Linear vs Bayesian Models for Classification

Linear Models for Classification

Using a linear model for classification is similar to using a linear model for regression. The difference is that predictions using regression involve estimating attributes of a new instance of data using the model and some given observations, whereas classification involves determining whether an instance belongs to one type or another. While some classification methods can handle multiple types of instances, linear classification is limited to binary classification, i.e., only two types can be classified using one model. Linear classification effectively forms a linear model of the data and then determines whether an instance is above or below a certain cut-off line, which is the line determined by the model.

Load Data

```
df <- read.csv('adult.data', header=FALSE)
df[,15] <-as.factor(ifelse (df[,15] == " >50K", 1, 0))
```

We used the UCI Adult Dataset and attempt to predict whether an individual makes more or less than $50K.

Divide Train and Test

```
set.seed(1234)
i <- sample(1:nrow(df), nrow(df)*0.80, replace=FALSE)
train <- df[i,]
test <- df[-i,]
```

Use five functions for data exploration.

```
str(df)
```

```
## 'data.frame':    32561 obs. of  15 variables:
##  $ V1 : int  39 50 38 53 28 37 49 52 31 42 ...
##  $ V2 : chr  " State-gov" " Self-emp-not-inc" " Private" " Private" ...
##  $ V3 : int  77516 83311 215646 234721 338409 284582 160187 209642 45781 159449 ...
##  $ V4 : chr  " Bachelors" " Bachelors" " HS-grad" " 11th" ...
##  $ V5 : int  13 13 9 7 13 14 5 9 14 13 ...
##  $ V6 : chr  " Never-married" " Married-civ-spouse" " Divorced" " Married-civ-spouse" ...
##  $ V7 : chr  " Adm-clerical" " Exec-managerial" " Handlers-cleaners" " Handlers-cleaners" ...
##  $ V8 : chr  " Not-in-family" " Husband" " Not-in-family" " Husband" ...
##  $ V9 : chr  " White" " White" " White" " Black" ...
##  $ V10: chr  " Male" " Male" " Male" " Male" ...
##  $ V11: int  2174 0 0 0 0 0 0 0 14084 5178 ...
##  $ V12: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V13: int  40 13 40 40 40 40 16 45 50 40 ...
##  $ V14: chr  " United-States" " United-States" " United-States" " United-States" ...
##  $ V15: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 2 2 ...
```

```
mean(df[,1])
```

```
## [1] 38.58165
```

```
min(df[,1])
```

```
## [1] 17
```

```
max(df[,1])
```

