ComS 573 Machine Learning

Lab 4 - Ensemble Learning

John Rachid

Task 1

Random Forest

Determining optimal hyperparameters based on training data

mtry	Accuracy	Kappa
2	**0.7247823**	**0.1952071**
3	0.7085137	0.1684876
4	0.7053391	0.1606402

Final Model on testing data

```
          Reference
Prediction  no yes
       no  218  49
       yes  20  14

               Accuracy : 0.7708
                 95% CI : (0.7191, 0.817)

    No Information Rate : 0.7907
    P-Value [Acc > NIR] : 0.8220501
                  Kappa : 0.1663
 Mcnemar's Test P-Value : 0.0007495
            Sensitivity : 0.9160
            Specificity : 0.2222
         Pos Pred Value : 0.8165
         Neg Pred Value : 0.4118
             Prevalence : 0.7907
         Detection Rate : 0.7243
   Detection Prevalence : 0.8870
      Balanced Accuracy : 0.5691
```

AdaBoost.M1

Determining optimal hyperparameters based on training data

maxdepth	mfinal	Accuracy	Kappa
1	3	0.7427216	0
1	6	0.7427216	0
1	9	0.7427216	0
3	3	0.74799	0.2214689
3	**6**	**0.7539492**	**0.234626**
3	9	0.7532085	0.2451847

Confusion Matrix and Statistics for the optimal model from training

```
Prediction  no yes
       no  227  47
       yes  11  16

               Accuracy : 0.8073
                 95% CI : (0.7581, 0.8503)

    No Information Rate : 0.7907
    P-Value [Acc > NIR] : 0.2646
                  Kappa : 0.263
 Mcnemar's Test P-Value : 4.312e-06
            Sensitivity : 0.9538
            Specificity : 0.2540
         Pos Pred Value : 0.8285
         Neg Pred Value : 0.5926
             Prevalence : 0.7907
         Detection Rate : 0.7542
   Detection Prevalence : 0.9103
      Balanced Accuracy : 0.6039
```

Discussion

These models had their hyper parameters determined by a grid search tune control featured in the caret library. This allows the program to determine the optimal hyperparameters for that model.

With AdaBoost.M1 I was not able to do a full grid search as the train time seemed to take an extremely long time even with parallel processing. As a result, it went from 1:9 as the mfinal hyperparameter and 1 to 3 as the depth of the model. The learning coefficient with the best result was “Breiman”. I do not know what this is however it did give the best results in my experiments.

With random forest, the tested hyperparameters were from an mtry of 2 to 4. The Accuracy of the tested models with random forest was lower for both the testing and training data. This is quite surprising as this model was trained with 10 fold validation and 5 repeats compared to Adaboost.M1 which used 5 fold validation and 3 repeats.

Task 2

Individual Models

When training the individual models I used tune control which allowed Caret to test the hyperparameters and choose the one which resulted in the highest accuracy. Below I will list the experiments on the training data which tune control used to determine the optimal hyperparameters. The bolded is the optimal model.

All of the training used 7 fold validation repeated 3 times. I decided on 7 fold as it seemed to give the optimal performance.

Neural Network

Determining optimal hyperparameters based on training data

Size	Decay	Accuracy
1	0e+00	0.7427479
1	1e-04	0.7427479
1	1e-01	0.7510388
3	0e+00	0.7427479
3	1e-04	0.7420153
3	1e-01	0.7421220
5	0e+00	0.7419803
5	1e-04	0.7412827
5	**1e-01**	**0.7583633**

Confusion Matrix and Statistics for the optimal model from training

Prediction	no	yes
no	227	36
yes	11	27

Accuracy : 0.8439
95% CI : (0.7978, 0.883)

No Information Rate : 0.7907
P-Value [Acc > NIR] : 0.0120165
Kappa : 0.4477
Mcnemar's Test P-Value : 0.0004639
Sensitivity : 0.9538
Specificity : 0.4286
Pos Pred Value : 0.8631
Neg Pred Value : 0.7105
Prevalence : 0.7907
Detection Rate : 0.7542
Detection Prevalence : 0.8738
Balanced Accuracy : 0.6912

'Positive' Class : no

K Nearest Neighbor

Determining optimal hyperparameters based on training data

Kmax	Accuracy	Kernal	Distance
5	0.7091477	2	Optimal
7	0.7248790	2	Optimal
9	**0.7427963**	**2**	**Optimal**

