Data Mining Project part 2

Omri Baron, Ilay Tzuberi

April 2022

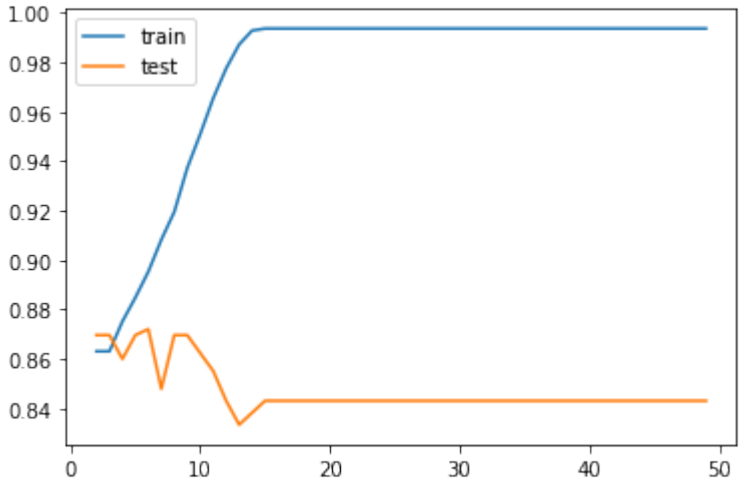
The Important Metrics

In this assignment we're trying to make the most efficient marketing campaign. This means we want to send advertisements to the smallest amount of people while getting the maximum amount of sales. Therefore our most important metric is accuracy/precision followed by AUC and lastly recall isn't really a priority but when choosing between similarly performing classifiers we'll use it as a tie breaker.

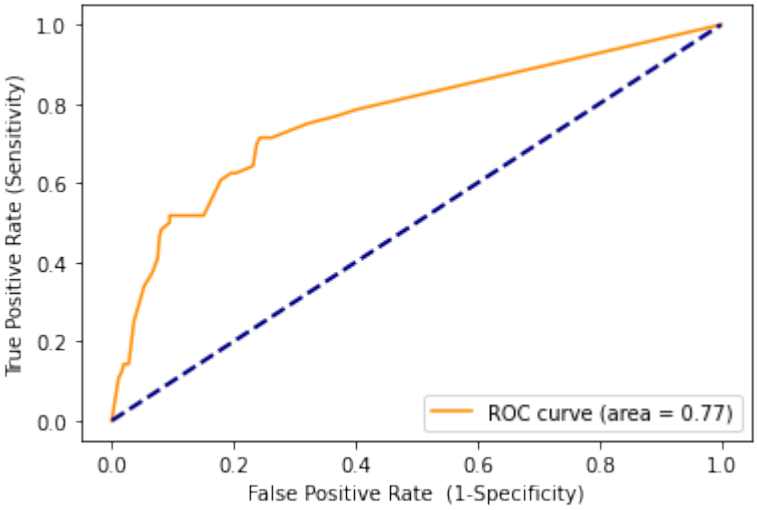
Project Work

We'll now go over the jupyter notebook and explain what we did.
After reading the csv file containing our clean data from part 1 we noticed there is an unnamed column with the indexes of the data. This column interfered with the classification and since it has no meaning we removed it.
Next we take a look at different classifiers beginning with decision tree. We created a classifier with default parameters and showed the big tree it makes. This causes overfitting as we have near 1 accuracy on the train data and 0.83 on the test.
To solve the overfitting problem we limited the depth of the tree as well as adding more parameters. This gave us an accuracy of 0.89 on the train and 0.85 on the test.

