Classification

Jonathan Ho

September 24, 2022

Logistical Regression

Logical regression is a way to predict a qualitative outcome. Similar to linear regression, it observes the relationship between the target and the predictors. However, the target would usually be a binary outcome where the target could be classified as one class or another. When all the target values are viewed, it would be possible to see a decision boundary. The decision boundary is what separates the data from one class with another. Logical regression can separate classes well with a linear line, not resource intensive, and the output looks generally nice in probabilities. The main drawback of logical regression however is underfitting can occur, especially if boundaries are complex and non-linear.

Loading in the data

Loading in the csv file. Attach is used so the $ notation is not needed.

```
df <- read.csv("toy_dataset.csv")
str(df)
```

```
## 'data.frame':    150000 obs. of  6 variables:
##  $ Number : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ City   : chr  "Dallas" "Dallas" "Dallas" "Dallas" ...
##  $ Gender : chr  "Male" "Male" "Male" "Male" ...
##  $ Age    : int  41 54 42 40 46 36 32 39 51 30 ...
##  $ Income : num  40367 45084 52483 40941 50289 ...
##  $ Illness: chr  "No" "No" "No" "No" ...
```

Illness need to be factored, so data cleaning is carried out.

```
df$Illness <- as.factor(df$Illness)
str(df)
```

```
## 'data.frame':    150000 obs. of  6 variables:
##  $ Number : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ City   : chr  "Dallas" "Dallas" "Dallas" "Dallas" ...
##  $ Gender : chr  "Male" "Male" "Male" "Male" ...
##  $ Age    : int  41 54 42 40 46 36 32 39 51 30 ...
##  $ Income : num  40367 45084 52483 40941 50289 ...
##  $ Illness: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
```

Dividing up the data

The data was split into 80/20 train/test.

```
set.seed(1234)
y <- sample(1:nrow(df), nrow(df)*0.8, replace=FALSE)
train <- df[y,]
test <- df[-y,]
```

Data exploration

1. See all the columns and what data type they are.

```
str(train)
```

```
## 'data.frame':    120000 obs. of  6 variables:
##  $ Number : int  106390 41964 146313 33702 126055 120410 83023 80756 85374 59944 ...
##  $ City   : chr  "Mountain View" "New York City" "Austin" "New York City" ...
##  $ Gender : chr  "Male" "Female" "Male" "Male" ...
##  $ Age    : int  62 53 42 35 29 28 62 61 29 61 ...
##  $ Income : num  147334 96932 92208 108560 76279 ...
##  $ Illness: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 ...
```

2. First and last six rows of each column.

```
head(train)
```

	Number <int>	City <chr>	Gender <chr>	Age <int>	Income <dbl>	Illness <fctr>
106390	106390	Mountain View	Male	62	147334	No
41964	41964	New York City	Female	53	96932	No
146313	146313	Austin	Male	42	92208	No
33702	33702	New York City	Male	35	108560	No
126055	126055	Washington D.C.	Male	29	76279	No
120410	120410	Boston	Male	28	95974	No

6 rows

```
tail(train)
```

	Number <int>	City <chr>	Gender <chr>	Age <int>	Income <dbl>	Illness <fctr>
127904	127904	Washington D.C.	Male	51	59995	No
61257	61257	New York City	Male	49	96246	No
107040	107040	Mountain View	Female	37	131467	No
133730	133730	San Diego	Female	63	109276	No

	Number <int>	City <chr>	Gender <chr>	Age <int>	Income <dbl>	Illness <fctr>
135778	135778	San Diego	Female	26	86934	Yes
114843	114843	Mountain View	Male	43	129287	No

6 rows

3. Checking how many NAs there are.

```
sapply(train, function(z) sum(is.na(z)))
```

```
##  Number    City  Gender     Age  Income Illness
##       0       0       0       0       0       0
```

4. Display how many rows and columns there are in the data.

```
dim(train)
```

```
## [1] 120000      6
```

5. Summary of the statistics of each column.

```
summary(train)
```

```
##      Number           City              Gender              Age        
##  Min.   :     1   Length:120000      Length:120000      Min.   :25.00  
##  1st Qu.: 37455   Class :character   Class :character   1st Qu.:35.00  
##  Median : 75079   Mode  :character   Mode  :character   Median :45.00  
##  Mean   : 75019                                         Mean   :44.95  
##  3rd Qu.:112462                                         3rd Qu.:55.00  
##  Max.   :150000                                         Max.   :65.00  
##      Income       Illness      
##  Min.   :   584   No :110261  
##  1st Qu.: 80863   Yes:  9739  
##  Median : 93696               
##  Mean   : 91299               
##  3rd Qu.:104568               
##  Max.   :177157
```

