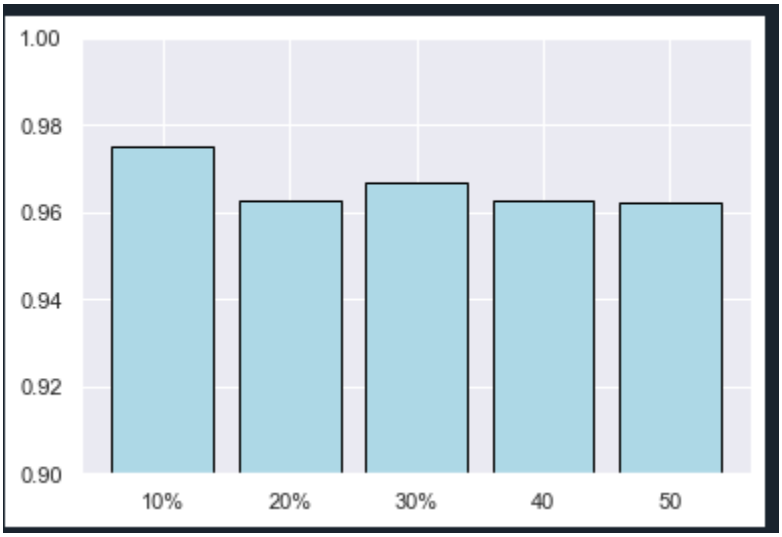
Task 3: I chose to use the DT model to try to achieve the best results. Most modifications to the parameters didn't improve the score, however I was able to slightly improve the score by increasing the max_depth, changing the criterion to 'gini', and adding a min_samples_leaf and min_samples_split value of 5. This increased the score to almost 97%, which is still not better than the KNN score.

DT MOD:

	precision	recall	f1-score	support
1	0.99	1.00	0.99	500
2	0.99	0.95	0.97	500
3	0.95	0.97	0.96	500
4	0.99	0.99	0.99	500
accuracy			0.98	2000
macro avg	0.98	0.98	0.98	2000
weighted avg	0.98	0.98	0.98	2000



The classification reports and confusion matrices for all the split tests (10%, 20%...) are shown in the .py file. Here is the bar figure displaying the difference in scores for these tests.



These test results are quite interesting because they decrease as the testing data pool is increased. With this data set, it seems that the model performs best with a smaller set of training data.