Classification Notebook

2/8/2023

Classification Notebook

Dinesh Angadipeta(DXA190032), Alejo Vinluan(ABV210001)

2/8/2023

How do linear models for Classification work?

Linear models for classification work by finding a linear relationship between vectors of data. With the vectors of data, the program can estimate where testing data fits in the model. There would be data X that can be used to predict data Y. For example, there may be a trend where incomes of $200,000 and up are most likely to come from employees that have Masters degrees. In the context of classification, we could use linear models to determine whether or not a person could be a good candidate for a loan or not.

Strengths

The linear models can be good for classification since it is estimating where the test data falls within the provided full set of data. The example within the textbook is based on women's height and weight. If we added another column with 'Human' or 'Alien' and had Alien rows that suggested Aliens are generally over 7 foot and 275 pounds, then the model would work well for predicting whether or not the test data given is Human or Alien. Furthermore, linear regression is simple and works well for larger datasets.

Weaknesses

Using linear models for classification may not work well since linear regression is for regression tasks rather than classification. Classification models don't use RSS. Instead, it uses counts of classes in regions. Linear regression is good for predicting Y from X rather than if the Y falls into a certain range, given X.

Data Set

This code will set the working directory to "Program 2", then reading the provided CSV and putting in variable "ford_listings". This dataset gives a model, year, price, transmission, mileage, fuelType, tax, mpg, and engineSizes for Fords sold in the UK.

```r
#setwd("Program 2") This line is required on my Windows device but breaks on my MacOS device.
ford_listings <- read.csv("./data/ford.csv", stringsAsFactors=TRUE)
# source - https://www.kaggle.com/datasets/adityadesai13/used-car-dataset-ford-and-mercedes
```

Splitting the data into 80/20

This code will divide the set into training set and test set. The "eighty" variable will take a sample of eighty percent of the dataset. Then, the data will be split into "training_data" and "testing_data". 80% of the data will be for training and the remaining will be for testing.

```
eighty <- sample(1:nrow(ford_listings), nrow(ford_listings)*0.8, replace=FALSE) # nolint
training_data  <- ford_listings[eighty, ]
testing_data   <- ford_listings[-eighty, ]
```

5 Examples of Built-in R functions

Here are 5 examples of R functions being used for data exploration. - head: View the first X rows of the given data - summary: View a quick summary of the testing data - mean: View the mean price of every used Ford sold in UK - median: View the median mileage of every used Ford sold in UK - sum: View the total tax price of all used Fords within the data

```
head(training_data, 5)
```

```
##         model year price transmission mileage fuelType tax  mpg engineSize
## 151    Fiesta 2017 10650       Manual   31800   Petrol 150 65.7        1.0
## 11851  Fiesta 2018  9900       Manual   33996   Petrol 145 65.7        1.0
## 17798  Fiesta 2016  9000    Automatic   29727   Petrol 145 47.9        1.6
## 8816   Fiesta 2017  9200       Manual    7391   Petrol   0 65.7        1.0
## 15260    Kuga 2019 19989    Semi-Auto    7600   Diesel 145 46.3        2.0
```

```
summary(training_data)
```

```
##        model           year          price          transmission
##   Fiesta   :5206   Min.   :1996   Min.   :  495   Automatic: 1085
##   Focus    :3672   1st Qu.:2016   1st Qu.: 8999   Manual   :12411
##   Kuga     :1803   Median :2017   Median :11290   Semi-Auto:  875
##   EcoSport: 932   Mean   :2017   Mean   :12279
##   C-MAX    : 438   3rd Qu.:2018   3rd Qu.:15299
##   Ka+      : 425   Max.   :2020   Max.   :54995
##  (Other)   :1895
##     mileage           fuelType          tax             mpg
##  Min.   :     1   Diesel   :4624   Min.   :  0.0   Min.   : 20.80
##  1st Qu.: 10000   Electric:    2   1st Qu.: 30.0   1st Qu.: 52.30
##  Median : 18381   Hybrid   :   19   Median :145.0   Median : 58.90
##  Mean   : 23381   Petrol   :9726   Mean   :113.1   Mean   : 57.91
##  3rd Qu.: 31068                    3rd Qu.:145.0   3rd Qu.: 65.70
##  Max.   :177644                    Max.   :580.0   Max.   :201.80
##
##    engineSize
##  Min.   :0.00
##  1st Qu.:1.00
##  Median :1.20
##  Mean   :1.35
##  3rd Qu.:1.50
##  Max.   :5.00
##
```

```r
mean(training_data$price)
```

```
## [1] 12279.25
```

```r
median(training_data$mileage)
```

```
## [1] 18381
```

