Portfolio Project 2: Regression File

David Teran & HuyNguyen

Linear Regression

This notebook will use a dataset consisting of 10k or more rows of data and will plot out a linear regression. First, dividing the data into 80/20 train/test blocks, then create 3 different linear regression models.

Created by David Teran & Huy Nguyen on February 15, 2023

Linear Regression has us trying to find a target quantitative value using one or more predictors. Some strengths are that it works well when the data follows a linear pattern and has low variance. The main weakness it has is that it has a high bias because it tries to fit into a linear shape.

First, Read in the dataset to use and clean it of empty data

```
TestData <- read.csv("car details v4.csv", na.strings = "NA", header = TRUE)
TestData <- TestData[c('Make','Model','Price','Year','Kilometer')]
data(TestData)
```

```
## Warning in data(TestData): data set 'TestData' not found
```

```
str(TestData)
```

```
## 'data.frame':    2059 obs. of  5 variables:
##  $ Make     : chr  "Honda" "Maruti Suzuki" "Hyundai" "Toyota" ...
##  $ Model    : chr  "Amaze 1.2 VX i-VTEC" "Swift DZire VDI" "i10 Magna 1.2 Kappa2" "Glanza G" ...
##  $ Price    : int  505000 450000 220000 799000 1950000 675000 1898999 2650000 1390000 575000 ...
##  $ Year     : int  2017 2014 2011 2019 2018 2017 2015 2017 2017 2015 ...
##  $ Kilometer: int  87150 75000 67000 37500 69000 73315 47000 75000 56000 85000 ...
```

```
#colSums(is.na.data.frame(TestData))
TestData <- na.omit(TestData)
```

Dividing Data into train/test sets

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

Using 5 or R functions for data exploration

```
summary(train)
```

```
##       Make               Model               Price               Year
##  Length:1647        Length:1647        Min.   :   49000   Min.   :1988
##  Class :character   Class :character   1st Qu.:  480000   1st Qu.:2014
##  Mode  :character   Mode  :character   Median :  850000   Median :2017
##                                        Mean   : 1755162   Mean   :2016
##                                        3rd Qu.: 1957500   3rd Qu.:2019
##                                        Max.   :35000000   Max.   :2022
##    Kilometer
##  Min.   :     0
##  1st Qu.: 28000
##  Median : 50000
##  Mean   : 52347
##  3rd Qu.: 72000
##  Max.   :261236
```

```
names(train)
```

```
## [1] "Make"      "Model"     "Price"     "Year"      "Kilometer"
```

```
# train$Price <- as.numeric(train$Price)
# train$Year <- as.numeric(train$Year)
# train$Kilometer <- as.numeric(train$Kilometer)
#
# test$Price <- as.numeric(test$Price)
# test$Year <- as.numeric(test$Year)
# test$Kilometer <- as.numeric(test$Kilometer)

print(paste("Correlations: "))
```

```
## [1] "Correlations: "
```

```
print(paste(""))
```

```
## [1] ""
```

```
cor(train$Year, train$Price)
```

```
## [1] 0.3055168
```

```
cor(train$Kilometer, train$Price)
```

```
## [1] -0.2625474
```

```
var(train$Price)
```

```
## [1] 6.520639e+12
```

```r
head(train)
```

```
##                Make                                     Model   Price Year
## 1004        Toyota          Innova 2.4 ZX 7 STR [2016-2020] 3100000 2021
## 623  Maruti Suzuki Vitara Brezza ZDi+ Dual Tone [2017-2018]  715000 2017
## 934         Nissan                               Magnite XL  625000 2021
## 400          Honda                            City S Diesel  645000 2015
## 1626       Hyundai                       Elite i20 Asta 1.2  550000 2015
## 1103         Honda                            City 1.5 S MT  500000 2013
##      Kilometer
## 1004     29000
## 623      71000
## 934      28000
## 400      64000
## 1626     33000
## 1103     65000
```

```r
tail(train)
```

```
##                Make                                    Model   Price Year
## 1654          Tata                         Nexon XM Diesel  725000 2018
## 579     Volkswagen                              Polo GT TSI  625000 2016
## 1601 Mercedes-Benz GLC 220d 4MATIC Progressive [2019-2021] 6500000 2020
## 401          Honda                           City SV Diesel  550000 2014
## 1734         Honda                            City 1.5 S MT  250000 2009
## 1957       Renault                       Kwid 1.0 RXT AMT Opt  490000 2019
##      Kilometer
## 1654     65000
## 579      68000
## 1601     13000
## 401      85000
## 1734     72256
## 1957     32000
```

```r
mean(train$Price)
```

```
## [1] 1755162
```

```r
mean(train$Kilometer)
```

```
## [1] 52346.91
```

```r
range(train$Kilometer)
```

