SS.R

Shalva Stulberg Data Science Final Project

2024-12-22

```r
# read in the csv file and save it as auto_data
auto_data <- read.csv("C:/Users/jbbpi/Downloads/auto-mpg(1).csv")
# see the dataset
View(auto_data)

#get the first 300 rows of the dataset and save it to a variable
beginning_auto <- head(auto_data, 300)
#see the beginning_auto dataset
View(beginning_auto)

# make horsepower into an integer, and save it back into the beginning_auto
dataset
horsepower_int <- as.integer(beginning_auto$horsepower)

## Warning: NAs introduced by coercion

beginning_auto$horsepower <- horsepower_int

#full linear regression model, with all continuous variables
full_model <- lm(mpg ~ cylinder + displacement + horsepower_int + weight +
acceleration + model.year + origin ,  data = beginning_auto)
#summarize the full model
summary(full_model)

##
## Call:
## lm(formula = mpg ~ cylinder + displacement + horsepower_int +
##     weight + acceleration + model.year + origin, data = beginning_auto)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -9.298 -1.641  0.089  1.578 13.587
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     5.8118427  4.9722754   1.169  0.24342
## cylinder       -0.4562389  0.2996507  -1.523  0.12896
## displacement    0.0101272  0.0067125   1.509  0.13246
## horsepower_int -0.0172493  0.0120542  -1.431  0.15351
## weight         -0.0053282  0.0005719  -9.316  < 2e-16 ***
## acceleration   -0.0278409  0.0956550  -0.291  0.77122
## model.year      0.4439943  0.0605543   7.332 2.27e-12 ***
## origin          0.9931335  0.2993730   3.317  0.00102 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.683 on 290 degrees of freedom
##   (2 observations deleted due to missingness)
## Multiple R-squared:  0.823,  Adjusted R-squared:  0.8187
## F-statistic: 192.7 on 7 and 290 DF,  p-value: < 2.2e-16
```

```
# get the coefficients on the full model, for the linear equation
coef(full_model)
```

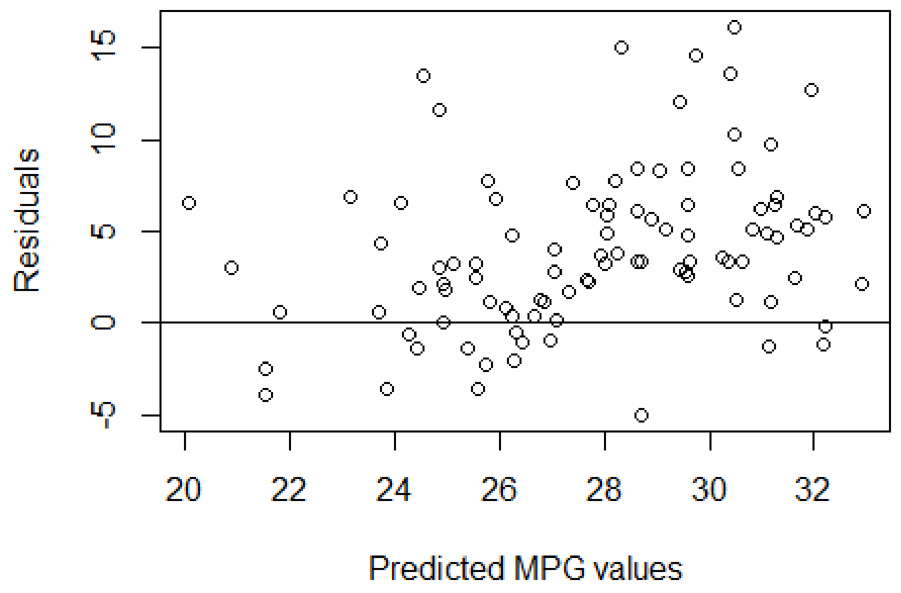
```
##    (Intercept)       cylinder   displacement horsepower_int         weight
##    5.811842694   -0.456238922    0.010127210   -0.017249273   -0.005328159
##   acceleration     model.year         origin
##   -0.027840888    0.443994255    0.993133529
```

```
# equation for full model
#mpg = 5.8 -.456*cylinder + .01*displacement - .017*horsepower -
.005*weight+.99*origin + .44*model.year - .028 *acceleration
# R squared = .823, adjusted R-squared = .8187


#run multiple linear regression on statistically significant variables
mult_reg_model <- lm(mpg ~ weight + model.year + origin, data =
beginning_auto)
#summarize the mult_reg_model
summary(mult_reg_model)
```