Confusion Matrix and Statistics for the optimal model from training

Prediction	no	yes
no	222	48
yes	16	15

Accuracy : 0.7874
95% CI : (0.7368, 0.8322)

No Information Rate : 0.7907
P-Value [Acc > NIR] : 0.5892345
Kappa : 0.2101
Mcnemar's Test P-Value : 0.0001066
Sensitivity : 0.9328

Specificity : 0.2381
Pos Pred Value : 0.8222
Neg Pred Value : 0.4839
Prevalence : 0.7907
Detection Rate : 0.7375
Detection Prevalence : 0.8970
Balanced Accuracy : 0.5854

'Positive' Class : no

Regression Logistic

Determining optimal hyperparameters based on training data

cost	loss	epsilon	Accuracy	Kappa
0.5	L1	0.001	0.750993	0.109022355
0.5	L1	0.01	0.7494694	0.096347842
0.5	L1	0.1	0.7487132	0.064136569
0.5	L2_dual	0.001	0.6999829	0.001107029
0.5	L2_dual	0.01	0.7028646	0.033759731
0.5	L2_dual	0.1	0.7094553	0.013227233
0.5	L2_primal	0.001	0.7532484	0.131370898
0.5	**L2_primal**	**0.01**	**0.7576898**	**0.158094263**
0.5	L2_primal	0.1	0.7427479	0
1	L1	0.001	0.751737	0.107938512
1	L1	0.01	0.7457374	0.074801766
1	L1	0.1	0.748725	0.069944963
1	L2_dual	0.001	0.6587796	0.03779757
1	L2_dual	0.01	0.7228075	0.011964286
1	L2_dual	0.1	0.6931028	0.039073918
1	L2_primal	0.001	0.7510162	0.124046568
1	L2_primal	0.01	0.7576898	0.158094263
1	L2_primal	0.1	0.7427479	0
2	L1	0.001	0.7509812	0.104204366
2	L1	0.01	0.7479695	0.086553185
2	L1	0.1	0.7472487	0.066395644

2	L2_dual	0.001	0.6928562	0.039613156
2	L2_dual	0.01	0.6903081	0.047426146
2	L2_dual	0.1	0.6384219	0.025061318
2	L2_primal	0.001	0.7510162	0.124046568
2	L2_primal	0.01	0.7576898	0.158094263
2	L2_primal	0.1	0.7427479	0

Confusion Matrix and Statistics for the optimal model from training

```
          Reference
Prediction  no yes
       no  229  46
       yes   9  17

               Accuracy : 0.8173
                 95% CI : (0.7689, 0.8593)

    No Information Rate : 0.7907
    P-Value [Acc > NIR] : 0.1435

                  Kappa : 0.2959

 Mcnemar's Test P-Value : 1.208e-06

            Sensitivity : 0.9622
            Specificity : 0.2698
         Pos Pred Value : 0.8327
         Neg Pred Value : 0.6538
             Prevalence : 0.7907
         Detection Rate : 0.7608
   Detection Prevalence : 0.9136
      Balanced Accuracy : 0.6160

       'Positive' Class : no
```

Naive Bayes

Determining optimal hyperparameters based on training data

usekernel	Accuracy	Kappa	laplace	Adjust
FALSE	0.7345284	0.1048408	0	1
TRUE	0.7278881	0.2641497	0	1

Confusion Matrix and Statistics for the optimal model from training

Prediction no yes
no 226 48
yes 12 15

Accuracy : 0.8007
95% CI : (0.751, 0.8443)

No Information Rate : 0.7907
P-Value [Acc > NIR] : 0.3662
Kappa : 0.2376
Mcnemar's Test P-Value : 6.228e-06
Sensitivity : 0.9496
Specificity : 0.2381
Pos Pred Value : 0.8248
Neg Pred Value : 0.5556
Prevalence : 0.7907
Detection Rate : 0.7508
Detection Prevalence : 0.9103
Balanced Accuracy : 0.5938

'Positive' Class : no

J48

Determining optimal hyperparameters based on training data

C	M	Accuracy	Kappa
0.01	1	0.7427479	0
0.01	2	0.7427479	0
0.01	3	0.7427479	0
0.255	1	0.766594	0.3093979
0.255	2	0.7651173	0.3064725
0.255	**3**	**0.7666058**	**0.3093762**
0.5	1	0.7635706	0.3069954
0.5	2	0.7620939	0.307336
0.5	3	0.7635824	0.3053591

