Classification

Chris Talley CLT190005

2023-02-16

Linear Models

Linear Classification Models are designed to classify qualitative data. This is usually done on binary target variables, belonging to one class or another. The main two methods are Linear Regression and Naive Bayes. LR is computationally inexpensive and works well if classes are linearly separable. However, it is also more likely to overfit the data. On the other hand, Naive Bayes makes the assumption that all predictors are independent. This allows it to be easily understood and implemented. It works well with small data sets in high dimensions. As the data set grows larger, NB can be outperformed by other classifiers. The assumptions made may also limit the results.

Analyzing a Data Set

The following steps use this data from Kaggle.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.1     v purrr   1.0.1
## v tibble  3.1.8     v dplyr   1.1.0
## v tidyr   1.3.0     v stringr 1.5.0
## v readr   2.1.4     v forcats 1.0.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
df <- read_csv("predictive_maintenance.csv")
```

```
## Rows: 10000 Columns: 10
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): Product ID, Type, Failure Type
## dbl (7): UDI, Air temperature [K], Process temperature [K], Rotational speed...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
df$`Product ID` <- NULL
```

Divide into 80/20 train/test

```
set.seed(5)
idx <- sample(1:nrow(df), nrow(df)*0.8, replace=FALSE)
train <- df[idx,]
test <- df[-idx,]
```

Data Exploration

```r
str(train)
```

```
## tibble [8,000 x 9] (S3: tbl_df/tbl/data.frame)
##  $ UDI                    : num [1:8000] 2255 6859 1833 3797 7239 ...
##  $ Type                   : chr [1:8000] "L" "L" "M" "M" ...
##  $ Air temperature [K]    : num [1:8000] 299 301 298 302 300 ...
##  $ Process temperature [K]: num [1:8000] 308 311 307 311 310 ...
##  $ Rotational speed [rpm] : num [1:8000] 1325 1467 1330 1504 1697 ...
##  $ Torque [Nm]            : num [1:8000] 46.9 45.4 51.7 37.5 36 27.6 36.9 48.3 40 26 ...
##  $ Tool wear [min]        : num [1:8000] 28 36 183 42 175 47 28 25 158 200 ...
##  $ Target                 : num [1:8000] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Failure Type           : chr [1:8000] "No Failure" "No Failure" "No Failure" "No Failure" ...
```

```r
head(train)
```

```
## # A tibble: 6 x 9
##     UDI Type  Air temperature [~1 Proce~2 Rotat~3 Torqu~4 Tool ~5 Target Failu~6
##   <dbl> <chr>               <dbl>   <dbl>   <dbl>   <dbl>   <dbl>  <dbl> <chr>
## 1  2255 L                    299.    308.    1325    46.9      28      0 No Fai~
## 2  6859 L                    301     311.    1467    45.4      36      0 No Fai~
## 3  1833 M                    298.    307.    1330    51.7     183      0 No Fai~
## 4  3797 M                    302.    311.    1504    37.5      42      0 No Fai~
## 5  7239 L                    300.    310     1697    36       175      0 No Fai~
## 6  1527 L                    298.    309     1787    27.6      47      0 No Fai~
## # ... with abbreviated variable names 1: `Air temperature [K]`,
## #   2: `Process temperature [K]`, 3: `Rotational speed [rpm]`,
## #   4: `Torque [Nm]`, 5: `Tool wear [min]`, 6: `Failure Type`
```

```r
tail(train)
```

```
## # A tibble: 6 x 9
##     UDI Type  Air temperature [~1 Proce~2 Rotat~3 Torqu~4 Tool ~5 Target Failu~6
##   <dbl> <chr>               <dbl>   <dbl>   <dbl>   <dbl>   <dbl>  <dbl> <chr>
## 1  8545 H                    298.    309.    2011    21        59      0 No Fai~
## 2  8129 L                    300.    311.    1629    31.3      41      0 No Fai~
## 3  3384 H                    301.    311.    1674    33.6      49      0 No Fai~
## 4  5731 L                    302.    312.    1793    25.3     200      0 No Fai~
## 5  3407 L                    301.    310.    1470    44       113      0 No Fai~
## 6  6527 L                    301.    310.    1515    38.4      51      0 No Fai~
## # ... with abbreviated variable names 1: `Air temperature [K]`,
## #   2: `Process temperature [K]`, 3: `Rotational speed [rpm]`,
## #   4: `Torque [Nm]`, 5: `Tool wear [min]`, 6: `Failure Type`
```