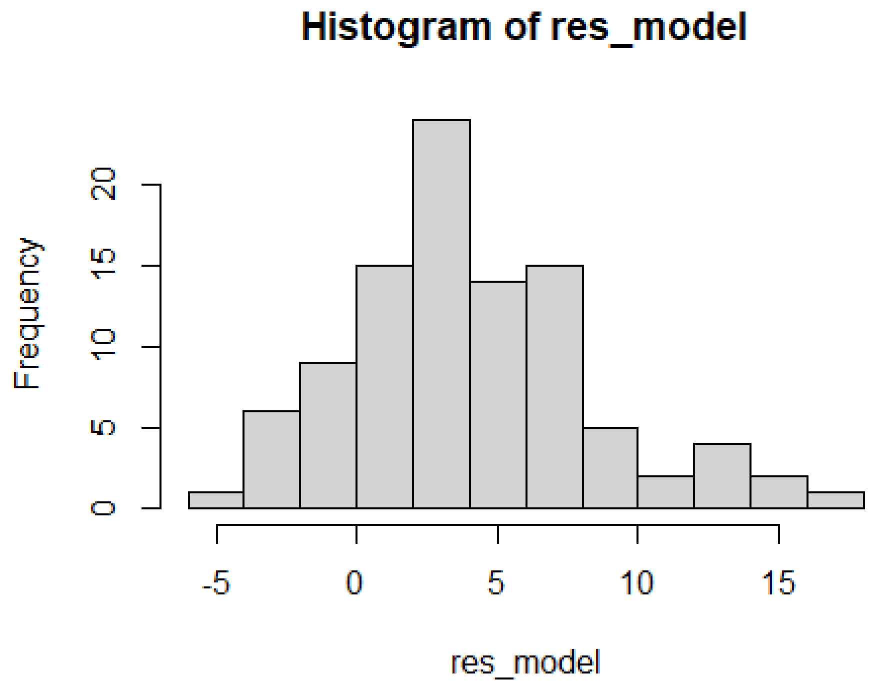
```
##
## Call:
## lm(formula = mpg ~ weight + model.year + origin, data = beginning_auto)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.1750 -1.5586  0.0463  1.6506 13.6915
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.2289898  4.3438467   0.743  0.45786
## weight      -0.0056852  0.0002215 -25.664  < 2e-16 ***
## model.year   0.4585332  0.0559760   8.192 7.82e-15 ***
## origin       0.8495008  0.2646593   3.210  0.00147 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.676 on 296 degrees of freedom
## Multiple R-squared:  0.8206, Adjusted R-squared:  0.8188
## F-statistic: 451.3 on 3 and 296 DF,  p-value: < 2.2e-16
```

```r
#get the coefficients for the multiple linear regression model
coef(mult_reg_model)

##  (Intercept)       weight   model.year       origin
##  3.228989828 -0.005685203  0.458533182  0.849500785

#equation for full model is: mpg = 3.223 - .006*weight + .46 * model.year +
.85*origin
# R-squared = .82, adjusted R-squared = .819


#simple linear regression model
simple_reg_model <- lm(mpg ~ weight, data = auto_data)
summary(simple_reg_model)

##
## Call:
## lm(formula = mpg ~ weight, data = auto_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -12.012  -2.801  -0.351   2.114  16.480
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 46.3173644  0.7952452   58.24   <2e-16 ***
## weight      -0.0076766  0.0002575  -29.81   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.345 on 396 degrees of freedom
## Multiple R-squared:  0.6918, Adjusted R-squared:  0.691
## F-statistic: 888.9 on 1 and 396 DF,  p-value: < 2.2e-16

#get the coefficients for the simple linear regression model
coef(simple_reg_model)

## (Intercept)      weight
## 46.31736442 -0.00767661

# equation for simple model is: mpg = 46.3 -.008*weight
#R-squared is .69, adjusted R-squared is .69


# get the last 98 rows of the dataset
end_data <- tail(auto_data, 98)
# see the end_data dataset
View(end_data)

#make prediction, with mult_reg_model as the training data, and end_data as
the testing data
```