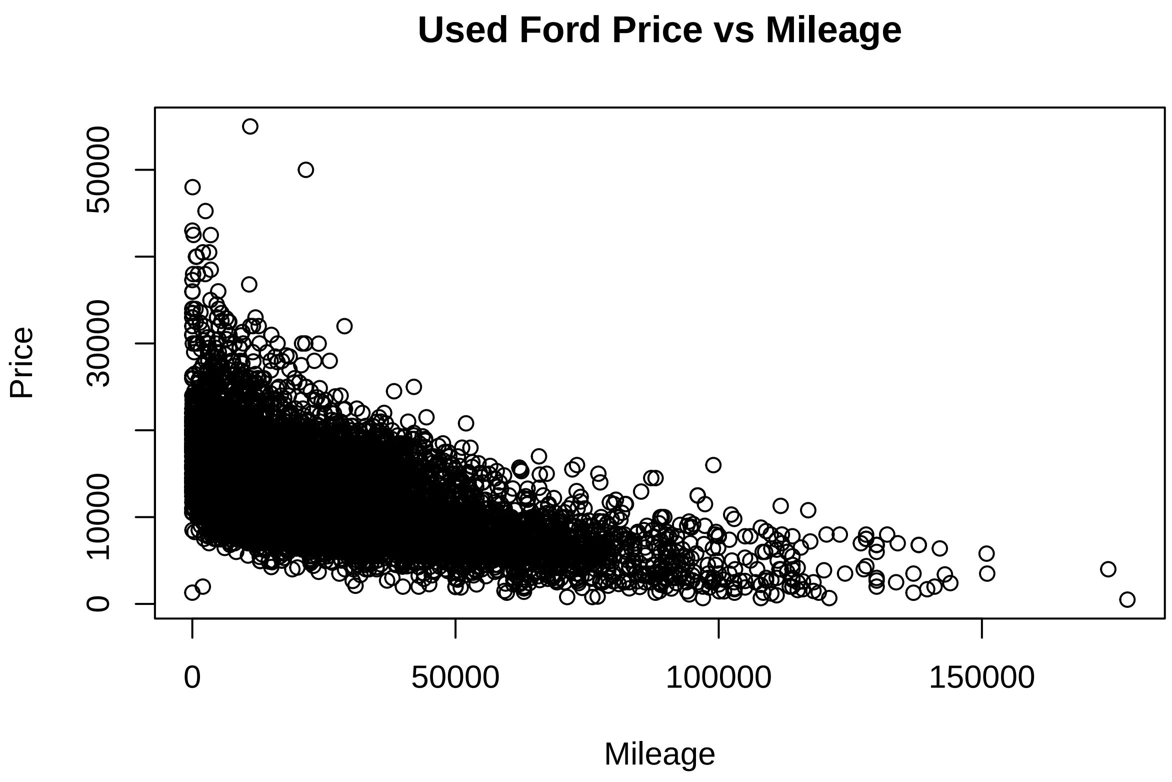
```r
sum(training_data$tax)
```

```
## [1] 1624665
```

Scatter Plot

This is how to create a scatter plot that compares a cars mileage to its price.