```r
names(train)
```

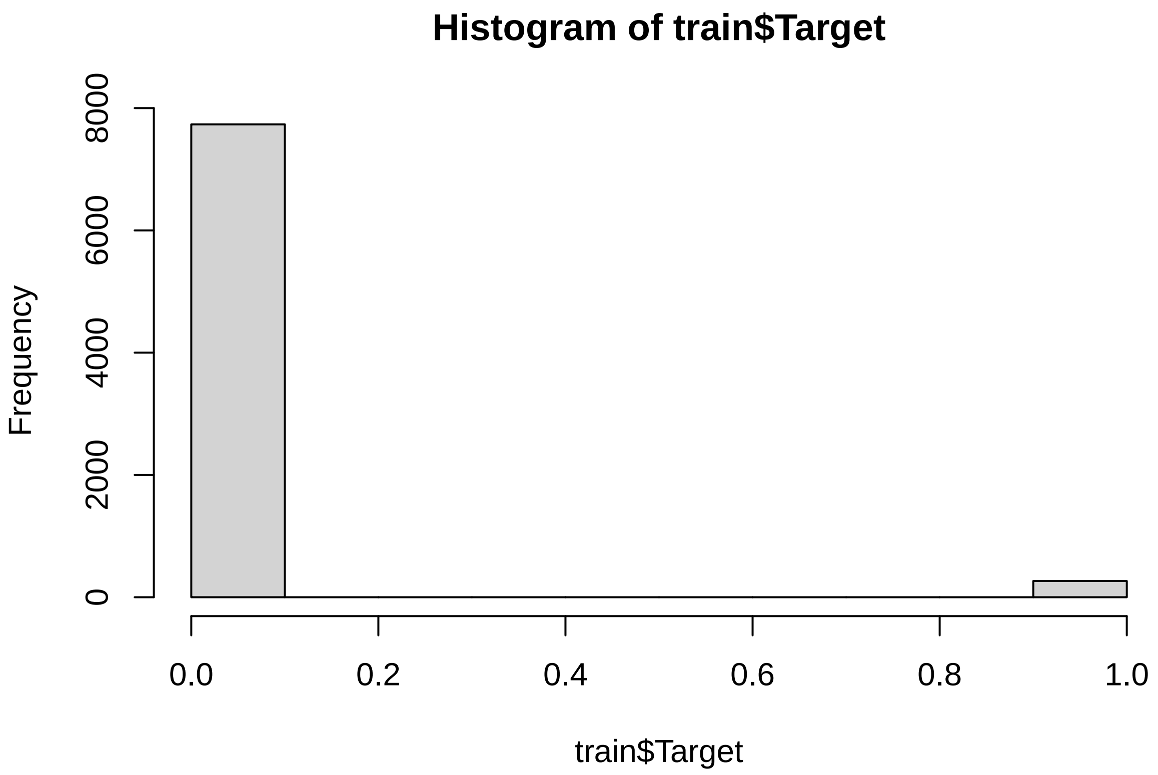
```
## [1] "UDI"                     "Type"
## [3] "Air temperature [K]"     "Process temperature [K]"
## [5] "Rotational speed [rpm]"  "Torque [Nm]"
## [7] "Tool wear [min]"         "Target"
## [9] "Failure Type"
```