6. See the number of males and females in Gender for the train and test.

```
table(train$Illness)
```

```
##
##     No    Yes
## 110261   9739
```

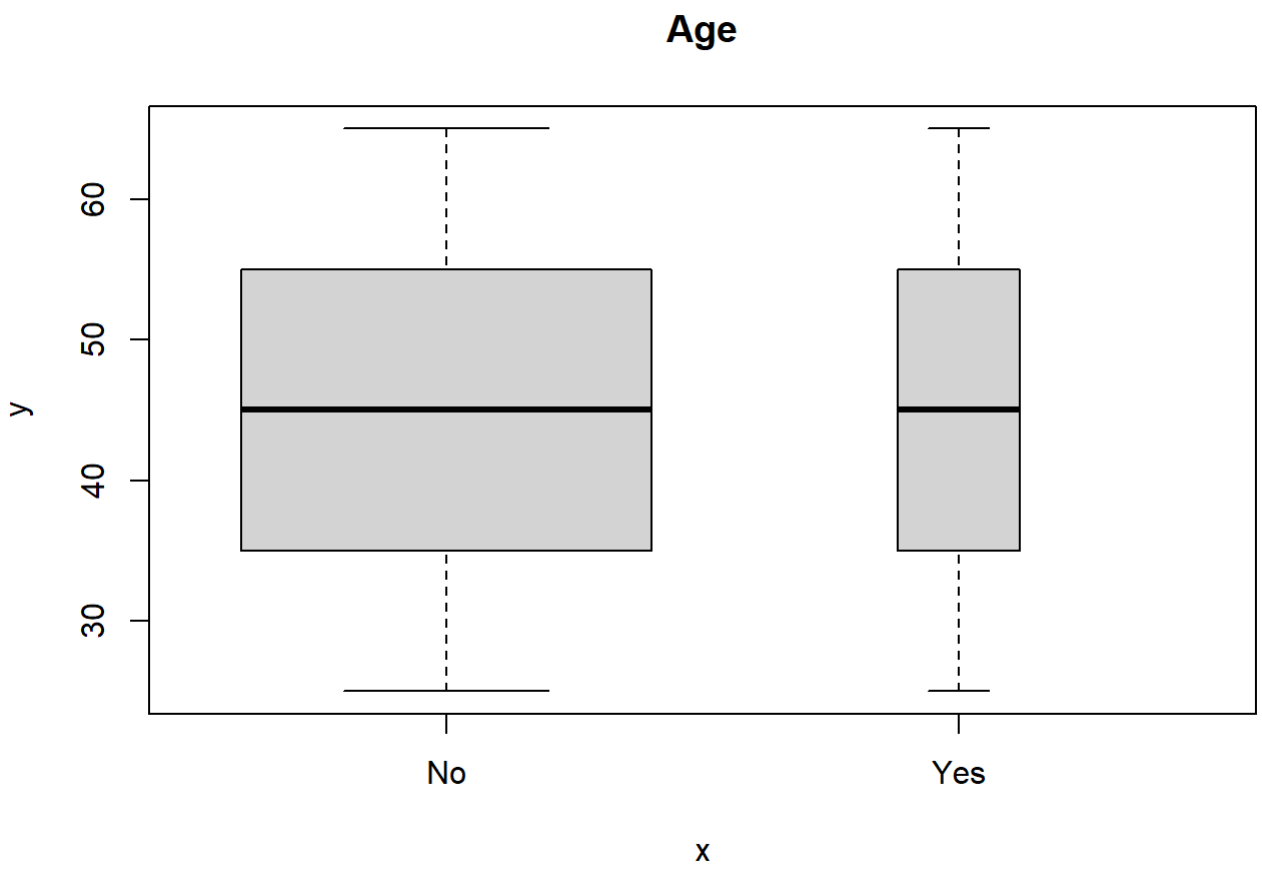
```
table(test$Illness)
```

```
##
##    No   Yes
## 27600  2400
```