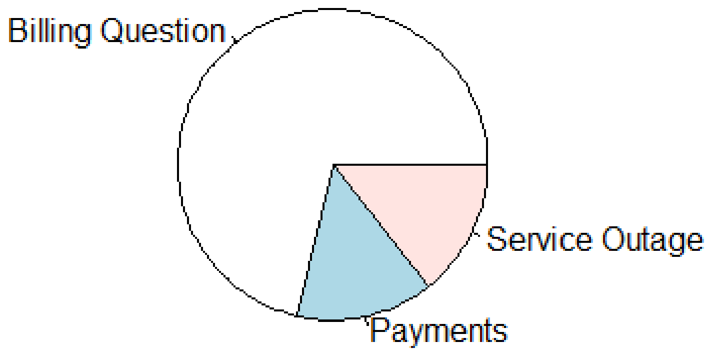
```
prediction <- predict(mult_reg_model,end_data)
prediction

##      301      302      303      304      305      306      307      308 
## 20.85922 27.79517 28.07943 30.51750 29.04263 25.12312 25.54951 24.95256 
##      309      310      311      312      313      314      315      316 
## 25.77123 29.42157 31.27167 28.70851 30.98172 25.53617 24.44461 23.68848 
##      317      318      319      320      321      322      323      324 
## 21.53947 29.17142 27.04756 28.00836 28.62236 29.58316 30.46437 24.84258 
##      325      326      327      328      329      330      331      332 
## 30.46437 29.75700 28.33570 24.83930 23.13374 31.94252 31.17830 30.26539 
##      333      334      335      336      337      338      339      340 
## 31.12145 25.91621 28.70196 27.39764 24.24563 29.44103 27.06352 26.23917 
##      341      342      343      344      345      346      347      348 
## 26.32445 25.72750 27.66047 32.94115 30.55992 32.91272 31.17874 31.69040 
##      349      350      351      352      353      354      355      356 
## 31.26401 31.63355 28.62695 29.59344 27.68890 29.61858 28.87951 30.35438 
##      357      358      359      360      361      362      363      364 
## 29.55845 28.05187 27.93817 23.70597 24.10394 26.43159 26.26104 21.80471 
##      365      366      367      368      369      370      371      372 
## 20.04230 23.82296 21.52045 26.86826 26.66928 28.06215 27.03881 27.32307 
##      373      374      375      376      377      378      379      380 
## 26.12918 25.39011 24.42362 31.27101 31.86468 32.17736 29.59716 29.59716 
##      381      382      383      384      385      386      387      388 
## 31.09717 30.84134 30.61393 32.20579 32.20579 32.03523 24.93529 24.53732 
##      389      390      391      392      393      394      395      396 
## 26.98196 25.56066 28.22615 28.20428 24.90686 25.81650 30.41823 28.63067 
##      397      398 
## 26.75455 26.21446

## make residual plot:
# get a list of the residuals, and plot them.Made a line at y=0 to help see
the model's fit.
res_model <- end_data$mpg - prediction
plot(prediction,res_model, xlab = "Predicted MPG values", ylab =
"Residuals",)
abline(h=0)
```