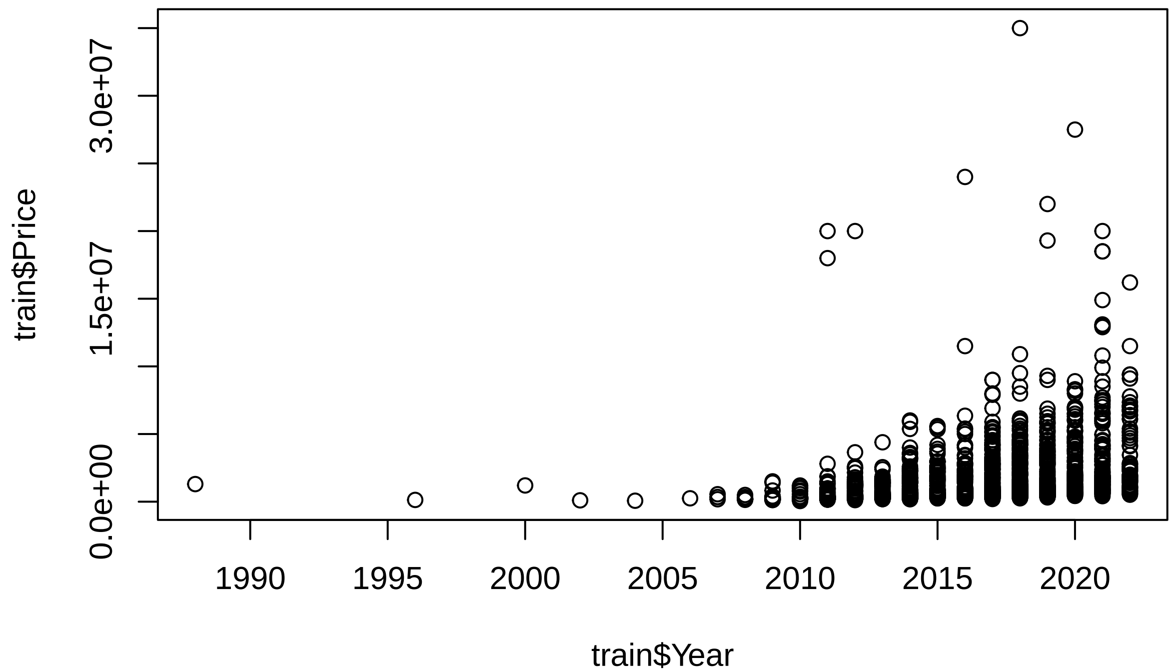
```
## [1]      0 261236
```

```r
range(train$Price)
```

```
## [1]    49000 35000000
```

Creating informative graphs using training data

```
plot(train$Year, train$Price)
```




```
hist(train$Year)
```

Building simple linear regression model

```
lmPrice <- lm(Price~Year, data = train)
summary(lmPrice)
```

