---
title: " Regression"
output: html_notebook

Manuel Romero
Sagar Darnal

We will use this data to predict the prices(target) of used cars.

Write a paragraph explaining in
general terms how linear regression works, and what are its strengths and weaknesses.

Linear Regression is used to predict data. We have values called predictors(x) and we use those values to predict a target value(y).
Furthermore it is used to examine the relationship of x and y by creating an equation with a slope and intercept. It's strenghts are in how simple an algorithm it is and is a good model to use when data is linear, however it;s weakness is also that it assumes the data is has a linear relationship and therefore may not be a good fit. Furhtermore it is hard to see which predictors are good predictors and if those predictors are also based on other predictors. Correlation is not causation, as seen in class at the beach ice cream sales might go up as well as shark attacks, a linear model may correlate that ice creams and shark attacks go together.

data taken from:
https://www.kaggle.com/datasets/245df9d02ee0cdff29a4cb6f099efb4e9f4bf69a1ff652838dde679450491a7a

sources
https://data.library.virginia.edu/diagnostic-plots/

true_car_listings

//getting data and saving it into df
reading only 10,000 entry points for memory sake
```{r}
df <- read.csv("true_car_listings.csv", na.strings = "NA", header =TRUE, nrows = 50000)
str(df)
```

split 80/20 train/test data set

```{r}
set.seed(1234)
i <- sample(1:nrow(df), nrow(df)*.80, replace = FALSE)
train <- df[i,]
test <- df[-1,]
```

creating lm1 with train data and runing data exploration functions on it
number of rows in train data set
number of columns in train data set
names of columns in train data set
summary of train data set
list structure of data set
```{r}
print('number of rows in train data set')
nrow(train)
print('number of columns in train data set')
ncol(train)
print('names of columns in train data set')
names(train)
print('summary of train data set')
summary(train)
print('list structure of train data set')
str(train)
```

```
creating informative graphs
plotting price as a function of mileage
```{r}
plot(train$Mileage, train$Price, xlab = "Mileage", ylab = "Price")
```
Here we see that the lower the mileage the higher the price, but we also see low mile vehicles that are cheap, therfore this is not the only predictor to consider

making a plot
plotting price as a function of year
```{r}
plot(train$Year, train$Price, xlab = "Year", ylab = "Price")

```
This graph shows that the higher the year the higher the price, in other words the newer the car the more it is priced at. As before there are still some fairly new cars that are worth lower and some old cars that are worth kinda high. This means mileage and year affect the price of a car but is not the only thing that affects it which makes sense since there are some more luxurious make and models.

creating lm1 with price as a target and mileage as a predictor
```{r}
lm1 <- lm(Price~Mileage, data = train)
summary(lm1)

```
As noted before from the plot of price of used cars as a function of their mileage this alone is not a good predictor and this is confirmed by the R-squared value of 0.3425. This is a low value which makes sense since a lot of low mileage cars had a huge range of price values. Therefore it would be very hard to calculate the price of a car with low miles without knowing any other information. We see that the resduals have a range from -26420 to 243031, this is a big range and wanting to predict something with a decent probability of accuracy would result in a big range, in other words not very accurate

plotting the residuals
```{r}
par(mfrow=c(2,2))
plot(lm1)
```
The residuals vs fitted plot shows that there was an attempt to linearly plot the points but this does not work since there are so many other points that the line is unable to capture. The Normal Q-Q shows a line at first and then curves a lot towards the end. The residuals are not normally distributed. From scale-location we see that the values aren't randomly distributed, they all bunch up to the right side of the graph.The residual vs leverage graph shows us that most of the data points are are clustered to the left and that there are some influential observations that influence the regression line, removing these may create a better fit line, however there are many others that are stacked on the left side that create concern. This shows that the current model is not an ideal linear regression model.

Building a Multiple linear regression model
```{r}
lm2 <- lm(Price ~ Mileage + Year, data = train)
summary(lm2)
par(mfrow=c(2,2))
plot(lm2)
```
Here we used a multiple linear regression model to target price and used mileage and year of the used cars to predict the value of price. The R squared went up by a small amount it shows that this is a better method but there must be a better version we can achieve.

creating a third linear regression with predictor model along with mileage and year
```{r}
lm3 <- lm(Price ~ poly(Mileage,2) + Year + Make, data = train)
summary(lm3)
par(mfrow=c(2,2))
```
```

```
plot(lm3)
```
Here once we factor in the make of the car we get a better line, we achieve a 0.6329 R squared with lm3. We also did mileage to a polynomial of 2. LM3 is a better multiple linear regression model. While the residual plots are still not ideal the line in Residuals vs Fitted is still not ideal but there are significantly more values on the line or near the line than other models.The Scale Location looks much more random but there is still a lot of clustering. The Residuals vs Leverage still shows influential points. However the R squared value shows that this is a better model than any of the others.

running correlation and mse on test data and Lm1

```{r}
pred1 <- predict(lm1, newdata=test)
cor1 <- cor(pred1, test$Price)
mse1 <- mean((pred1-test$Price)^2)
rmse1 <- sqrt(mse1)

print(paste('correlation for lm1:', cor1))
print(paste('mse: for lm1', mse1))
print(paste('rmse for lm1:', rmse1))
#-------------------------------------------
pred2 <- predict(lm2, newdata=test)
cor2 <- cor(pred2, test$Price)
mse2 <- mean((pred2-test$Price)^2)
rmse2 <- sqrt(mse2)

print(paste('correlation for lm2:', cor2))
print(paste('mse: for lm2', mse2))
print(paste('rmse for lm2:', rmse2))
#-------------------------------------------
pred3<- predict(lm3, newdata=test)
cor3 <- cor(pred3, test$Price)
mse3 <- mean((pred3-test$Price)^2)
rmse3 <- sqrt(mse3)

print(paste('correlation for lm3:', cor3))
print(paste('mse: for lm3', mse3))
print(paste('rmse for lm3:', rmse3))

```
Our correlation for lm1 between our test and train data is decent but not ideal and lm2 has very little improvement. The rmse for both lm1 and lm2 is over 12000 on the other hand lm3 has a rmse of just over 9300 which is still a big number but better than 12000. The correlation is at nearly .80 which is better although still not ideal. These results happened because the first 2 models weren't good linear regression models and lm3 is slightly better, a way to get a much better fit would be by using make not model since each model has makes that are cheaper or more expensive, we werent able to do this in this data set since the data set is very extensive and we ran out of memory if we use the whole data set so we only used 50,000 entries which unforunatly didnt get all the makes, so using make in lm3 instead of model results in some data points in test to have make values not in the train data, therefore making lm3 not work.

Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Cmd+Option+I*.

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Cmd+Shift+K* to preview the HTML file).

The preview shows you a rendered HTML copy of the contents of the editor. Consequently, unlike *Knit*, *Preview* does not run any R code chunks. Instead, the output of the chunk when it was last run in the editor is displayed.