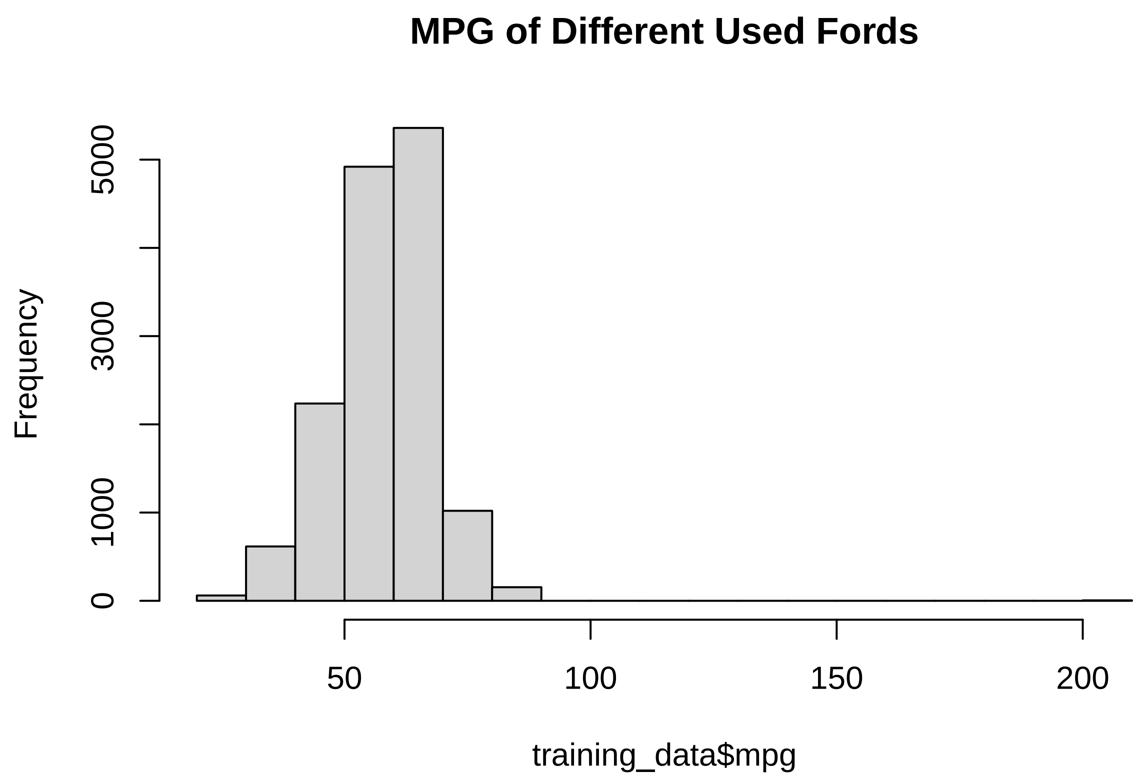
```r
plot(training_data$mileage, training_data$price, xlab = "Mileage", ylab = "Price", main="Used Ford Pric
```



Histogram

Here is another example of a visual graph which is a histogram of a car’s MPG.

```
hist(training_data$mpg, main='MPG of Different Used Fords')
```



Logistic Regression

This chunk of code runs a logistic regression model by finding the vehicle's model based on it's year, price, transmission, mileage, fuel type, tax, mpg, and fuel size. A summary is then printed about the regression model.

```
set.seed(1234)
regression_model <- glm(training_data$model ~ ., data = training_data, family = binomial(link = "logit"
```