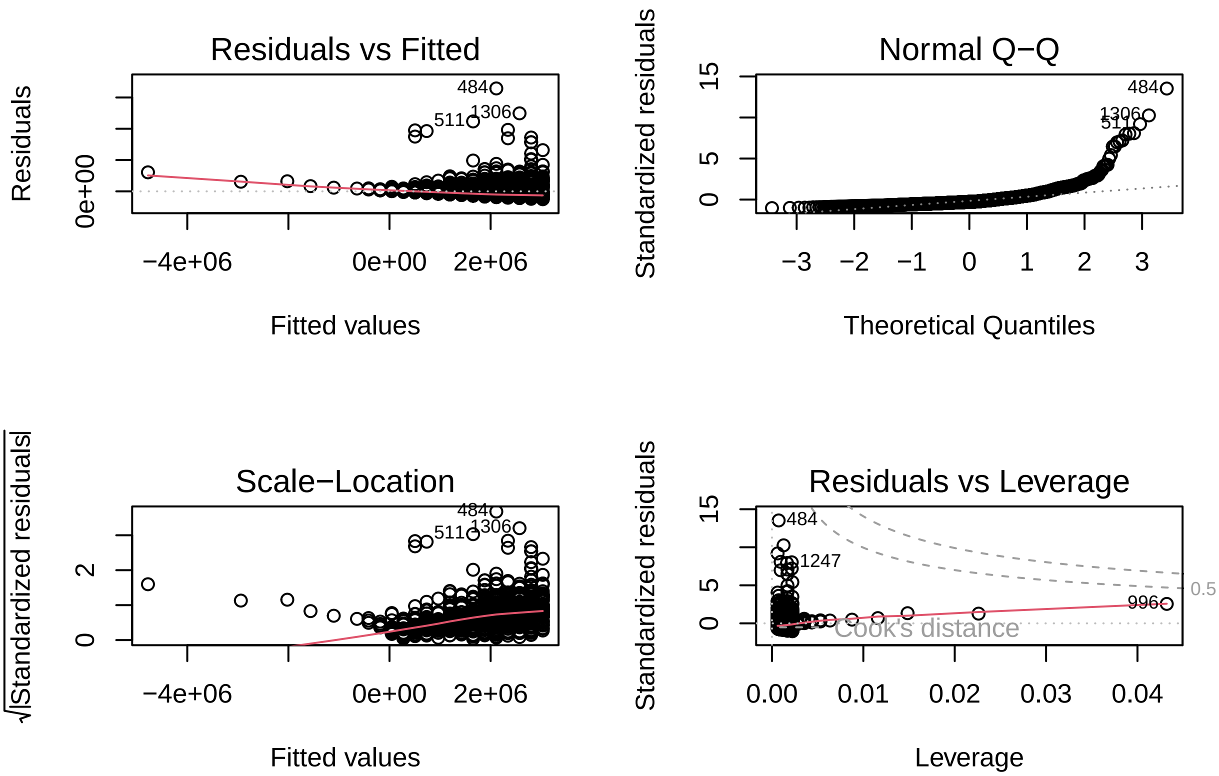
```
##
## Call:
## lm(formula = Price ~ Year, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2516597 -1205048  -669133   442574 32887135
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -461387461   35589296  -12.96   <2e-16 ***
## Year            229683      17650   13.01   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2432000 on 1645 degrees of freedom
## Multiple R-squared:  0.09334,    Adjusted R-squared:  0.09279
## F-statistic: 169.4 on 1 and 1645 DF,  p-value: < 2.2e-16
```

Write a thorough explanation of the information in the model summary.

The model summary for the linear regression model presents several different metrics used concerning model. The estimated coefficients of the intercept and the year value are given along with the standard error, t-value, and p-values. The standard error gives an estimate of the variation in the coefficient value. The p-value helps indicate if there exists a relationship between the predictor and the target variable. The residual standard error is 2432000 on 1645 degrees of freedom, indicating how far off the model was from the data, which for the data and predictors used is quite off. The multiple r-squared is 0.09334, meaning that not much of the variance in Price is predicted by the Year. The F-statistic indicates that the Price and Year variables are good predictors.

