PREDICTION REPORT

Review Stars Predcition

Model Selection

In order to accurately choose the best model to predict the review count stars, I first needed to clean up and convert the dataset. To do this, I used the skeleton code sent to the class and amended it to convert the text to integers. To narrow my dataset, I sought used only restaurants from the states of Wisconsin, Arizona, and North Carolina. In my dataset_json_to_csv2 code, I amended it to where if the business was in the state, use int 1 and otherwise use int 0. For example, for Wisconsin, the code looked like:

```
cell_wi = (cell_state == "WI")
                    if cell_state == "WI":
                              data_wi.append(1)
                    else:
                              data_wi.append(0)
```

This was done for the states of North Carolina and Arizona as well. I did the same process to narrow if it was a restaurant, the restaurant price, and if it was by_appointment. With over 30 possible attributes to use, it is well understood that the more attributes I would have used, the higher my accuracy score would be. For the sake of time and this project, however, I decided to narrow the dataset to a generalized form. After writing this code and saving the dataset to a .csv, I wrote my runme.py code which contained two for loops:

```
n_estimators_list = [5,10,20,30,40]
max_depth_list = [5,10,20,40,50]
results_list = []
```

```
for i in max_depth_list:
    machine = RandomForestClassifier(n_estimators=20, criterion='gini', max_depth=i,
n_jobs=4)
    accuracy_score, confusion_matrix = kfold_template.run_kfold(4, data, target, machine)
    results_list.append(['gini ',accuracy_score,str(i),'n_estimators = 20'])
    print(accuracy_score)
    for i in confusion_matrix:
        print(i)


for j in n_estimators_list:
    machine = RandomForestClassifier(n_estimators=j, criterion='gini', max_depth=20,
n_jobs=4)
    accuracy_score, confusion_matrix = kfold_template.run_kfold(4, data, target, machine)
    results_list.append(['gini ',accuracy_score,str(j),'max_depth_list = 20'])
    print(accuracy_score)
    for i in confusion_matrix:
        print(i)
```

First, I needed to set the list for n_estimators and max_depth, by choosing a range of integers. The first for loop ran with n_estimators set at 20 but running each new max_depth integer from the list. The second for loop ran with max_depth set at 20 but running each new n_estimator integer from the list. This gave me a range of accuracy scores and confusion matrixes with my RandomForest code which I could then compare and choose the best fitting model. These accuracy scores and confusion matrixes can be viewed in my runme_regression_results_text and runme_prediction_csv files.

To compare another model, I wrote a linear model, using a logistic regression on the same dataset. These accuracy scores can be seen in the linear_regression_prediction_csv file. I was then able to compare the accuracy scores from my RandomForest model and Linear Regression model. The accuracy scores are roughly as below:

Linear Regression
[0.343275,   0.340225,   0.340175,   0.34245]

RandomForest
[0.349475, 0.342925, 0.34445, 0.34685]

There are a couple things to note with these accuracy scores. First, the accuracy scores are low for both. Using a real-life imperfect dataset makes it difficult to achieve a high accuracy score. There are many variables which affect the accuracy of this dataset. Additionally, the more attributes used, the higher accuracy score will be achieved. Second, the accuracy scores are very similar. This is to be expected, as they are trained for the same dataset. With this, however, you can see that my RandomForest model produces a higher accuracy score. Because of this, I chose my RandomForest model as a good model selection for the prediction of the review count stars.

To further my accuracy, using the for loops in my RandomForest model, as taught in class, allowed me to compare each of the classifiers and choose the best set of n_estimators and max_depth. I found that the combination of n_estimators =20 and max_depth =40 produced the most accurate confusion matrix.

Prediction

After proving that my RandomForest model was superior, and finding the best combination of classifiers, I created my runme_prediction.py code. I converted the business_no_stars_review.json to a .csv file in the same way as done previously. Using the classifiers and RandomForest model, I predicted the review count stars for this new dataset, which can be viewed in my runme_stars_prediction_csv.csv file.

Conclusion

It is important to understand the process I used to compare both models, choose the best, and predict the stars. It is by no means perfect (i.e. the accuracy scores are low, I only used 10 different combinations of n_estimators and max_depth, I only predicted this off of a few dataset attributes, etc.), but given an imperfect dataset, I believe that the RandomForest model I created is a better option than the Linear Regression model. Running the RandomForest model with many different classifier combinations allowed me to choose the best combination, improving my RandomForest model selection.