Data visualization

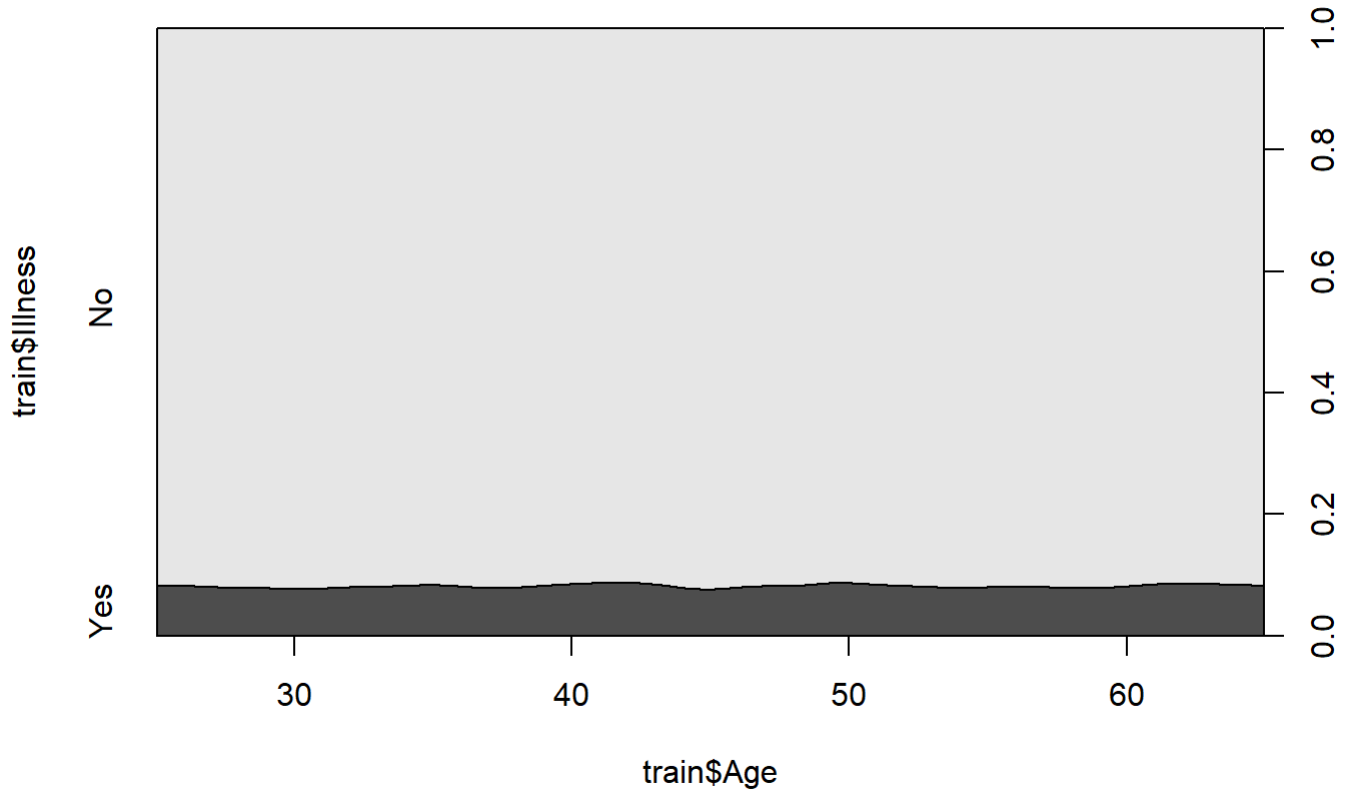
Box plot of age with respect to gender.

```
plot(train$Illness, train$Age, main="Age", varwidth=TRUE)
```



A conditional density plot is also created with age and illness.

```
cdplot(train$Illness~train$Age)
```



Logical Regression

Age is used as a predictor to predict the target gender.

```
glm1 <- glm(train$Illness~train$Age, data=train, family=binomial)
summary(glm1)
```

```
##
## Call:
## glm(formula = train$Illness ~ train$Age, family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.4148  -0.4129  -0.4111  -0.4093   2.2481
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.4649232  0.0424910 -58.011   <2e-16 ***
## train$Age    0.0008492  0.0009136   0.929    0.353
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 67581  on 119999  degrees of freedom
## Residual deviance: 67581  on 119998  degrees of freedom
## AIC: 67585
##
## Number of Fisher Scoring iterations: 5
```

Calling the summary() function on variable glm1 generates statistical data for the logical regression. Deviance residuals, coefficient, and more on deviance, Akaike Information Criterion (AIC), and the Fisher Scoring. The deviance residual display similar statistics like summary() in linear regression.