```r
summary(train)
```

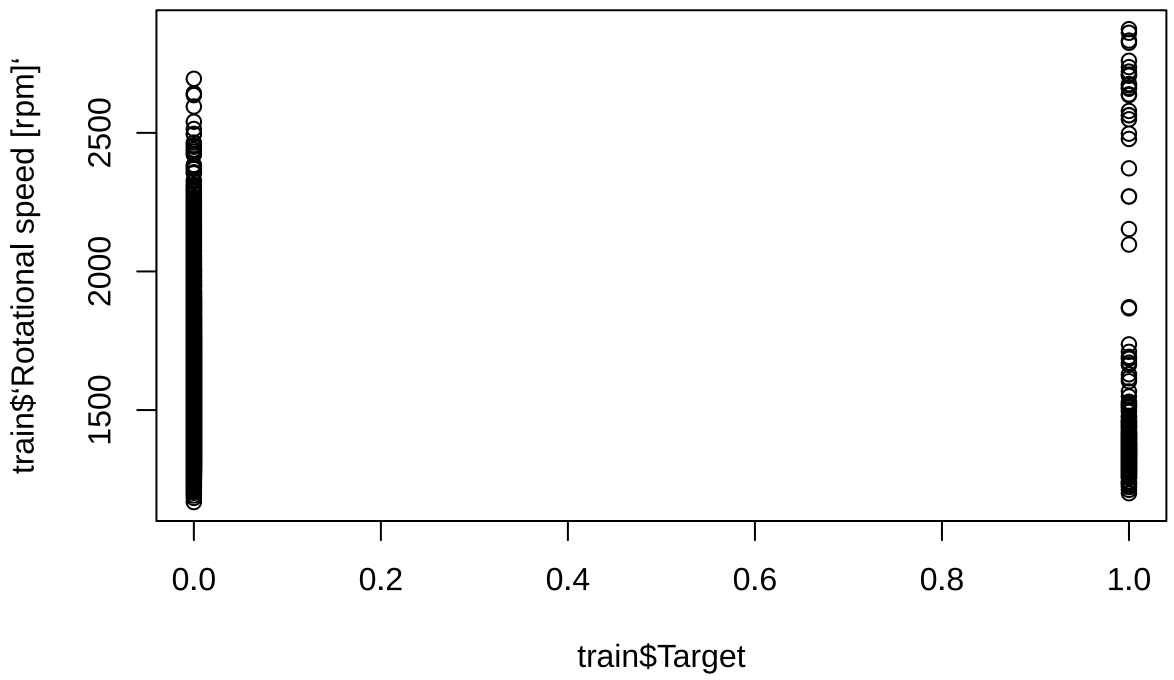
```
##       UDI             Type           Air temperature [K] Process temperature [K]
##  Min.   :   1   Length:8000        Min.   :295.3       Min.   :305.7
##  1st Qu.: 2498   Class :character   1st Qu.:298.3       1st Qu.:308.8
##  Median : 5008   Mode  :character   Median :300.1       Median :310.0
##  Mean   : 5004                   Mean   :300.0      Mean   :310.0
##  3rd Qu.: 7502                   3rd Qu.:301.5      3rd Qu.:311.0
##  Max.   :10000                   Max.   :304.5      Max.   :313.8
##  Rotational speed [rpm]  Torque [Nm]    Tool wear [min]     Target
##  Min.   :1168          Min.   : 4.20   Min.   :  0.0   Min.   :0.00000
##  1st Qu.:1423          1st Qu.:33.20   1st Qu.: 53.0   1st Qu.:0.00000
##  Median :1504          Median :40.10   Median :107.0   Median :0.00000
##  Mean   :1539          Mean   :40.01   Mean   :107.8   Mean   :0.03313
##  3rd Qu.:1611          3rd Qu.:46.83   3rd Qu.:162.0   3rd Qu.:0.00000
##  Max.   :2874          Max.   :76.60   Max.   :253.0   Max.   :1.00000
##  Failure Type
##  Length:8000
##  Class :character
##  Mode  :character
##
##
##
```

Creating Informative Graphs

```
hist(train$Target)
```



```
plot(train$Target, train$`Rotational speed [rpm]`)
```



Logistic Regression Model

```r
glm1 <- glm(Target~`Rotational speed [rpm]`, data=train, family=binomial)
summary(glm1)
```

```
##
## Call:
## glm(formula = Target ~ `Rotational speed [rpm]`, family = binomial,
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.3455  -0.2786  -0.2607  -0.2367   3.3617
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -0.8318155  0.6582752  -1.264 0.206363
## `Rotational speed [rpm]` -0.0016754  0.0004373  -3.832 0.000127 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2327.1  on 7999  degrees of freedom
## Residual deviance: 2310.2  on 7998  degrees of freedom
## AIC: 2314.2
##
## Number of Fisher Scoring iterations: 6
```

The summary shows the results of our classification model. As indicated by the asterisks, rotational speed is a good predictor for failure. The std. error is extremely low, meaning our predictions may turn out more accurate.

Naive Bayes Model

