Multiple Linear Regression

Dom

1 July 2018

## Packages used

library(tidyverse)  
library(caTools)

The extension of the previous post will be adding in more features into the pastures dataset to see if we can predict **rent.grassy** better than just using one feature (**rent.arable**). We’ll use linear regression again.

## Aim: find *multiple features* that can be used to predict the dependent variable

## Caveats: it’s a small dataset (67 rows), this is more of an experiment

Jumping straight into the data prep we’ll rename columns, pull out the features we want and split the data into test and training sets.

pastures %>%  
 rename(index = Rental,  
 rent.arable = price,  
 cows = per,  
 difference = grassy,  
 rent.grassy = acre) %>%  
 select(c("rent.arable",  
 "cows",  
 "difference",  
 "rent.grassy")) -> pastures.processed  
  
pastures.processed

## # A tibble: 67 x 4  
## rent.arable cows difference rent.grassy  
## <dbl> <dbl> <dbl> <dbl>  
## 1 15.5 17.2 0.240 18.4  
## 2 22.3 18.5 0.200 20.0  
## 3 12.4 11.1 0.120 11.5  
## 4 31.8 5.54 0.120 25.0  
## 5 83.9 5.44 0.0400 62.5  
## 6 72.2 20.4 0.0500 82.5  
## 7 27.1 31.2 0.270 25.0  
## 8 40.4 4.29 0.100 30.7  
## 9 12.4 8.69 0.410 12.0  
## 10 69.4 6.63 0.0400 61.2  
## # ... with 57 more rows

set.seed(345) # Set a random seed to allow the same split to be replicated  
  
split <- sample.split(pastures.processed$rent.grassy, SplitRatio = 0.75) # Assign T/F to each row (100%)  
train <- subset(pastures.processed, split == T) # Pull out those that have T (75%)  
test <- subset(pastures.processed, split == F) # Pull out those that have F (25%)

Now using this data let’s generate a regressor using all the features and see which ones have low p values.

### Round 1

#### Significant level =< 0.05

regressor <- lm(formula = rent.grassy ~ .,  
 data = train)  
  
summary(regressor)

##   
## Call:  
## lm(formula = rent.grassy ~ ., data = train)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -22.1338 -5.0120 0.3311 4.5997 27.1046   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -8.23921 5.00885 -1.645 0.10680   
## rent.arable 0.96771 0.08983 10.772 3.63e-14 \*\*\*  
## cows 0.39794 0.12812 3.106 0.00324 \*\*   
## difference 3.92431 19.18753 0.205 0.83885   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 9.364 on 46 degrees of freedom  
## Multiple R-squared: 0.8548, Adjusted R-squared: 0.8453   
## F-statistic: 90.26 on 3 and 46 DF, p-value: < 2.2e-16

Ok we can see **difference** has quite a high p value so we’ll discount it and run the regressor again (the Signif. codes is a great function to identify feature importance). This is **backwards elimination**.

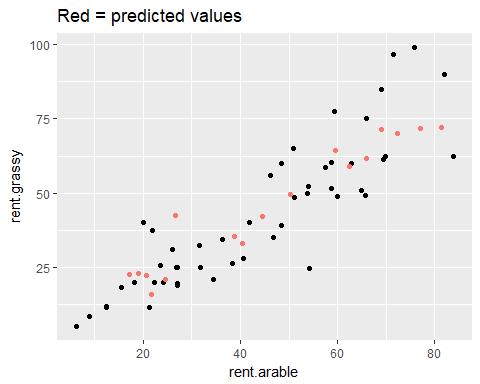
### Round 2

regressor <- lm(formula = rent.grassy ~ rent.arable + cows,  
 data = train)  
  
summary(regressor)

##   
## Call:  
## lm(formula = rent.grassy ~ rent.arable + cows, data = train)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -22.0794 -4.9476 0.2529 4.5329 27.0301   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -7.47171 3.28357 -2.275 0.0275 \*   
## rent.arable 0.95486 0.06355 15.025 < 2e-16 \*\*\*  
## cows 0.41764 0.08366 4.992 8.64e-06 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 9.268 on 47 degrees of freedom  
## Multiple R-squared: 0.8547, Adjusted R-squared: 0.8485   
## F-statistic: 138.2 on 2 and 47 DF, p-value: < 2.2e-16

Now **cows** p value has decreased. But **rent.arable**’s p value has decreased as well, looking good and the **Adjusted R-sqaured** value has creeped up a tad. Let’s test this regressor on our test data keeping the same plot as the previous test (**rent.arable** against **rent.grassy**).