Deviance residuals are derived from transforming the loss function, which helps to quantify one data point's contribution to the likelihood. This would help with forming RSS-like statistics. Estimated coefficients are the same indicators as those used in linear regression.

Null deviance and residual deviance are similar but are on different scales. Null deviance only refers to the model not fitting well in respect to the intercept. Meanwhile, residual deviance is the model not fitting well in respect to everything. Ideally, having a significantly lower residual deviance than the Null deviance would be ideal, but it does not seem to be the case in this situation. AIC is another model indicator based on deviance. Lastly, there is the Fisher Scoring which indicates how many steps it needs to take to reach the max likelihood of outcomes.

Naïve Bayes Model

Given the training and test set from earlier, a Naïve Bayes Model can be built and an output can be shown.

```
library(e1071)
nb1 <- naiveBayes(train$Illness~train$Age, data=train)
nb1
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##         No        Yes
## 0.91884167 0.08115833
##
## Conditional probabilities:
##      train$Age
## Y         [,1]     [,2]
##   No  44.94237 11.57244
##   Yes 45.05606 11.55424
```

When the Naïve Bayes model was created, A-priori probabilities were given, which were the prior probabilities of Illness. This is saying that most people are not ill. Looking at the Age variable, it seems to be a continuous variable. Instead of probabilities being shown out of 1, it is giving the average age for not ill and ill. From the results, it seems that those around 45 and 11 are the ones to be ill and not ill.

Results

Evaluation on the logistic regression is carried out first.

```
probs <- predict(glm1, newdata=test, type="response")
```

```
## Warning: 'newdata' had 30000 rows but variables found have 120000 rows
```

```
pred <- ifelse(probs>0.5, 1, 0)
acc <- mean(pred==test$Illness)
print(paste("accuracy = ", acc))
```

```
## [1] "accuracy =  0"
```

Evaluation on the Naïve Bayes is done next.

```
p1 <- predict(nb1, newdata=test, type="class")
```

```
## Warning in predict.naiveBayes(nb1, newdata = test, type = "class"): Type
## mismatch between training and new data for variable 'train$Age'. Did you use
## factors with numeric labels for training, and numeric values for new data?
```

```
table(p1, test$Illness)
```

```
##
## p1       No   Yes
##   No  27600  2400
##   Yes     0     0
```

```
mean(p1==test$Illness)
```

```
## [1] 0.92
```

I believe there may have been some errors in the dataset when conducting logical regression and Naïve Bayes. Considering this is a toy dataset, I believe that the data should be significantly better. However, for some reason, the logistic regression evaulation ended up with an accuracy of 0. This is most likely due to amount mismatch. The Naïve Bayes however, actually gives a mean for those who are ill. However, it seems there are 27600 true positives and 2400 false positives, again I believe it potentially could be the dataset or how I carried out the Naïve Bayes.

Naïve Bayes and Logistic Regression Strengths and Weaknesses

Strength and weaknesses for logicstic regression were discussed in the beginning of this document. To reiterate, logical regression can separate classes well with a linear line, not resource intensive, and the output looks generally nice in probabilities. The main drawback of logical regression however is underfitting can occur, especially if boundaries are complex and non-linear. Naïve Bayes on the other hand has a couple of strengths as well as weaknesses. Naïve Bayes strengths include favoring small datasets, easy implementation and interpretation, and a high amount of dimensions can be handled well. However, when the datasets become large, it may not perform nearly as well as other classification methods, guesses are made in test data, and dependent predictors are assumed to limit the performance of the algorithm.

Classification Metrics

The classification metrics are accuracy, sensitivity, and specificity, Kappa, AUC, and ROC curves. Accuracy can tell how accurate a classification is, while the sensitivity and specificity tell true positives and true negatives respectfully. Kappa tries to adjust accuracy by accounting for a possibility of chance. The ROC curve mentions the visual performance of a machine learning algorighm, and the AUC, which is related to it, is the area under the ROC curve ranging from 0.5 (no predictive value) to 1.0 (perfect classifier).