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

```
summary(regression_model)
```

```
##
## Call:
## glm(formula = training_data$model ~ ., family = binomial(link = "logit"),
##     data = training_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.3962   0.0473   0.1000   0.1989   1.7362
```

```
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)            5.630e+02  9.618e+01   5.854 4.80e-09 ***
## year                  -2.859e-01  4.790e-02  -5.968 2.40e-09 ***
## price                  6.448e-04  4.294e-05  15.018  < 2e-16 ***
## transmissionManual     7.636e-01  2.675e-01   2.855  0.00431 **
## transmissionSemi-Auto -1.012e+00  2.806e-01  -3.607  0.00031 ***
## mileage                3.458e-05  4.537e-06   7.621 2.51e-14 ***
## fuelTypeElectric       9.401e+00  1.028e+03   0.009  0.99270
## fuelTypeHybrid         8.353e+00  2.536e+02   0.033  0.97372
## fuelTypePetrol         1.635e+00  3.170e-01   5.158 2.50e-07 ***
## tax                    9.183e-03  1.456e-03   6.309 2.81e-10 ***
## mpg                    1.246e-01  1.456e-02   8.554  < 2e-16 ***
## engineSize             1.755e-01  4.067e-01   0.431  0.66618
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2830.1  on 14370  degrees of freedom
## Residual deviance: 2223.1  on 14359  degrees of freedom
## AIC: 2247.1
##
## Number of Fisher Scoring iterations: 14
```

The summary states that the year, price, transmission, mileage, tax, and mpg are good indicators to find the whether the vehicle is a Fiesta. This is most likely because a vehicle's price and tax (which go hand in hand) will generally be the same of all vehicle models with similair mileage. Furthermore, mpg should be the same for vehicles of the same model.

```r
predicted <- predict(regression_model, newdata = testing_data, type = "response") #nolint
test_matrix <- table(testing_data$model, predicted > 0.5)
sum(test_matrix)
```

```
## [1] 3593
```

```r
colSums(test_matrix)
```

```
## FALSE  TRUE
##     1  3592
```

This model shows that after attempting to predict the model, there are 3593 total values from the testing data. Of the 3593 values, 3592 are are correct. This suggests a 99.9% accuracy. This data may be too accurate and that there is too much consistent data that the model can have 100% accuracy rating. For example, miles per gallon is unique amongst vehicle models. This suggests that mpg alone could be used to predict the model.

Naive Bayes

This chunk of code runs a naive Bayes model. After the model is run, it predicts a car's model based on it's year, price, transmission, mileage, fuel type, tax, mpg, and engine size. Accuracy is then calculated by comparing the prediction model's guesses with the actual information within the testing data.

```r
library(e1071)
bayes_model <- naiveBayes(as.factor(training_data$model)~., data = training_data, type="class") # nolin
prediction_model <- predict(bayes_model, newdata = testing_data)
accuracy <- mean(prediction_model == testing_data$model)
accuracy
```

```
## [1] 0.4247147
```

In this instance, the prediction model only had an accuracy of 41.9%. I believe this may be the case since Naive Bayes is viewing each factor as independent. This suggests that the model cannot reply on 1 factor, like mpg, in such a way that the logical regression was able to rely on mpg to make a complete prediction.

Evaluations of each model

Logical regression can be strong because it's easy to use. I was able to train my model by simply using the glm() function. Furthermore, the only trouble I had with logical regression was converting each Ford Model into a factor. That's because logical regression would not initially accept strings as valid for classification. Logical regression can be weak since it can be reliant on certain factors to determine the overall prediction. In the instance above, I suspect that the 99.9% accuracy rating was because it was relying on numbers unique to each vehicle. The mpg is unique, so logical regression can rely on those factors.

Naive Bayes can be strong since I was easily able to use it without having to convert the categorical data into numbers. In logical regression, I had to convert each string into factors. I was able to avoid this in the Naive Bayes approach. Furthermore, checking the accuracy of the data was simple as all I needed to do was compare each prediction to the real data located in the testing_data table. Naive Bayes can be weak since it assumes all factors are independent. For example, price and tax are directly correlated within the dataset. In this instance, there should be less weight on tax or less weight on price since they are dependent on each other.

Evaluation of classification metrics

I used accuracy in both instances to determine how the model is doing. I was able to compare how many correct predictions there were out of the entire testing set and find that the models had accuracies of 99% and 41.9%, respectively. This classification metric may not be the best since it only shows what the model predicted correctly. If the model had a 50% chance of guessing correctly, the accuracy would significantly change.