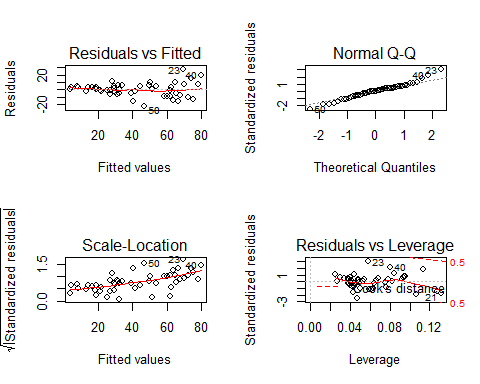
ypred <- predict(regressor,test) # Predicting rent.grassy  
  
ggplot()+  
 geom\_point(aes(x = train$rent.arable, y = train$rent.grassy)) +  
 geom\_point(aes(x = test$rent.arable, y = predict(regressor, test), color = "red")) +  
 ylab("rent.grassy") +  
 xlab("rent.arable") +  
 ggtitle("Red = predicted values") +  
 theme(legend.position = "none")



It’s looking like the regressor is fitting the data well but isn’t too precisely fitted.

Next up let’s look at the diagnostic plots. This might be overkill but let’s investigate!

par(mfrow = c(2,2))  
plot(regressor)



### Residuals vs Fitted

A relatively straight line with points spread somewhat evenly and no noticeable patterns so we can assume that there a **no non-linear relationships**.

On the other hand there could have been non-linear relationships that the model couldn’t have picked up and this plot would have shown us/hinted at them. In the case of a quadratic relationship there would be a parabola like curve.

### Normal Q-Q

We’re using this plot to check if our residuals are normally distributed. In the case of normally distributed residuals the points would lie on the dotted line. Here most do but not all of them. Overall it doesn’t look too bad but there are some points I’m worried about at the upper and lower end of the graph.

### Scale-Location

Ideally we want a horizontal line to indicate **uniform variance** across the range i.e (**Homoscedasticity**). Not the steepest line and the residuals aren’t too evenly spread but they start to widen. Let’s try removing #23, #50 and #40 to see if this can be improved upon.

### Residuals vs Leverage

All of the points are located inside the Cook’s distance thus we have no influential point(s). If there were points outside the Cook’s distance and we removed them from the model there would be a noticeable change e.g. R Squared value.

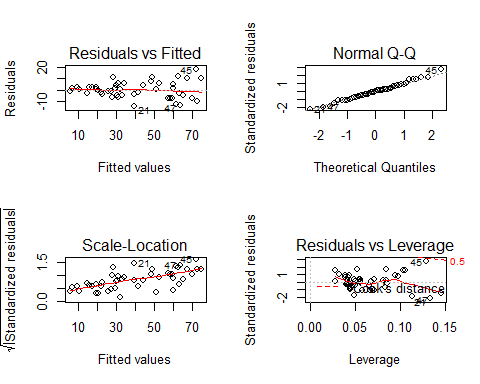
Let’s remove #23, #50 and #40 and see if we can improve the model.

### Round 3

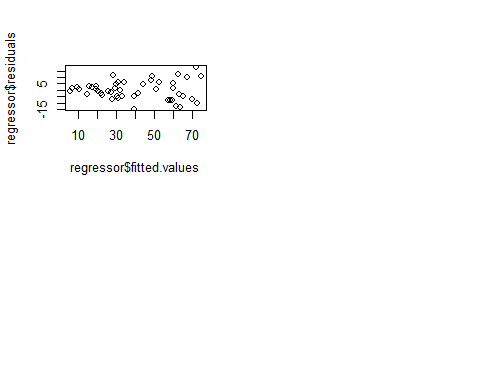
regressor <- lm(formula = rent.grassy ~ rent.arable + cows,  
 data = train[c(-23,-50,-40),])  
  
summary(regressor)

##   
## Call:  
## lm(formula = rent.grassy ~ rent.arable + cows, data = train[c(-23,   
## -50, -40), ])  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -14.4335 -4.6055 0.1436 4.3684 18.1676   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -4.71961 2.63453 -1.791 0.0801 .   
## rent.arable 0.89767 0.05244 17.119 < 2e-16 \*\*\*  
## cows 0.37301 0.06674 5.589 1.35e-06 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 7.273 on 44 degrees of freedom  
## Multiple R-squared: 0.8893, Adjusted R-squared: 0.8843   
## F-statistic: 176.7 on 2 and 44 DF, p-value: < 2.2e-16

par(mfrow = (c(2,2)))  
plot(regressor)



plot(regressor$fitted.values, regressor$residuals)



The p value of cows has decreased again, adjusted R squared value has increased and the slope coefficient has decreased and is now less significant in the model. Looking at the diagnostic plots

* Residuals vs Fitted maintains it’s somewhat straightness
* Normal Q-Q has improved with less points away from the dotted line
* Scale-Location has changed minimally, not good
* Residuals vs Leverage has #45 edging close to the Cook’s distance line, not liking that (could become influential on the model)

## Summary

Overall I think this is an improvement on the previous model involving only one variable. The data isn’t overfitting which is good as well.

How could we improve it further? More data. Given that there are only 50 points to train the model on it’s not enough in my opinion. Maybe there is data that we could incorporate to improve it further that we don’t know about e.g. the weather on each day, type of cow or type of grass.

I think the use of diagnostic plots can be handy to discover hidden insights in the data that weren’t necessarily considered when creating the model, whether its excluding points or tweaking the formula of the model.