Plot out residuals

```
par(mfrow=c(2,2))
plot(lmPrice)
```



write a thorough explanation of what the residual plot tells you

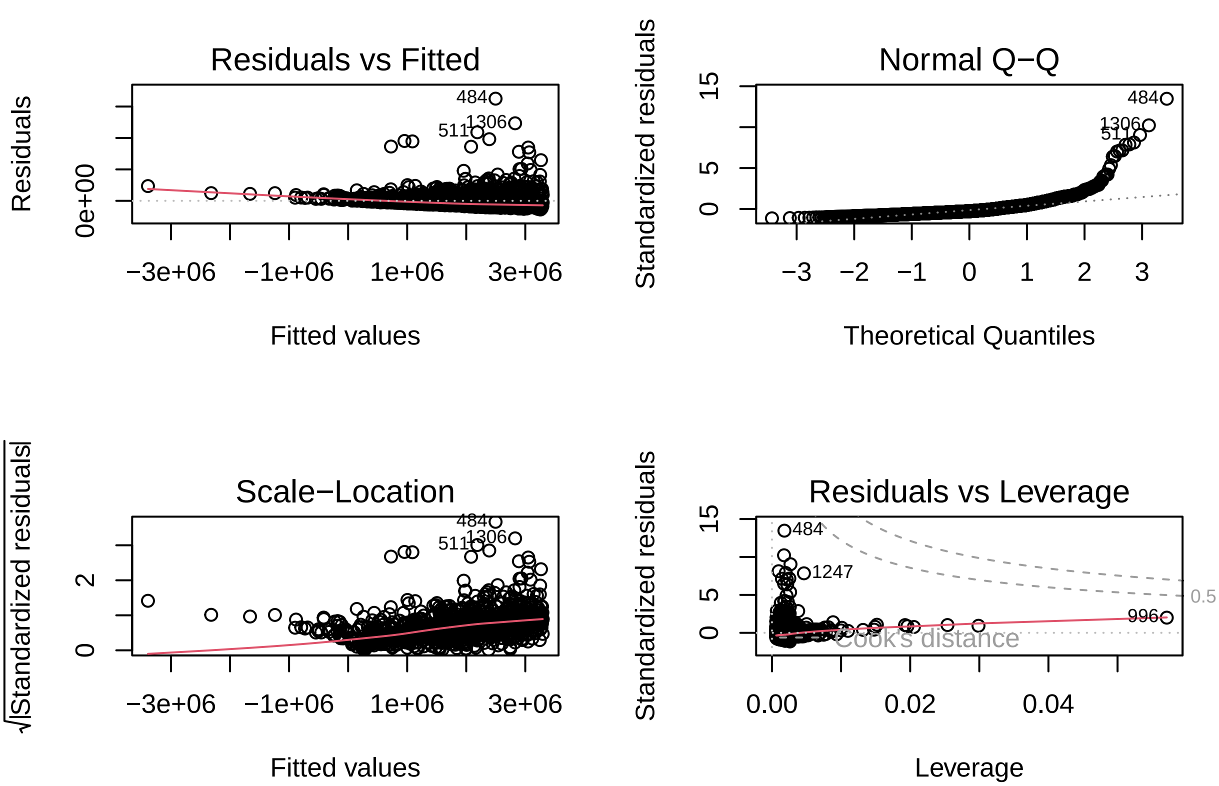
The residual plots will help determine how well a model will represent the data from this dataset. The residuals vs fitted plot model will reveal any non-linear patterns from the residuals, which in this data set, the plots are mostly declining and bunched up on the far right of the graph, similar to a negative linear association. The normal Q-Q shows whether the residuals deviate much or very little, which in this data does not deviate much in the beginning and curves upward halfway, meanning it deviates sharply at some point. Scale-location shows if the residuals are distributed equally along the range of the predictors, so in this dataset, the values are not spread as equally along the predictors. The last plot, residual vs. leverage, indicating whether if there are any outliers in the data present, which for this dataset there are only a few observations that have a large distance between the rest of the data but still within the Cook's distance lines.

Building a multiple linear regression model

```r
lmforecast <- lm(Price~Year + Kilometer, data = train)
summary(lmforecast)
```

```
##
## Call:
## lm(formula = Price ~ Year + Kilometer, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2743701 -1199596  -636806   533249 32503121
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.479e+08  4.234e+07  -8.216 4.20e-16 ***
## Year         1.737e+05  2.096e+04   8.284 2.44e-16 ***
## Kilometer   -1.100e+01  2.258e+00  -4.871 1.22e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2416000 on 1644 degrees of freedom
## Multiple R-squared:  0.1062, Adjusted R-squared:  0.1052
## F-statistic: 97.71 on 2 and 1644 DF,  p-value: < 2.2e-16
```

```r
par(mfrow=c(2,2))
plot(lmforecast)
```

