Karthikeyan Chellamuthu_10.2.2

Karthikeyan Chellamuthu

2/19/2022

Fit a logistic regression model to the binary-classifier-data.csv dataset from the previous assignment.

10.2.2.a What is the accuracy of the logistic regression classifier?

```
glm1 <- glm(label ~ x + y,family = binomial(),data = training)

tidy(glm1)
```

```
## # A tibble: 3 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)  0.250     0.151       1.65  0.0990
## 2 x           -0.00108   0.00232    -0.465 0.642
## 3 y           -0.00702   0.00242    -2.90  0.00368
```

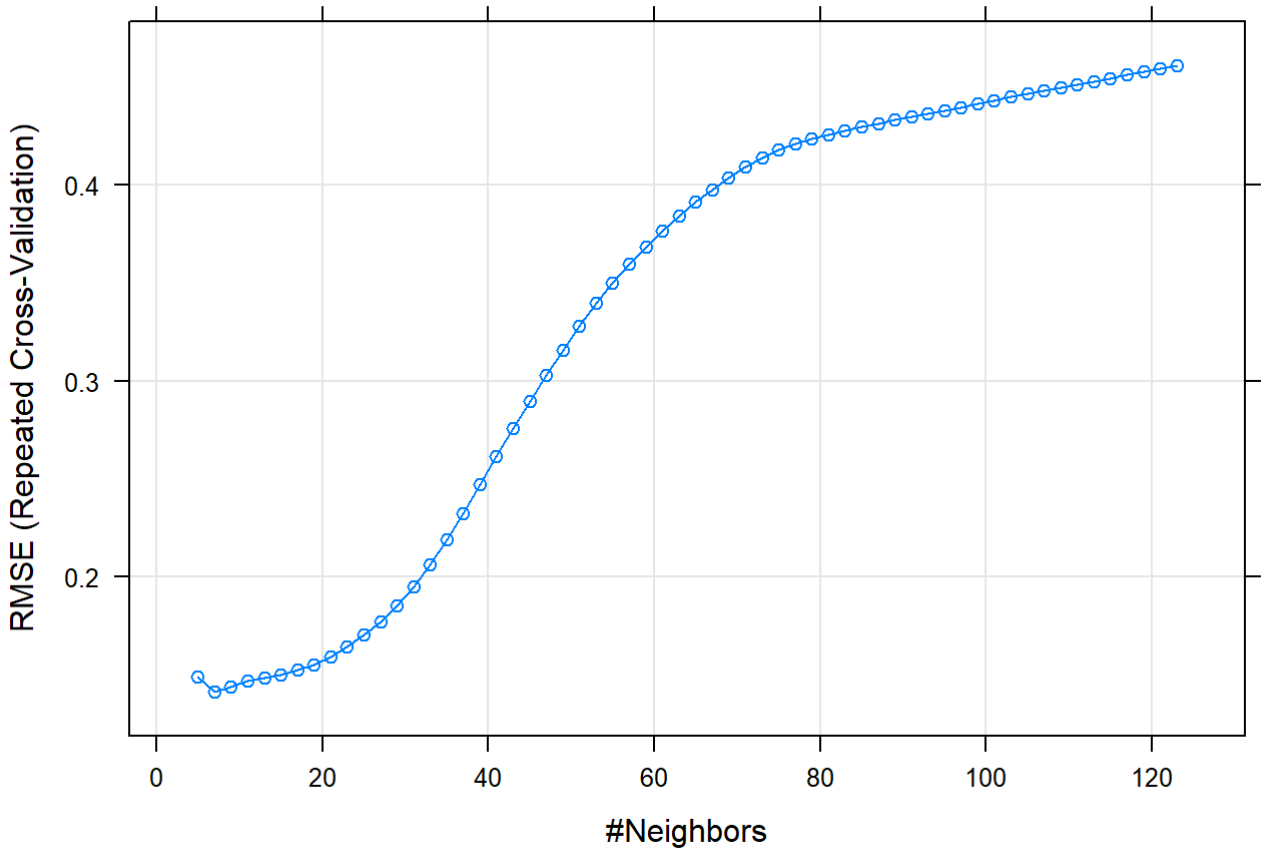
Accuracy of the model : **54.42%**

10.2.2.b How does the accuracy of the logistic regression classifier compare to the nearest neighbors algorithm?

```
## Warning in train.default(x, y, weights = w, ...): You are trying to do
## regression and your outcome only has two possible values Are you trying to do
## classification? If so, use a 2 level factor as your outcome column.
```

```
## k-Nearest Neighbors
##
## 899 samples
##   2 predictor
##
## Pre-processing: centered (2), scaled (2)
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 809, 809, 809, 809, 809, 809, ...
## Resampling results across tuning parameters:
##
##   k    RMSE       Rsquared   MAE
##     5  0.1490352  0.9066327  0.04181190
##     7  0.1413331  0.9155302  0.04254129
##     9  0.1438473  0.9132437  0.04618729
##    11  0.1470662  0.9094873  0.04985053
##    13  0.1484326  0.9084050  0.05283202
##    15  0.1502995  0.9064725  0.05662391
##    17  0.1529360  0.9038774  0.06068321
##    19  0.1551890  0.9018578  0.06459020
##    21  0.1596203  0.8971319  0.07038776
##    23  0.1643796  0.8921291  0.07725586
##    25  0.1704517  0.8857859  0.08585179
##    27  0.1773667  0.8785970  0.09571682
##    29  0.1853221  0.8705393  0.10716067
##    31  0.1953247  0.8602106  0.12080229
##    33  0.2064542  0.8489358  0.13624728
##    35  0.2190129  0.8347045  0.15262669
##    37  0.2326253  0.8175484  0.16963764
##    39  0.2471317  0.7961319  0.18678822
##    41  0.2616127  0.7722075  0.20332677
##    43  0.2757641  0.7461797  0.21915795
##    45  0.2894428  0.7186406  0.23436404
##    47  0.3027246  0.6893679  0.24907347
##    49  0.3156617  0.6584765  0.26343179
##    51  0.3279458  0.6267796  0.27698425
##    53  0.3394216  0.5952754  0.28965634
##    55  0.3498985  0.5649165  0.30114058
##    57  0.3593993  0.5361069  0.31162632
##    59  0.3682934  0.5079594  0.32142584
##    61  0.3764931  0.4804730  0.33036890
##    63  0.3840116  0.4538926  0.33844664
##    65  0.3909947  0.4282065  0.34590866
##    67  0.3974651  0.4040716  0.35291838