```r
library(e1071)
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
nb1 <- naiveBayes(Target~., data=train)
nb1
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##        0        1
## 0.966875 0.033125
##
## Conditional probabilities:
##    UDI
## Y       [,1]     [,2]
##   0 5019.742 2908.667
##   1 4545.313 2372.350
##
##    Type
## Y           H         L         M
##   0 0.1016160 0.5959922 0.3023917
##   1 0.0754717 0.6867925 0.2377358
##
##    Air temperature [K]
## Y       [,1]     [,2]
##   0 299.9604 1.998852
##   1 300.8823 2.067137
##
##    Process temperature [K]
## Y       [,1]     [,2]
##   0 309.9836 1.491065
##   1 310.2981 1.360772
##
##    Rotational speed [rpm]
## Y       [,1]     [,2]
##   0 1540.281 168.1511
##   1 1497.596 387.2886
##
##    Torque [Nm]
## Y       [,1]      [,2]
##   0 39.65811  9.525761
##   1 50.35698 16.437889
```

```
##
##    Tool wear [min]
## Y       [,1]     [,2]
##   0 106.4237 62.84126
##   1 146.5887 72.12164
##
##    Failure Type
## Y   Heat Dissipation Failure  No Failure Overstrain Failure Power Failure
##   0              0.000000000 0.998190045        0.000000000   0.000000000
##   1              0.316981132 0.026415094        0.233962264   0.294339623
##    Failure Type
## Y   Random Failures Tool Wear Failure
##   0     0.001809955       0.000000000
##   1     0.000000000       0.128301887
```

The model output shows an extremely high model accuracy. The A-priori probabilities are used independently with each predictor to generate this model. The most impactful predictor from this summary seems to be rotational speed. Other predictors such as both temperature measurements are also useful. The naive assumption made in this model could be responsible for the little error.

Predictions and Evaluations Using Classification Metrics

```r
probs1 <- predict(glm1, newdata=test, type="response")
pred1 <- ifelse(probs1>0.5, 1, 0)
acc1 <- mean(pred1==as.integer(test$Target))
print(paste("glm1 accuracy = ", acc1))
```

```
## [1] "glm1 accuracy =  0.963"
```

```r
table(pred1, as.integer(test$Target))
```

```
##
## pred1    0    1
##     0 1926   74
```

```r
pred2 <- predict(nb1, newdata=test, type="class")
confusionMatrix(pred2, as.factor(test$Target), positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1919    4
##          1    7   70
##
##                Accuracy : 0.9945
##                  95% CI : (0.9902, 0.9973)
##     No Information Rate : 0.963
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9243
##
##  Mcnemar's Test P-Value : 0.5465
##
##             Sensitivity : 0.9459
##             Specificity : 0.9964
##          Pos Pred Value : 0.9091
##          Neg Pred Value : 0.9979
##              Prevalence : 0.0370
##          Detection Rate : 0.0350
##    Detection Prevalence : 0.0385
##       Balanced Accuracy : 0.9712
##
##        'Positive' Class : 1
##
```

The results for both models are very good. The logistic model had an accuracy of 96.3%, while the Naive Bayes model had 99.5%. The increase in accuracy could be chalked up to the multiple predictors tracked in NB. The Kappa value, 92.4%, shows that when taken raw the odds are still quite high. The logistic model can be improved with the knowledge gained from the confusion matrix. The most prominent predictors can be used in a multiple logistic regression model to improve accuracy closer to the NB.

The main two methods are Linear Regression and Naive Bayes. LR is computationally inexpensive and works well if classes are linearly separable. However, it is also more likely to overfit the data. On the other hand, Naive Bayes makes the assumption that all predictors are independent. This allows it to be easily understood and implemented. It works well with small data sets in high dimensions. As the data set grows larger, NB can be outperformed by other classifiers. The assumptions made may also limit the results.

Classification metrics used are as follows: accuracy, kappa, roc/auc, and log odds. Accuracy determines how closely predictions match expected test values. Sensitivity and specificity measure true positive and negative rates respectively. Kappa is the accuracy adjusted for guesses. This can show the agreement between two annotators. ROC and AUC are used to visualize the performance of machine learning algorithms. This is used to determine the relationship between false positives and true positives. Log odds is used to determine the change in Y for 1 unit change in X. This is closely related to probability, or the raw chance.