P1-Boston-Data

Jeetendra Gan

11/4/2019

Following are the features in the boston data set. With 'crim' as the quantitative variable. I have replaced crim with TRUE/FALSE, TRUE if the crime rate is greater than the median value, else false.

```
##  [1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"      "age"
##  [8] "dis"     "rad"     "tax"     "ptratio" "black"   "lstat"   "medv"
```

The median for crime rate is:

```
## [1] 0.25651
```

Here is the statistic after crim is transformed as a categorical variable.

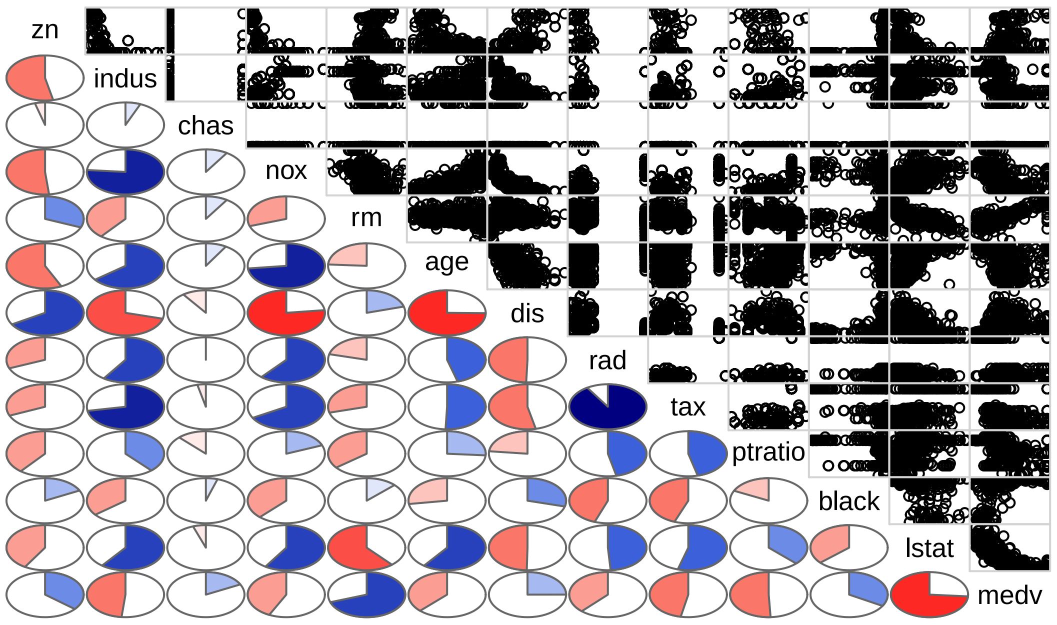
```
##    Mode   FALSE    TRUE
## logical     253     253
```

We have a perfectly balanced class set. 253 variables on either side. I will be comparing each of the algorithms on the basis of the accuracy, precision, and recall. As the classes are well balanced, we can hope to have a good comparision between algorithms on the basis of above written metrics.

Data Cleaning

None of the features has N.A. values, negative values. So, no pre-processing is required. In order to identify the correlation between the features, I have plotted, the correlation matrix.

Boston Correlation



Following are the observations with respect to correlation between variables

Positive correlation

1. Indus - Chas
2. Indus - Age
3. Indus - Rad
4. Indus - Tax
5. Indus - Lstat
6. Nox - Rad
7. Nox - Tax
8. Nox - Lstat
9. rm - medv
10. age - Lstat
11. rad - tax
12. zn - dist

Negative correlation

1. Zn - age
2. indus - dis
3. rm - lstat
4. age - dist
5. lstat - medv

Train-test division

I have selected 75% of the samples randomly as the training data, the rest will be the test data.

Our train and test data sets are also very well balanced, with train data set containing 190, and 189 TRUE, FALSE cases respectively, and the test set containing 63, and 64 TRUE, FALSE cases respectively.

Logistic Regression

Here is the summary of logistic regression on the training data

```
##
## Call:
## glm(formula = crim ~ ., family = binomial, data = trainData)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7109  -0.1721   0.0001   0.0054   3.5188
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -33.939299   7.488211  -4.532 5.83e-06 ***
## zn           -0.078757   0.038047  -2.070  0.03845 *
## indus        -0.070021   0.052616  -1.331  0.18326
## chas          0.401738   0.853233   0.471  0.63775
## nox          47.398517   8.643372   5.484 4.16e-08 ***
## rm           -0.331606   0.809108  -0.410  0.68192
## age           0.042799   0.015352   2.788  0.00531 **
## dis           0.746777   0.262032   2.850  0.00437 **
## rad           0.586171   0.179218   3.271  0.00107 **
## tax          -0.006918   0.003313  -2.088  0.03678 *
## ptratio       0.250405   0.141495   1.770  0.07678 .
## black        -0.006728   0.005234  -1.285  0.19864
## lstat        -0.011002   0.059629  -0.185  0.85362
## medv          0.120250   0.076044   1.581  0.11380
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 525.38  on 378  degrees of freedom
## Residual deviance: 159.39  on 365  degrees of freedom
## AIC: 187.39
##
## Number of Fisher Scoring iterations: 9
```

```
##
## glm.pred FALSE TRUE
##    FALSE    62    7
##    TRUE      3   55
```

The accuracy obtained is:

```
## [1] 0.9212598
```

The miss-classification percentage is:

```
## [1] 7.874016
```

The model precision is:

```
## [1] 0.9482759
```

The model recall is:

```
## [1] 0.8870968
```

The most significant features are nox, age, dis, rad.