##    69  0.4035951  0.3808684  0.35966583
##    71  0.4089987  0.3601071  0.36564904
##    73  0.4137700  0.3420291  0.37111900
##    75  0.4175845  0.3277436  0.37575316
##    77  0.4208431  0.3157945  0.37992736
##    79  0.4233956  0.3063874  0.38327945
##    81  0.4255466  0.2984481  0.38622836
##    83  0.4275346  0.2909045  0.38891458
##    85  0.4294206  0.2836522  0.39148454
##    87  0.4312952  0.2764267  0.39406884
##    89  0.4329054  0.2703864  0.39637688
##    91  0.4345496  0.2638705  0.39864281
##    93  0.4361678  0.2575341  0.40091310
```

```
##    95  0.4377555  0.2512930  0.40316293
##    97  0.4394281  0.2444477  0.40548068
##    99  0.4411376  0.2375216  0.40786156
##   101  0.4429041  0.2304148  0.41030437
##   103  0.4446785  0.2232254  0.41273671
##   105  0.4464227  0.2162365  0.41514343
##   107  0.4480013  0.2099897  0.41740494
##   109  0.4495682  0.2039087  0.41966500
##   111  0.4512065  0.1974581  0.42195464
##   113  0.4527519  0.1915770  0.42416865
##   115  0.4542342  0.1857998  0.42628364
##   117  0.4558855  0.1793150  0.42851483
##   119  0.4574736  0.1731450  0.43069459
##   121  0.4591389  0.1666727  0.43291210
##   123  0.4608606  0.1599397  0.43517252
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 7.
```




```
kNNM <- knn(train = training, test = test, cl = training$label, k = 7)
confusionMatrix(table(kNNM, test$label))
```

```
## Confusion Matrix and Statistics
##
##
## kNNM   0   1
##    0 281  10
##    1  11 297
##
##                Accuracy : 0.9649
##                  95% CI : (0.9469, 0.9782)
##     No Information Rate : 0.5125
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.9298
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9623
##             Specificity : 0.9674
##          Pos Pred Value : 0.9656
##          Neg Pred Value : 0.9643
##              Prevalence : 0.4875
##          Detection Rate : 0.4691
##    Detection Prevalence : 0.4858
##       Balanced Accuracy : 0.9649
##
##        'Positive' Class : 0
##
```

Accuracy of the kNN model : **97%**

10.2.2.c Why is the accuracy of the logistic regression classifier different from that of the nearest neighbors?

- KNN is lazy execution, which means it fits and predicts at the time of prediction. KNN is better than logistic regression when the data contains high SNR
- KNN is a completely non-parametric approach: No assumptions are made about the shape of the decision boundary.
- KNN supports non-linear solutions where LR supports only linear solutions.
- KNN does not tell us which predictors are important as we don’t get a table of coefficients with p-values.