Confusion Matrix and Statistics for the optimal model from training

```
Prediction  no yes
       no  221  39
       yes  17  24

               Accuracy : 0.814
                 95% CI : (0.7653, 0.8563)

    No Information Rate : 0.7907
    P-Value [Acc > NIR] : 0.179046

                  Kappa : 0.3551
 Mcnemar's Test P-Value : 0.005012
            Sensitivity : 0.9286
            Specificity : 0.3810
         Pos Pred Value : 0.8500
         Neg Pred Value : 0.5854
             Prevalence : 0.7907
         Detection Rate : 0.7342
   Detection Prevalence : 0.8638
      Balanced Accuracy : 0.6548

       'Positive' Class : no
```

Unweighted Voting Ensemble Confusion matrix and accuracy

Confusion Matrix and Statistics

```
Prediction  no yes
       no  230  42
       yes   8  21

               Accuracy : 0.8339
                 95% CI : (0.7869, 0.8741)

    No Information Rate : 0.7907
    P-Value [Acc > NIR] : 0.03561

                  Kappa : 0.3739
 Mcnemar's Test P-Value : 3.058e-06
            Sensitivity : 0.9664
            Specificity : 0.3333
         Pos Pred Value : 0.8456
         Neg Pred Value : 0.7241
             Prevalence : 0.7907
         Detection Rate : 0.7641
   Detection Prevalence : 0.9037
```

Balanced Accuracy : 0.6499

Weighted Voting Ensemble

J48 Weight	Naive Bayes Weight	Regression Logistic Weight	K Nearest Neighbor Weight	Neural Network Weight	Accuracy
1.25	1	1	.5	1.25	.8306
1	**1**	**.5**	**.5**	**2**	**.8472**
.5	.5	.5	.5	3	.8439
.5	.5	1.5	2	.5	.8173

Discussion

When determining the five base classifiers I used 4 models that we talked about in class and another model that seemed interesting. The model with the best performance on training data ended up being J48. I was quite surprised by this. I had fully expected the neural network to vastly outperform all other models. However, the neural network vastly outperformed all other networks on the testing data. Which was expected.

One surprising observation from these experiments is every network performed worse on the training data then on the testing data. I have never observed this before and I am curious as to what the reasoning could be. One guess would there was mislabeled data in the training data, however, this is just a guess.

In regards to the unweighted network, It performed worse than just the neural network. This was pretty surprising however it does make sense as some of the other models performed poorly on the test data compared to the neural network.

When creating the weighted voting ensemble model I experimented with manually setting the weights semi proportionately with the accuracy of the performance. However, when giving the neural network a very high weight of 3 it performed slightly worse than when the weight was 2. This is a great example of the power of the ensemble model and how many of these networks used together far exceed one when weighted correctly.

Task 3

Unweighted Voting Ensemble Confusion Matrix and Statistics with RF and AdaBoost.M1 Added

```
Prediction  no yes
       no  224  39
       yes  14  24

               Accuracy : 0.8239
                 95% CI : (0.7761, 0.8652)

    No Information Rate : 0.7907
    P-Value [Acc > NIR] : 0.0871624

                  Kappa : 0.3772
 Mcnemar's Test P-Value : 0.0009784

            Sensitivity : 0.9412
            Specificity : 0.3810
         Pos Pred Value : 0.8517
         Neg Pred Value : 0.6316
             Prevalence : 0.7907
         Detection Rate : 0.7442
   Detection Prevalence : 0.8738
      Balanced Accuracy : 0.6611
```

Weighted Voting Ensemble

J48 Weight	Naive Bayes Weight	Regression Logistic Weight	K Nearest Neighbor Weight	Neural Network Weight	Adaboost. M1 weight	Random Forest Weight	Accuracy
1.25	1	1.5	.5	1.25	.75	.75	.8206
1.5	.5	.5	.5	2	1.5	.5	.8239
.5	.5	.5	.5	3	1	1	.8339
2	**.5**	**1.5**	**.5**	**2**	**.25**	**.25**	**.8439**

Discussion

In this experiment I added the Adaboost.M1 and Random Forest to the ensemble. Then I did experiments with weighted and unweighted voting. Both of these resulted in lower accuracy then the Ensemble with the five models. This was expected as the Random Forest and AdaBoost.M1 models both had quite poor performance compared to the other models in the ensemble. The best performance was also found when the two new models had the lowest weights.

These experiments were also similar to the weighted voting in task 2 for the results of giving the neural network a large weight. A large weight on the neural network results in a higher accuracy for all of these experiments.