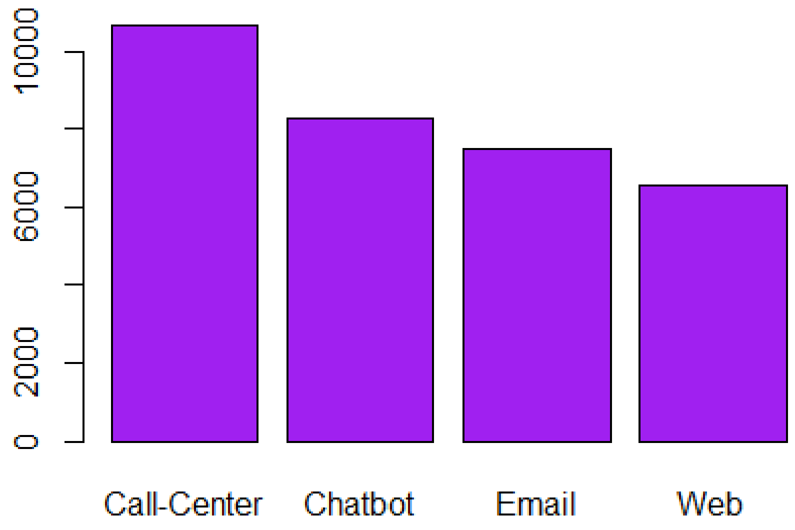
```
# histogram plotting residuals
hist(res_model)
```



```
#import the call dataset
call_data <- read.csv("C:/Users/jbbpi/Downloads/Call_Center.csv")
#see the dataset
View(call_data)

###why do most customers call us?
# made Reason into a table to see the counts of each reason
# made pie chart of all reasons
Reasons <- table(call_data$Reason)
pie(Reasons)
```

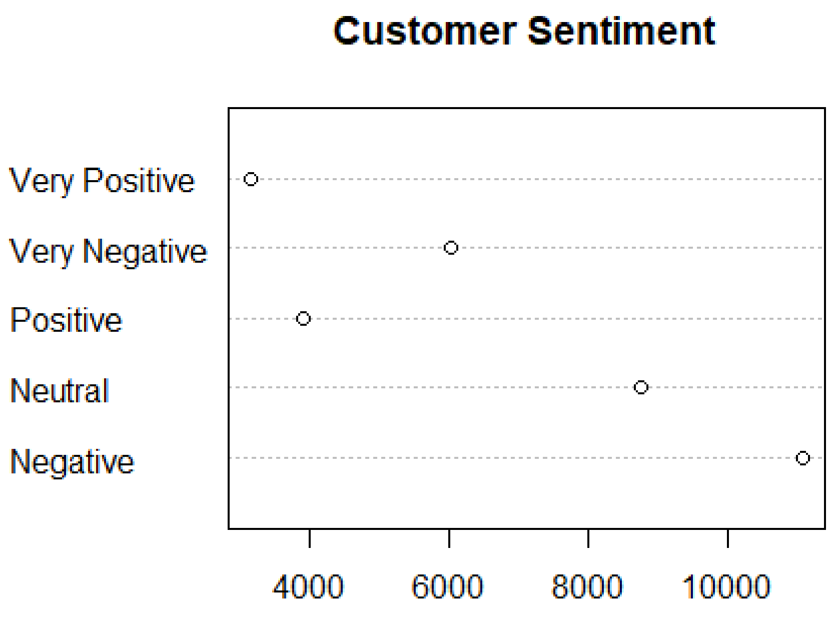


```
###Through which channel do our customers contact us?
# made Channel into a table to see get counts of each Channel
# made a bar graph showing the counts of Channels
Channel <- table(call_data$Channel)
barplot(Channel, col = "purple" )
```



```
###how do our customers feel about us?
# made Sentiment into a table to get the counts of each Sentiment
# made a dotchart showing the counts of each sentiment
Sentiment <- table(call_data$Sentiment)
dotchart(Sentiment,main="Customer Sentiment")
```

```
## Warning in dotchart(Sentiment, main = "Customer Sentiment"): 'x' is
neither a
## vector nor a matrix: using as.numeric(x)
```



Customer Sentiment
Very Positive
Very Negative
Positive
Neutral
Negative
4000
6000
8000
10000