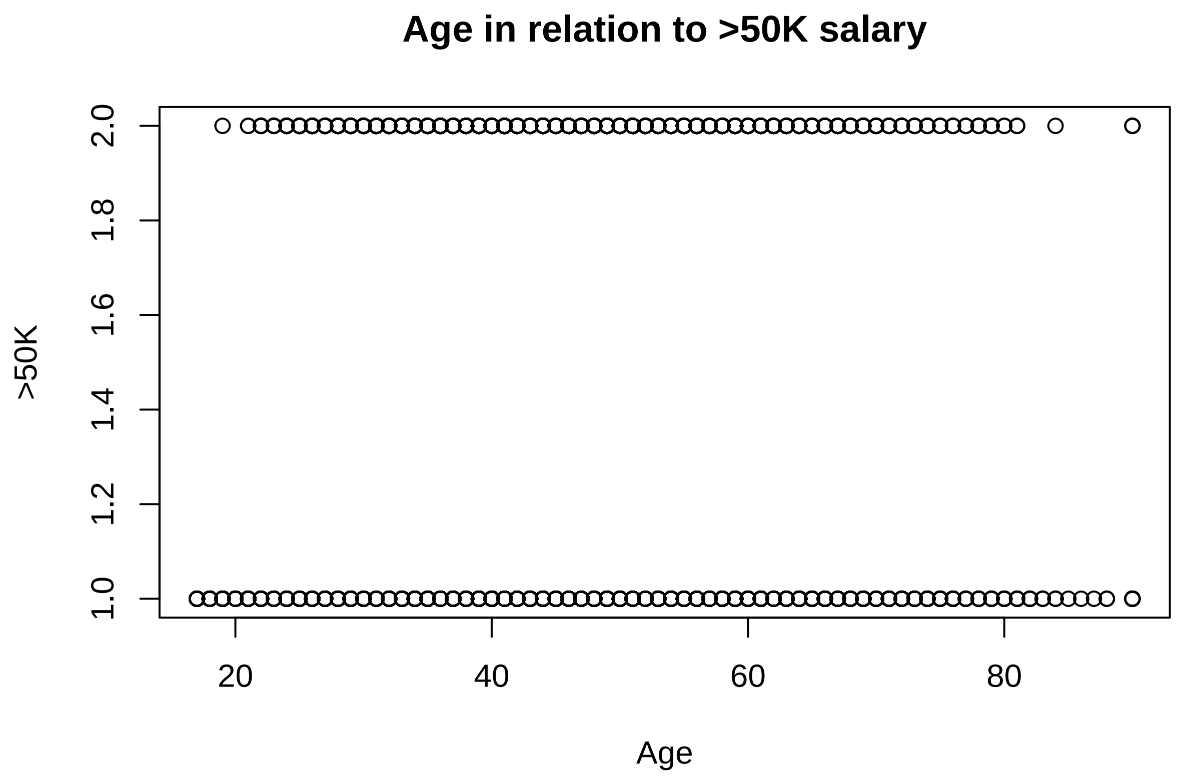
```
## [1] 90
```

```
sd(df[,1])
```

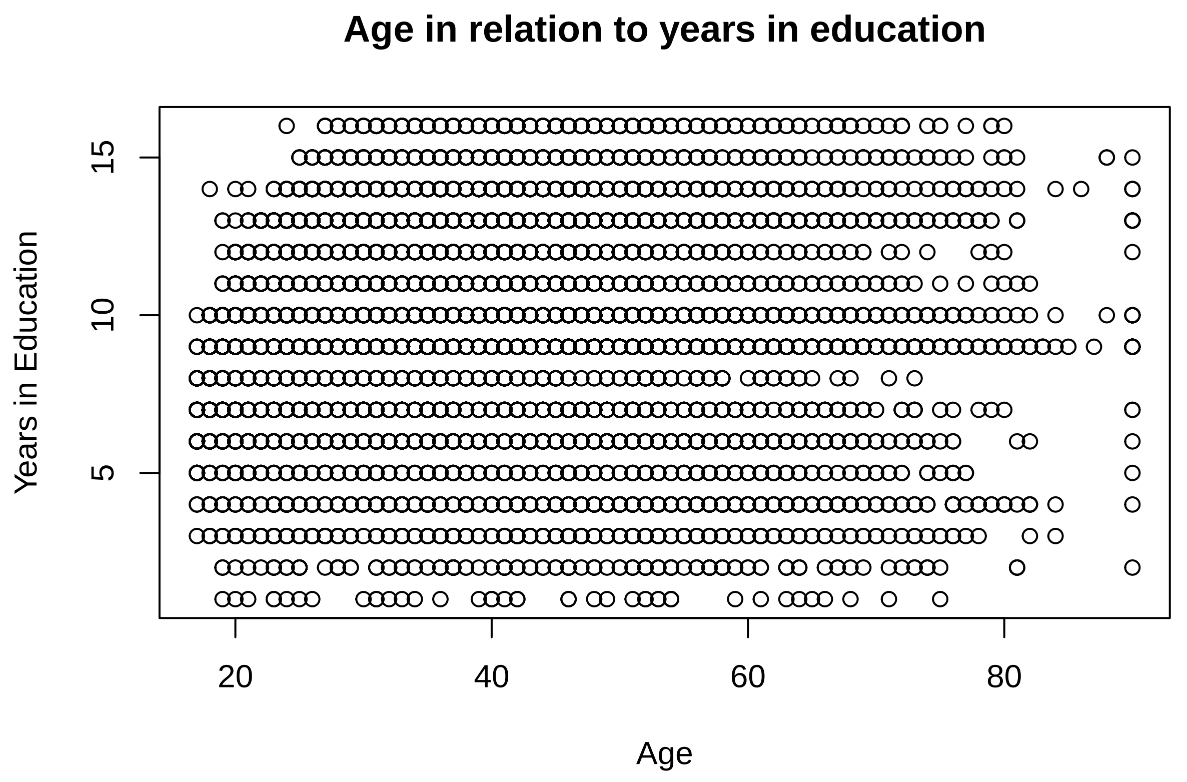
```
## [1] 13.64043
```

Make two informative graphs

```
plot(train[,1], train[,15], main="Age in relation to >50K salary", xlab="Age", ylab=">50K")
```



```
plot(train[,1], train[,5], main="Age in relation to years in education", xlab="Age", ylab="Years in Edu
```



Build a logistic regression model and output the summary

```
glm1 <- glm(train[,15]~train[,1]+train[,3]+train[,5]+train[,11]+train[,12]+train[,13], family = "binomi
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(glm1)
```

```
##
## Call:
## glm(formula = train[, 15] ~ train[, 1] + train[, 3] + train[,
##     5] + train[, 11] + train[, 12] + train[, 13], family = "binomial")
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -5.3250  -0.6370  -0.4066  -0.1319   2.8452
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept) -8.485e+00  1.356e-01 -62.569  < 2e-16 ***
## train[, 1]   4.306e-02  1.370e-03  31.438  < 2e-16 ***
## train[, 3]   6.393e-07  1.662e-07   3.845  0.00012 ***
## train[, 5]   3.231e-01  7.629e-03  42.348  < 2e-16 ***
## train[, 11]  3.187e-04  1.077e-05  29.598  < 2e-16 ***
## train[, 12]  7.154e-04  3.663e-05  19.530  < 2e-16 ***
## train[, 13]  4.150e-02  1.491e-03  27.832  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 28706  on 26047  degrees of freedom
## Residual deviance: 21112  on 26041  degrees of freedom
## AIC: 21126
##
## Number of Fisher Scoring iterations: 7
```

The coefficients displayed in the summary above are a numerical description of the relationship between the input variables and the resulting class as determined by the linear model. The significance codes seem to indicate that all predictors we used were beneficial, as '***' appears next to each of the coefficients. Deviance residuals should be small if the model has a good fit and larger if not. Null deviance indicates that the data can be well represented simply by the intercept of the line (in which case a model with many variables is unnecessary), and residual deviance is an approximate indicator of how well our current model fits the data (should be close to the degrees of freedom). AIC is yet another approximation of the quality of the model; lower is better. The number of Fisher Scoring iterations should be low, and increases if the model fails to converge.

Evaluate and predict on test data using the logistic regression model

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

```
## Warning: 'newdata' had 6513 rows but variables found have 26048 rows
```

```r
pred <- ifelse(probs>0.5, 1, 0)
acc <- mean(pred==test[,15])
```

```
## Warning in `==.default`(pred, test[, 15]): longer object length is not a
## multiple of shorter object length
```

```
## Warning in is.na(e1) | is.na(e2): longer object length is not a multiple of
## shorter object length
```

```r
print(paste("accuracy = ", acc))
```

```
## [1] "accuracy =  0.685465294840295"
```