I have fit fit a new model using these features.

```
##
## Call:
## glm(formula = crim ~ ., family = binomial, data = trainSubset1)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.00368  -0.32581   0.00089   0.01368   2.77463
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -20.85580    3.61357  -5.772 7.86e-09 ***
## nox          28.56868    5.70827   5.005 5.59e-07 ***
## age           0.02712    0.01048   2.588 0.009665 **
## dis           0.30173    0.16576   1.820 0.068706 .
## rad           0.43134    0.11647   3.703 0.000213 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 525.38  on 378  degrees of freedom
## Residual deviance: 191.49  on 374  degrees of freedom
## AIC: 201.49
##
## Number of Fisher Scoring iterations: 8
```

‘dis’ has lost a lot of its significance as can be seen by its increase in the p-value. After testing the model on the test data following results are obtained.

```
##
## glm.pred FALSE TRUE
##    FALSE    58   12
##    TRUE      7   50
```

The misclassification percentage is:

```
## [1] 14.96063
```

The model precision is:

```
## [1] 0.877193
```

The model recall is:

```
## [1] 0.8064516
```

The miss-classification is almost twice as much as it is with all the variables. Both false positives and false negatives have increased, i.e. both recall and precision have dropped.

We could also remove dis, and fit the model. Now we have “crim”, “nox”, “age”, “rad” in the feature set.

Here is the summary of the fit.

```
##
## Call:
## glm(formula = crim ~ ., family = binomial, data = trainSubset2)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.98043  -0.35075   0.00148   0.01995   2.66280
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -16.18450    2.25269  -7.185 6.74e-13 ***
## nox          22.37067    4.25921   5.252 1.50e-07 ***
## age           0.02337    0.01021   2.289 0.022053 *
## rad           0.43485    0.11557   3.763 0.000168 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 525.38  on 378  degrees of freedom
## Residual deviance: 194.77  on 375  degrees of freedom
## AIC: 202.77
##
## Number of Fisher Scoring iterations: 8
```

```
##
## glm.pred FALSE TRUE
##    FALSE    60   11
##    TRUE      5   51
```

The misclassification percentage is:

```
## [1] 12.59843
```

The model precision is:

```
## [1] 0.9107143
```

The model recall is:

```
## [1] 0.8225806
```

The model without dis is better than the one without it in all respects like the precision, recall, and accuracy(1 - Miss-classifications) but not better than all the features. But, it can be a better fit as it simpler than a model that has all the features.

Linear Discriminant Analysis

The Accuracy obtained using LDA is:

```
## [1] 0.8582677
```

The precision for LDA is:

```
## [1] 0.94
```

The recall for LDA is:

```
## [1] 0.7580645
```

The model above considers all the featueres. It also has been noted that for the same features, the results are significantly different for different models, which is expected.

Results for the subset - nox, age, dis, rad

```
##
##          FALSE TRUE
##   FALSE     63   17
##   TRUE       2   45
```

The accuracy is:

```
## [1] 0.8503937
```

Precision is:

```
## [1] 0.9574468
```

Recall is:

```
## [1] 0.7258065
```

It can be seen that the recall is the lowest. This model is not suitable for applications where it is safer to err on the side of caution due to low recall.

Results for the subset - nox, age, rad

```
##
##         FALSE TRUE
##   FALSE    63   17
##   TRUE      2   45
```

The accuracy is:

```
## [1] 0.8503937
```

Precision is:

```
## [1] 0.9574468
```

Recall is:

```
## [1] 0.7258065
```

Even after removing the dis feature, the results are exactly the same. This suggests that dis was insignificant in the context of the above 4 variables for LDA.

Knn classification

Here are the results of KNN classification for all features.

```
##          testY
## knn.pred FALSE TRUE
##    FALSE    64    9
##    TRUE      1   53
```

The accuracy obtained with knn with k = 3 is:

```
## [1] 0.9212598
```

The precision obtained for Knn is:

```
## [1] 0.9814815
```

Recall for knn is:

```
## [1] 0.8548387
```

For subset with “nox”, “age”, “rad”, “dis” features following results are obtained

```
##         testY
## knn.pred FALSE TRUE
##    FALSE    57   14
##    TRUE      8   48
```

Accuracy:

```
## [1] 0.8267717
```

Precision:

```
## [1] 0.8571429
```

Recall:

```
## [1] 0.7741935
```

For subset with "nox", "age", "rad" features following results are obtained

```
##         testY
## knn.pred FALSE TRUE
##    FALSE    54   12
##    TRUE     11   50
```

Accuracy:

```
## [1] 0.8188976
```

Precision:

```
## [1] 0.8196721
```

Recall:

```
## [1] 0.8064516
```

After dropping 'dis' there is a significant difference in each of the three parameters. But, in general the model will all the features preforms better for Knn.

```
##           Accuracy Precision    Recall
## Logistic 0.9212598 0.9482759 0.8870968
## LDA      0.8582677 0.9400000 0.7580645
## kNN      0.9212598 0.9814815 0.8548387
```

Logistic and Knn perform better than LDA on every parameter. KNN is more precise as compared to Logistic, whereas logistic has a better recall. The accuracy of Knn and logistic is almost the same. We can use KNN when we want proportion of positive identifications marked as actually correct to be greater. Logistic can be used when we want proportion of actual positives identified correctly to be greater.