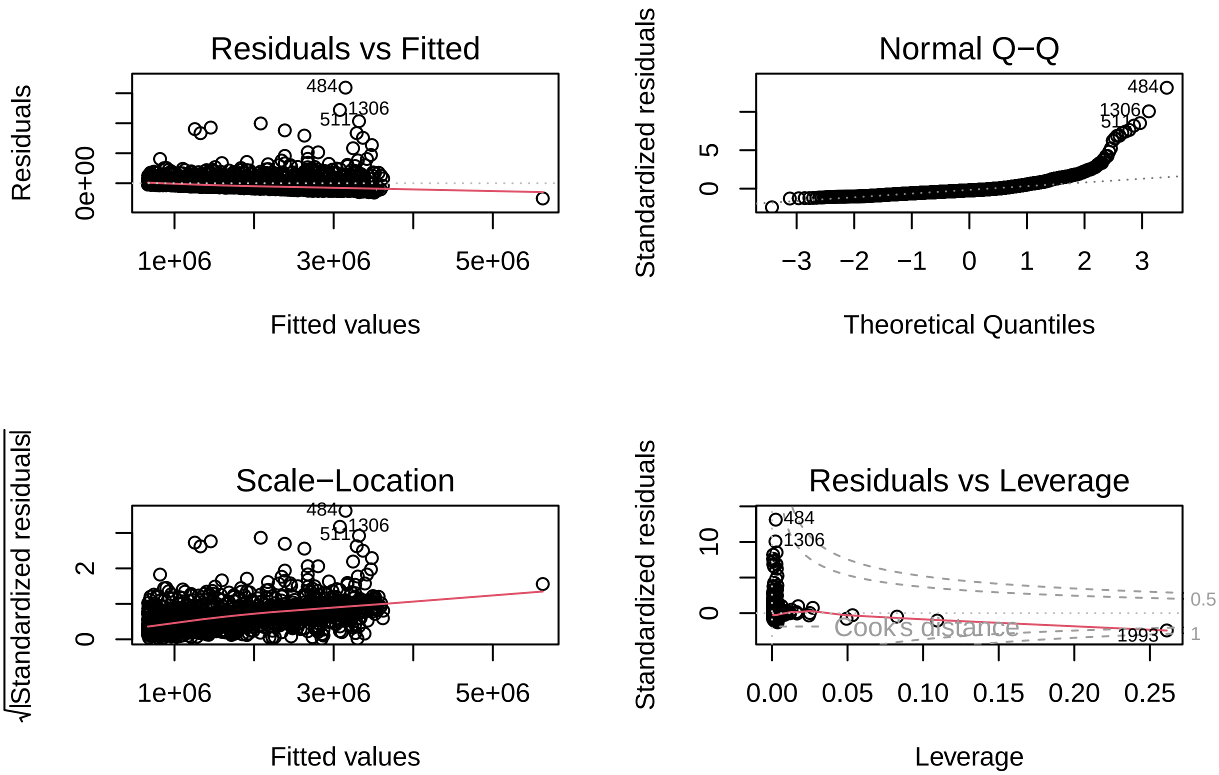


Build out a third linear regression model to try and improve results

```
lmPoly <- lm(Price~Kilometer + I(Kilometer^2), data = train)
summary(lmPoly)
```

```
##
## Call:
## lm(formula = Price ~ Kilometer + I(Kilometer^2), data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -5072493 -1178650  -545017   400742 31850845
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     3.622e+06  1.575e+05  22.996  < 2e-16 ***
## Kilometer      -5.194e+01  4.733e+00 -10.974  < 2e-16 ***
## I(Kilometer^2)  2.282e-04  3.225e-05   7.077 2.17e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2429000 on 1644 degrees of freedom
## Multiple R-squared:  0.09646,    Adjusted R-squared:  0.09536
## F-statistic: 87.75 on 2 and 1644 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(2,2))
plot(lmPoly)
```



Write a paragraph or more comparing the results. Indicate which model is better and why you think that is the case.

All three models illustrated the patterns and distribution of data observations in the dataset used for the single, multiple, and polynomial linear regression models. Comparing all three models based on the plots produced and the values from the coefficients, residual standard error, r-squared, and F-statistic, the third model is the one that gave the better results. The third model, which is the polynomial linear regression model, has a lower r-squared value in comparison to the other models, and low p-value and low residual value. However, even with the lower values, there isn't much improvement in all three plots.

Predict and evaluate with test data

```
pred1 <- predict(lmPrice, newdata = test)
pred1 <- exp(pred1)
cor1 <- cor(pred1, test$Price)
mse1 <- mean((pred1-test$Price)^2)
rmse1 <- sqrt(mse1)

print(paste("correlation: ", cor1))
```

```
## [1] "correlation:  NaN"
```

```
print(paste("mse: ", mse1))
```

```
## [1] "mse:  Inf"
```

```
print(paste("rse: ", rmse1))
```

```
## [1] "rse:  Inf"
```

```
pred2 <- predict(lmforecast, newdata = test)
pred2 <- exp(pred2)
cor2 <- cor(pred2, test$Kilometer)
mse2 <- mean((pred2-test$Kilometer)^2)
rmse2 <- sqrt(mse2)

print(paste("correlation: ", cor2))
```

```
## [1] "correlation:  NaN"
```

```
print(paste("mse: ", mse2))
```

```
## [1] "mse:  Inf"
```

```
print(paste("rse: ", rmse2))
```

```
## [1] "rse:  Inf"
```

```
pred3 <- predict(lmPoly, newdata = test)
pred3 <- exp(pred3)
cor3 <- cor(pred3, test$Price)
mse3 <- mean((pred3-test$Price)^2)
rmse3 <- sqrt(mse3)

print(paste("correlation: ", cor3))
```

```
## [1] "correlation:  NaN"
```

```
print(paste("mse: ", mse3))
```

```
## [1] "mse:  Inf"
```

```
print(paste("rse: ", rmse3))
```

```
## [1] "rse:  Inf"
```

The results returned from the prediction and evaluation of the three models show that this dataset has some issues in the data itself, considering it is returning NaN and Inf values for correlation and mse and rse. For correlation, this indicates a calculation done in the prediction and evaluation tests. For the mse and rse, there might have been some values present that cause the mse and rse to run infinitely. Overall, its possible that there might be issues in the method of collecting data for the set.