Build a Naive Bayes model

```r
library(e1071)
nb1 <- naiveBayes(train[,15]~train[,1]+train[,3]+train[,5]+train[,11]+train[,12]+train[,13], data=train
nb1
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##         0         1
## 0.7600584 0.2399416
##
## Conditional probabilities:
##    train[, 1]
## Y       [,1]     [,2]
##   0 36.81897 14.08123
##   1 44.21488 10.41266
##
##    train[, 3]
## Y       [,1]     [,2]
##   0 189674.5 105991.1
##   1 187899.4 101477.8
##
##    train[, 5]
## Y        [,1]     [,2]
##   0  9.597939 2.452206
##   1 11.630400 2.359654
##
##    train[, 11]
## Y        [,1]       [,2]
##   0  149.6236   970.3204
##   1 4018.6848 14581.7440
##
##    train[, 12]
## Y        [,1]     [,2]
##   0  51.62001 307.1221
##   1 194.93104 594.0050
##
##    train[, 13]
## Y       [,1]     [,2]
##   0 38.84317 12.29426
##   1 45.44384 10.94613
```

The a-priori probabilities learned by the model indicate the likelihood of one instance belonging to either class without actually evaluating that specific instance. For example, this output show that the model would expect a person to have a 76% chance of making less than $50K without knowing anything about that person. The conditional probabilities are the actual weights given by the model to each attribute when determining an output.

Evaluate and predict on test data using Naive Bayes model

```r
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[, 1]'. Did you use
## factors with numeric labels for training, and numeric values for new data?

## Warning in predict.naiveBayes(nb1, newdata = test, type = "class"): Type
## mismatch between training and new data for variable 'train[, 3]'. Did you use
## factors with numeric labels for training, and numeric values for new data?

## Warning in predict.naiveBayes(nb1, newdata = test, type = "class"): Type
## mismatch between training and new data for variable 'train[, 5]'. Did you use
## factors with numeric labels for training, and numeric values for new data?

## Warning in predict.naiveBayes(nb1, newdata = test, type = "class"): Type
## mismatch between training and new data for variable 'train[, 11]'. Did you use
## factors with numeric labels for training, and numeric values for new data?

## Warning in predict.naiveBayes(nb1, newdata = test, type = "class"): Type
## mismatch between training and new data for variable 'train[, 12]'. Did you use
## factors with numeric labels for training, and numeric values for new data?

## Warning in predict.naiveBayes(nb1, newdata = test, type = "class"): Type
## mismatch between training and new data for variable 'train[, 13]'. Did you use
## factors with numeric labels for training, and numeric values for new data?
```

```
mean(p1==test[,15])
```

```
## [1] 0.7557193
```

```
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
confusionMatrix(as.factor(p1), reference=test[,15])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 4922 1591
##          1    0    0
##
##                Accuracy : 0.7557
##                  95% CI : (0.7451, 0.7661)
##     No Information Rate : 0.7557
##     P-Value [Acc > NIR] : 0.5067
##
##                   Kappa : 0
##
```

```
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.7557
##          Neg Pred Value :    NaN
##              Prevalence : 0.7557
##          Detection Rate : 0.7557
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : 0
##
```

Comparison of models

The linear model works well with correlated features in the data, whereas the naive Bayes model assumes the conditions of each variable to be independent of each other. Additionally, the linear model has a lower bias and higher variance, while the naive Bayes model has a higher bias and lower variance. The naive Bayes model trains more quickly on large datasets, but for this use case the difference was negligible.

Evaluation based on classification metrics

The naive Bayes model appears to have approximately 76% accuracy, which is better than the linear model at only 69%. The Bayes model was trained additional predictors, so that partially explains the accuracy increase. Furthermore, given that one of the a-priori probabilties was 76%, we know that this model is not doing any better than simply guessing based on the average probability of someone making less than $50K.

The confusion matrix above indicates the true negatives to be 4922 and the number of false negatives at 1591. Thus, we realize the issue with the model; it is so heavily biased in the negative direction that is simply returns a negative output every time, and that is why the accuracy is almost identical to the negative A-priori probability.

The benefit of the accuracy metric is that it yields a simple number which can be referenced to immediately get an estimate of the "goodness" of the model. Nevertheless, as seen in the case above, the accuracy can be misleading when the model is performing no better than a simple guess. The confusion matrix is useful to further clarify what is actually happening.