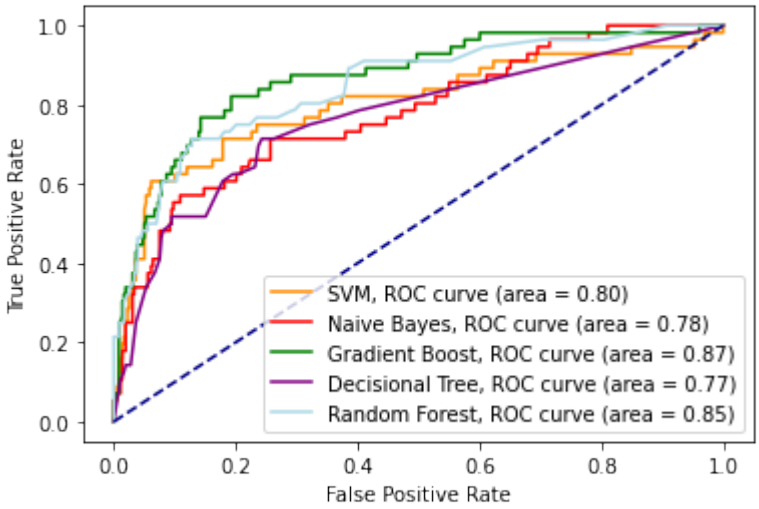
The following graph shows the effect tree depth has on accuracy. As can be seen from the graph for a low depth the difference in accuracy between the train and test data is minimal but for depth bigger than 5 the difference become big fast meaning there is overfitting. We then present a ROC curve.



We want the area under the graph to be as close to 1 as possible. Here we have 0.77, which is decent, but maybe we could do better with a different classifier.



We then looked at the following classifiers: SVM, naive bayes, random forest, and gradient boosting.
The gradient boosting classifier wasn't taught in class. It is part of the sklearn library. From the documentation in sklearn: "builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage n_classes_ regression trees are fit on the negative gradient of the binomial or multinomial deviance loss function."
Gradient boosting works similarly to random forest, they both work by combining outputs of individual trees. Although, while random forest works by building independent decision trees and combining them in parallel. Gradient boosting uses a method called boosting, which works by combining weak learners in a sequence so each one corrects the errors of the previous. These "weak learners" are usually trees with only a single split.
We plotted a ROC curve for all the classifiers together and it looks like this:



We also performed cross validation and looked at the average stats of each classifier:

```
GradientBoostingClassifier(random_state=42)
mean accuracy:  0.8742971887550202
mean recall:  0.3108619206279205
mean AUC:  0.8310480423280839
GaussianNB()
mean accuracy:  0.7786454910551296
mean recall:  0.5129132652449623
mean AUC:  0.7553952800092516
SVC(C=0.9, probability=True, random_state=42)
mean accuracy:  0.8688864549105514
mean recall:  0.18915137591043696
mean AUC:  0.7656776358990701
DecisionTreeClassifier(criterion='entropy', max_depth=8, min_samples_leaf=5,
                       min_samples_split=15, random_state=42)
mean accuracy:  0.8453048557867835
mean recall:  0.2849710367850915
mean AUC:  0.720305656391347
RandomForestClassifier(random_state=42)
mean accuracy:  0.8694742606790798
mean recall:  0.3082533648286213
mean AUC:  0.8279930303695953
```

From these results we expect the gradient boosting or the random forest classifiers to be our final choice, but there are still more things we can do to try and improve our classification.

Hyperparameter Tuning

We used both randomized and grid search on all the previously mentioned classifiers and calculated the improvement in accuracy. Unfortunately we didn't get many improvements the notable ones are:

- Naive Bayes with random search has 0.28% improvement
- Decision tree with random search has 9.92% improvement
- Decision tree with grid search has 5.34% improvement
- Gradient boost with grid search has 0.71% improvement

For the random search on random forest we chose the parameters that were used in the TA classes. Since gradient boost works similarly to random forest and has similar parameters we used the same parameters for the gradient boosting classifier with the addition of the learning rate and subsample that affect the learning.
For Naive Bayes there are only 2 parameters: priors and var_smoothing. since we wanted to change something, and priors looked difficult to change, we tried var_smoothing and as we got a positive result we kept it. lastly, SVM has many parameters, but after reading about their effects in the documentation only gamma, C and tol seemed relevant.
After doing all that we used cross validation to find the best model to use for classifying the test data. Looking at the averages the models with the highest accuracy and AUC are the gradient boost and random forest in all different versions (default, with random search, and with grid search). Which matches our expectations from looking at the ROC curves previously. To find if there is a definitive answer to which one is better, we did a "tournament" using ttest. At the end of said tournament, we reached the conclusion that the best model is gradient boost with grid search.

All that remained to do was clean the test data. We chose to stay with the cleaning method we used in part 1 of the project since we got an accuracy of 86% on average in the cross validation on gradient boosting with grid search which we consider satisfactory. We the used our chosen classifier on the test data and displayed a distribution plot of the prediction which looks like so: