Ogólny przepis na budowanie modeli

> Dobry chrześcijanin powinien wystrzegać się matematyków i tych wszystkich, którzy tworzą puste proroctwa. Istnieje niebezpieczeństwo, że matematycy zawarli przymierze z diabłem, aby zgubić duszę człowieka i wtrącić go w odmęty piekieł. – św. Augustyn

Co prawda św. Augustyn nie miał raczej na myśli matematyków w dzisiejszym rozumieniu, a raczej astrologów, ale weźmiemy sobie do serca ostrzeżenie o „pustych proroctwach".

Zauważmy, że do tej pory nasze postępowanie było następujące:

1. Budujemy model w oparciu o wszystkie dostępne dane
2. Sprawdzamy czy model został zbudowany poprawnie (diagnostyka)
3. Ewentualnie naprawiamy model

Poznaliśmy też jedną miarę dobroci dopasowania - R^2. Zobaczyliśmy, że nie jest ona idealna, im więcej zmiennych „wrzucimy" do modelu, tym wyższe będzie R^2. W zagadnieniu analizy wariancji poznaliśmy metodę porównywania modeli zagnieżdżonych.

Jaki jest problem z R^2 lub jakąkolwiek inną miarą? Oceniamy dopasowanie danych do modelu, który został zbudowany w oparciu o nie. Zależy nam na tym, żeby model był uniwersalny, i dawał dobrą predykcję na nowych danych. Każdy model ma tendencję do „przeuczania", to znaczy lepiej wypada na danych, które „widział" niż na nowych. To trochę jak kolokwium z zadań z list, coś mierzy, ale nie jest dobrą miarą wiedzy jaką opanowali studenci. Podsumowując, aby ocenić przydatność modelu musimy go ocenić na innym, nowym zbiorze danych.

Generalna zasada, za książką Hadelya Wickhama, jest następująca:

> Each observation can either be used for exploration or confirmation, not both.

Eksploracja, czyli przyglądanie się danym, budowanie modeli, transformacje zmiennych itd. Potwierdzenie (confirmation) to jedynie ocena jakości końcowego modelu. W momencie kiedy zostanie to zbadane, nasze szukanie i budowa modelu MUSZĄ się zakończyć. W przeciwnym razie obserwacja stała by się „esploracyjna".

Podział danych na 3 grupy

Poniżej zamieszczam ogólnie przyjętą praktykę. Nie jest ona jedyna możliwa, ale jest sensowna i warto jej się trzymać póki nie zdobędzie się więcej doświadczenia. Pochodzi z książki R for Data Science.

Przed rozpoczęciem analizy dzielimy zbiór danych na trzy części:

- 60% danych stanowi zbiór treningowy - czyli taki, na którym możemy robić wszystko. Od wizualizacji do budowy dowolnej liczby modeli
- 20% danych stanowi zbiór walidacyjny. Na tej części danych możemy porównywać między sobą modele, ale nie tworzymy nowych modeli w oparciu o te dane. Na podstawie porównania modeli na zbiorze walidacyjnym wybieramy ostateczny model
- 20% danych stanowi zbiór testowy. Można go użyć tylko raz, w celu sprawdzenia dokładności wybranego, ostatecznego modelu

Przykład - ceny diamentów

```
?diamonds
data(diamonds)
str(diamonds)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    53940 obs. of  10 variables:
##  $ carat  : num  0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
##  $ cut    : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 2 4 2 3 3 3 1 3 ...
```

```
##  $ color  : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 6 5 2 5 ...
##  $ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 3 5 4 2 6 7 3 4 5 ...
##  $ depth  : num  61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
##  $ table  : num  55 61 65 58 58 57 57 55 61 61 ...
##  $ price  : int  326 326 327 334 335 336 336 337 337 338 ...
##  $ x      : num  3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
##  $ y      : num  3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
##  $ z      : num  2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
```

```
summary(diamonds)
```

```
##      carat               cut        color        clarity
##  Min.   :0.2000   Fair      : 1610   D: 6775   SI1    :13065
##  1st Qu.:0.4000   Good      : 4906   E: 9797   VS2    :12258
##  Median :0.7000   Very Good:12082   F: 9542   SI2    : 9194
##  Mean   :0.7979   Premium   :13791   G:11292   VS1    : 8171
##  3rd Qu.:1.0400   Ideal     :21551   H: 8304   VVS2   : 5066
##  Max.   :5.0100                     I: 5422   VVS1   : 3655
##                                     J: 2808   (Other): 2531
##      depth           table           price             x
##  Min.   :43.00   Min.   :43.00   Min.   :  326   Min.   : 0.000
##  1st Qu.:61.00   1st Qu.:56.00   1st Qu.:  950   1st Qu.: 4.710
##  Median :61.80   Median :57.00   Median : 2401   Median : 5.700
##  Mean   :61.75   Mean   :57.46   Mean   : 3933   Mean   : 5.731
##  3rd Qu.:62.50   3rd Qu.:59.00   3rd Qu.: 5324   3rd Qu.: 6.540
##  Max.   :79.00   Max.   :95.00   Max.   :18823   Max.   :10.740
##
##        y               z
##  Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 4.720   1st Qu.: 2.910
##  Median : 5.710   Median : 3.530
##  Mean   : 5.735   Mean   : 3.539
##  3rd Qu.: 6.540   3rd Qu.: 4.040
##  Max.   :58.900   Max.   :31.800
##
```

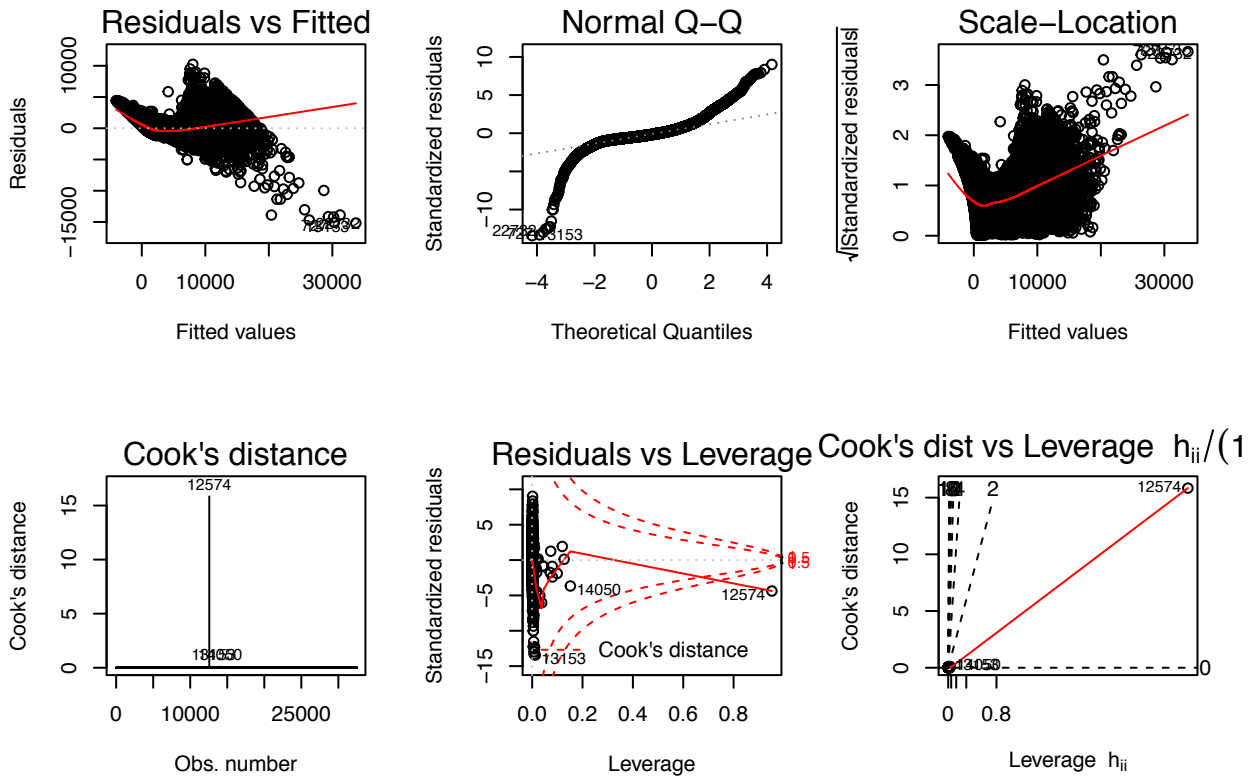
```
set.seed(23) #MEGA WAŻNE!!!!!
train=sample(1:nrow(diamonds), floor(0.6 * nrow(diamonds)))
diamonds_train=diamonds[train,]
diamonds_rest=diamonds[-train,]
query=sample(1:nrow(diamonds_rest), floor(0.5 * nrow(diamonds_rest)))
diamonds_query=diamonds_rest[query,]
diamonds_test=diamonds_rest[-query,]
```

```
diamonds.lm=lm(price~., data=diamonds_train)
summary(diamonds.lm)
```

```
##
## Call:
## lm(formula = price ~ ., data = diamonds_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15169.3   -595.0   -185.2    384.4  10239.7
##
## Coefficients:
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4565.013    570.982   7.995 1.34e-15 ***
## carat       11084.635     61.934 178.975  < 2e-16 ***
## cut.L         585.492     29.578  19.795  < 2e-16 ***
## cut.Q        -289.646     23.729 -12.207  < 2e-16 ***
## cut.C         148.433     20.194   7.350 2.02e-13 ***
## cut^4         -14.883     16.109  -0.924 0.355553
## color.L     -1937.173     22.543 -85.931  < 2e-16 ***
## color.Q      -671.164     20.498 -32.742  < 2e-16 ***
## color.C      -171.017     19.148  -8.931  < 2e-16 ***
## color^4        20.490     17.572   1.166 0.243599
## color^5      -105.283     16.614  -6.337 2.38e-10 ***
## color^6       -54.608     15.084  -3.620 0.000295 ***
## clarity.L    4181.220     39.083 106.983  < 2e-16 ***
## clarity.Q   -2015.426     36.398 -55.372  < 2e-16 ***
## clarity.C    1026.686     31.167  32.941  < 2e-16 ***
## clarity^4    -422.542     24.913 -16.961  < 2e-16 ***
## clarity^5     246.397     20.412  12.071  < 2e-16 ***
## clarity^6      -3.344     17.829  -0.188 0.851229
## clarity^7     103.952     15.730   6.608 3.94e-11 ***
## depth         -52.749      7.086  -7.444 9.97e-14 ***
## table         -22.396      3.805  -5.887 3.98e-09 ***
## x            -855.263     53.944 -15.855  < 2e-16 ***
## y               7.331     22.546   0.325 0.745077
## z            -184.145     78.493  -2.346 0.018982 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1137 on 32340 degrees of freedom
## Multiple R-squared:  0.9193, Adjusted R-squared:  0.9192
## F-statistic: 1.602e+04 on 23 and 32340 DF,  p-value: < 2.2e-16
```

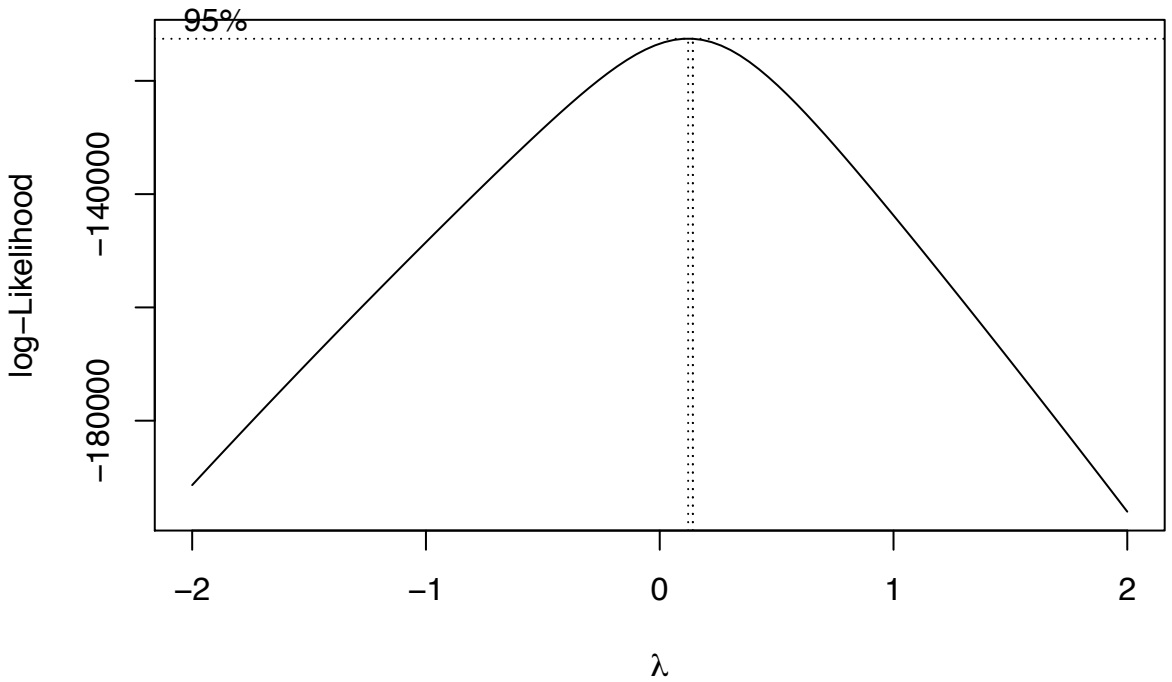
```
par(mfrow=c(2,3))
plot(diamonds.lm, 1:6)
```




```
par(mfrow=c(1,1))
```

So we fix the model

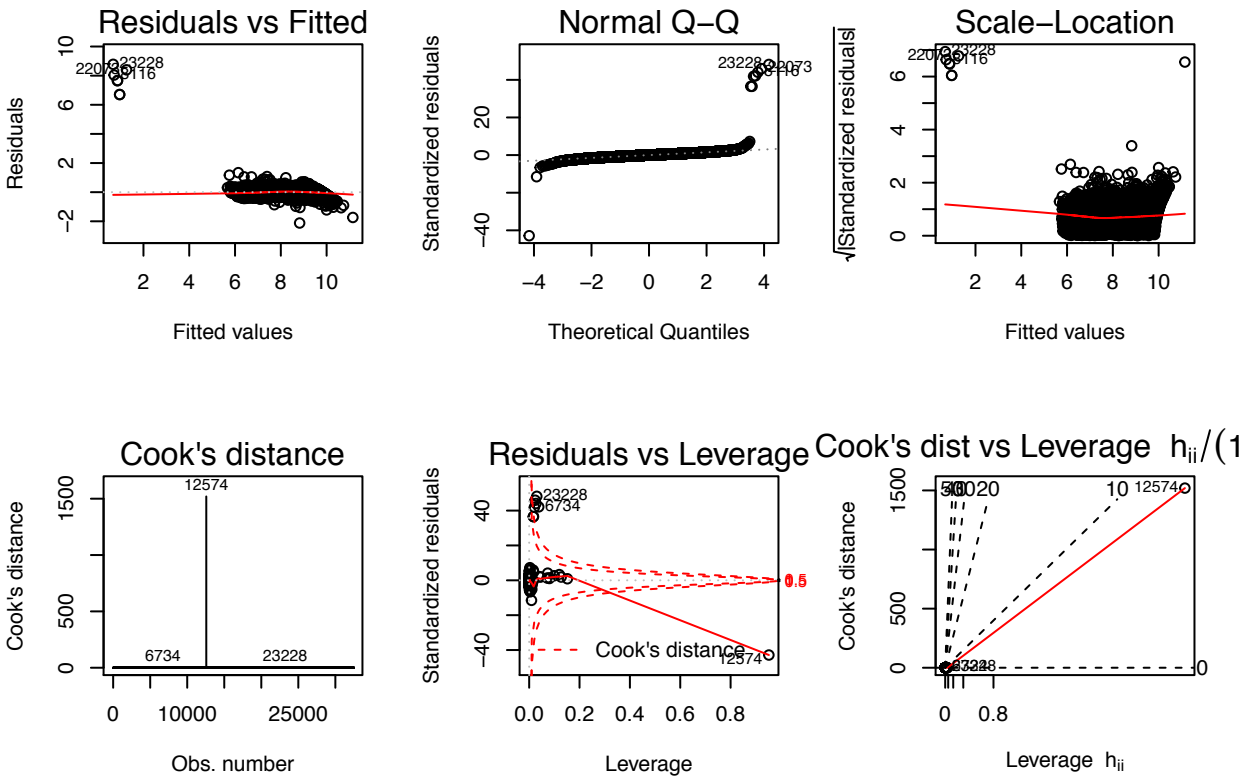
```
MASS::boxcox(diamonds.lm)
```




```
diamonds_log.lm=lm(log(price)~., data=diamonds_train)
summary(diamonds_log.lm)
```

```
##
## Call:
## lm(formula = log(price) ~ ., data = diamonds_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.1183 -0.0935  0.0026  0.0916  8.7796
##
## Coefficients:
##               Estimate Std. Error  t value Pr(>|t|)
## (Intercept) -2.0791333  0.0929477  -22.369  < 2e-16 ***
## carat       -0.5222847  0.0100820  -51.804  < 2e-16 ***
## cut.L        0.0990528  0.0048149   20.572  < 2e-16 ***
## cut.Q       -0.0305458  0.0038627   -7.908 2.70e-15 ***
## cut.C        0.0361554  0.0032873   10.999  < 2e-16 ***
## cut^4        0.0114273  0.0026223    4.358 1.32e-05 ***
## color.L     -0.4506654  0.0036698 -122.805  < 2e-16 ***
## color.Q     -0.1063451  0.0033368  -31.870  < 2e-16 ***
## color.C     -0.0094749  0.0031171   -3.040 0.002370 **
## color^4      0.0182231  0.0028604    6.371 1.91e-10 ***
## color^5     -0.0098414  0.0027046   -3.639 0.000274 ***
## color^6      0.0005404  0.0024554    0.220 0.825795
## clarity.L    0.8878973  0.0063621  139.560  < 2e-16 ***
## clarity.Q   -0.2713668  0.0059250  -45.800  < 2e-16 ***
## clarity.C    0.1472104  0.0050736   29.015  < 2e-16 ***
## clarity^4   -0.0701374  0.0040554  -17.295  < 2e-16 ***
## clarity^5    0.0298673  0.0033228    8.989  < 2e-16 ***
## clarity^6   -0.0063753  0.0029023   -2.197 0.028053 *
## clarity^7    0.0302420  0.0025606   11.810  < 2e-16 ***
## depth        0.0465526  0.0011534   40.360  < 2e-16 ***
## table        0.0095099  0.0006193   15.355  < 2e-16 ***
## x            1.0879618  0.0087814  123.894  < 2e-16 ***
## y            0.0270715  0.0036702    7.376 1.67e-13 ***
## z            0.1104964  0.0127775    8.648  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.185 on 32340 degrees of freedom
## Multiple R-squared:  0.9669, Adjusted R-squared:  0.9669
## F-statistic: 4.105e+04 on 23 and 32340 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(2,3))
plot(diamonds_log.lm, 1:6)
```




```
par(mfrow=c(1,1))
```

Nadal czeka nas sporo naprawiania:

```
outliers=which(residuals(diamonds_log.lm)>5)
big_cook_distance=which(cooks.distance(diamonds_log.lm)>1)
observations_to_exclude=union(outliers, big_cook_distance)
diamonds_train[observations_to_exclude,]
```

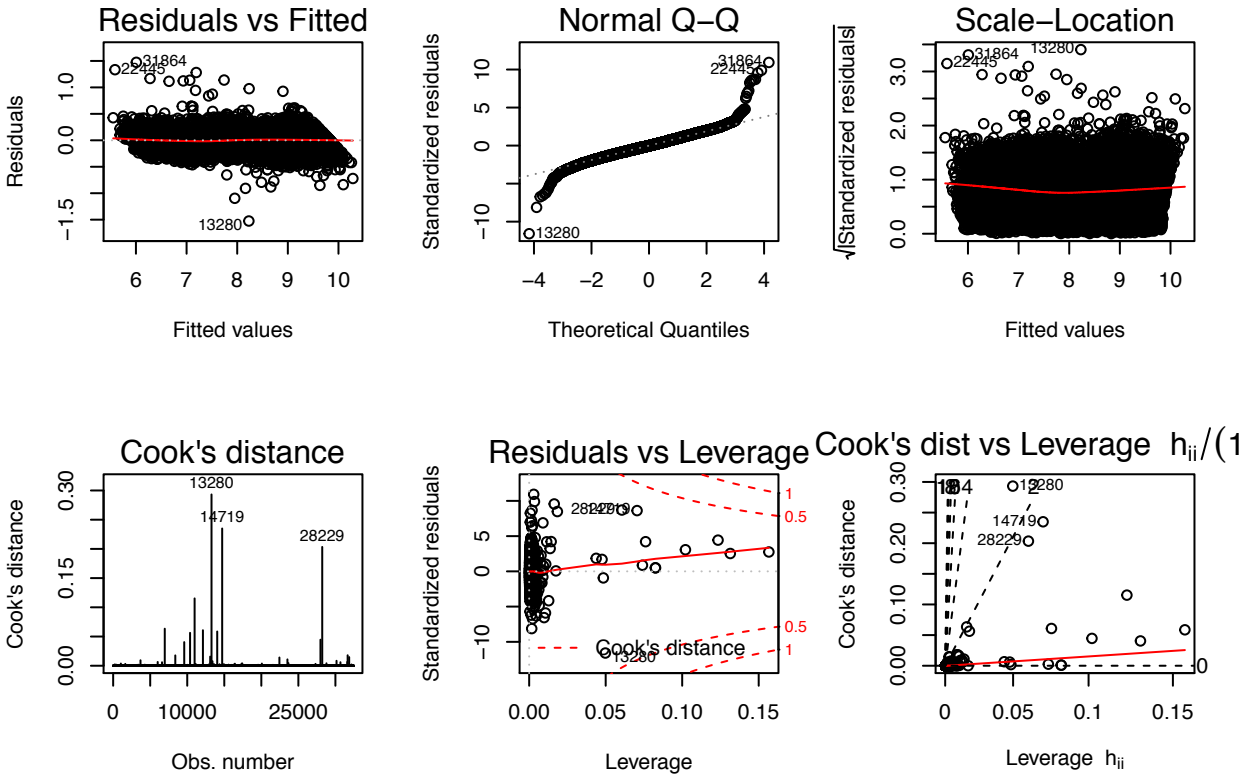
```
## # A tibble: 8 × 10
##   carat       cut color clarity depth table price     x     y     z
##   <dbl>     <ord> <ord>   <ord> <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1  0.71      Good     F     SI2  64.1    60  2130  0.00  0.00  0.00
## 2  1.07     Ideal     F     SI2  61.6    56  4954  0.00  6.62  0.00
## 3  1.14      Fair     G     VS1  57.5    67  6381  0.00  0.00  0.00
## 4  1.00 Very Good     H     VS2  63.3    53  5139  0.00  0.00  0.00
## 5  1.20   Premium     D    VVS1  62.1    59 15686  0.00  0.00  0.00
## 6  1.56     Ideal     G     VS2  62.2    54 12800  0.00  0.00  0.00
## 7  0.71      Good     F     SI2  64.1    60  2130  0.00  0.00  0.00
## 8  2.00   Premium     H     SI2  58.9    57 12210  8.09 58.90  8.06
```

```
diamonds_log.lm2=lm(log(price)~., data=diamonds_train, subset = -observations_to_exclude)
summary(diamonds_log.lm2)
```

```
##
## Call:
## lm(formula = log(price) ~ ., data = diamonds_train, subset = -observations_to_exclude)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.52570 -0.08650 -0.00178  0.08542  1.47426
```

```
##
## Coefficients:
##               Estimate Std. Error  t value Pr(>|t|)
## (Intercept) -3.8064232  0.0694514  -54.807  < 2e-16 ***
## carat       -1.0317947  0.0079594 -129.632  < 2e-16 ***
## cut.L        0.1028447  0.0035266   29.162  < 2e-16 ***
## cut.Q       -0.0240374  0.0028629   -8.396  < 2e-16 ***
## cut.C        0.0149211  0.0024895    5.994 2.07e-09 ***
## cut^4        0.0014355  0.0019395    0.740   0.4592
## color.L     -0.4388962  0.0026796 -163.793  < 2e-16 ***
## color.Q     -0.0954062  0.0024367  -39.154  < 2e-16 ***
## color.C     -0.0112104  0.0022753   -4.927 8.39e-07 ***
## color^4      0.0130232  0.0020880    6.237 4.51e-10 ***
## color^5     -0.0039281  0.0019748   -1.989   0.0467 *
## color^6      0.0003685  0.0017926    0.206   0.8371
## clarity.L    0.8947462  0.0046549  192.214  < 2e-16 ***
## clarity.Q   -0.2397231  0.0043303  -55.359  < 2e-16 ***
## clarity.C    0.1268424  0.0037066   34.221  < 2e-16 ***
## clarity^4   -0.0621832  0.0029608  -21.002  < 2e-16 ***
## clarity^5    0.0230144  0.0024264    9.485  < 2e-16 ***
## clarity^6   -0.0035443  0.0021186   -1.673   0.0943 .
## clarity^7    0.0340161  0.0018692   18.198  < 2e-16 ***
## depth        0.0597515  0.0008593   69.532  < 2e-16 ***
## table        0.0103390  0.0004529   22.829  < 2e-16 ***
## x            0.8390045  0.0145295   57.745  < 2e-16 ***
## y            0.5154088  0.0145254   35.483  < 2e-16 ***
## z            0.0841846  0.0094907    8.870  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1351 on 32332 degrees of freedom
## Multiple R-squared:  0.9824, Adjusted R-squared:  0.9823
## F-statistic: 7.826e+04 on 23 and 32332 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(2,3))
plot(diamonds_log.lm2, 1:6)
```




```
par(mfrow=c(1,1))
```

> Now it would be very remarkable if any system existing in the real world could be exactly represented by any simple model. However, cunningly chosen parsimonious models often do provide remarkably useful approximations. For example, the law PV = RT relating pressure P, volume V and temperature T of an "ideal" gas via a constant R is not exactly true for any real gas, but it frequently provides a useful approximation and furthermore its structure is informative since it springs from a physical view of the behavior of gas molecules. For such a model there is no need to ask the question "Is the model true?". If "truth" is to be the "whole truth" the answer must be "No". The only question of interest is "Is the model illuminating and useful?".

Zbudowaliśmy dwa modele. Jeden niepoprawny (ze względu na diagnostykę, którą przeprowadziliśmy), a drugi wymuskany i pięknie zbudowany. Który jest lepszy? Zobaczmy jaką mają predykcję.

```
query_residuals_lm=diamonds_query$price-predict(diamonds.lm, diamonds_query)
query_residuals_log_lm=diamonds_query$price-exp(predict(diamonds_log.lm, diamonds_query))
query_residuals_log_lm_no_influential=diamonds_query$price-exp(predict(diamonds_log.lm2, diamonds_query
```

Porównajmu R^2 na zbiorze treningowym i walidacyjnym

```
#liniowy, bez log
summary(diamonds.lm)$r.squared
```

```
## [1] 0.9192896
```

```
1-var(query_residuals_lm)/var(diamonds_query$price)
```

```
## [1] 0.9225783
```

```
#liniowy, bez usuwania
summary(diamonds_log.lm)$r.squared
```

```
## [1] 0.9668812
```

```
1-var(query_residuals_log_lm)/var(diamonds_query$price)
```

```
## [1] 0.9220435
```

```
#liniowy, wyrzucone wplywowe
summary(diamonds_log.lm2)$r.squared
```

```
## [1] 0.9823555
```

```
1-var(query_residuals_log_lm_no_influential)/var(diamonds_query$price)
```

```
## [1] 0.9554848
```

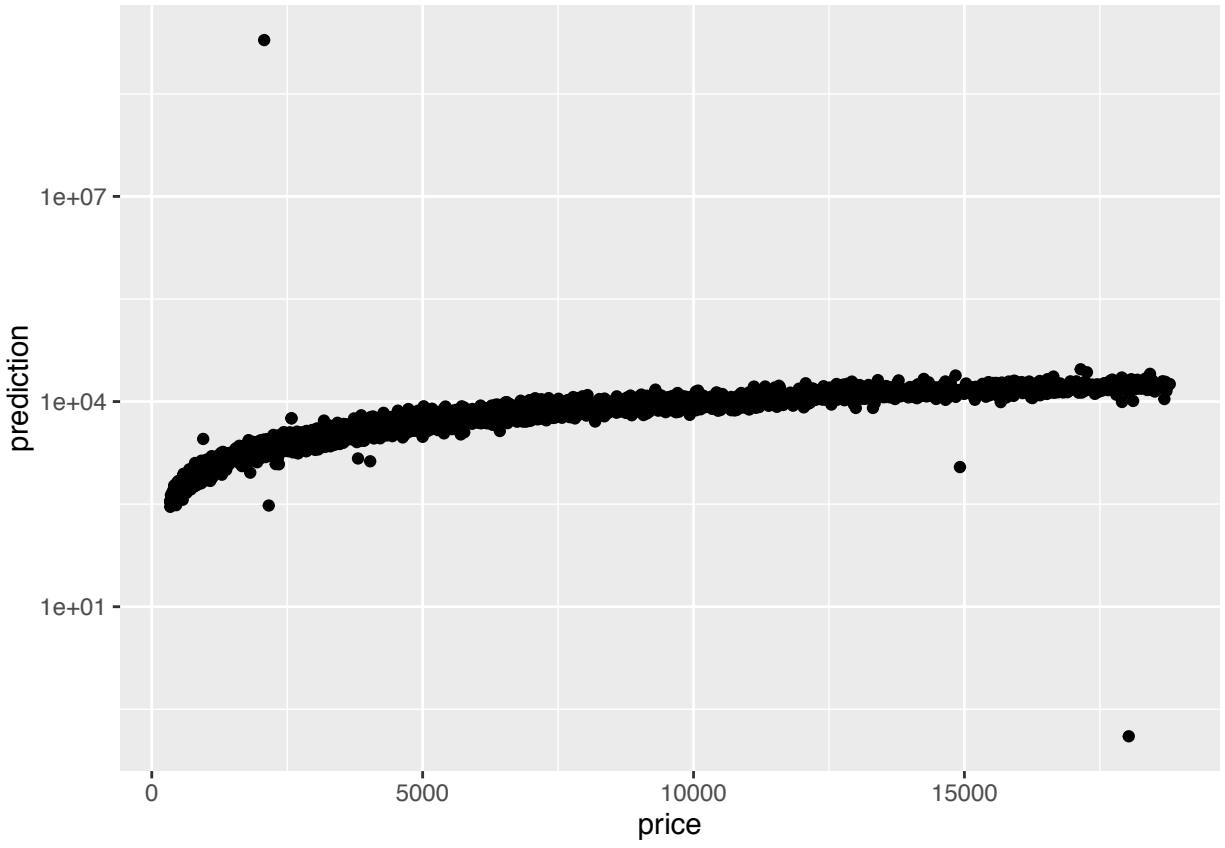
Wybieramy zatem model, zbudowany na logarytmie, bez obserwacji wpływowych. Zbadajmy na koniec jaka jest jego dokładność

```
test_residuals=diamonds_test$price-exp(predict(diamonds_log.lm2, diamonds_test))
1-var(test_residuals)/var(diamonds_test$price)
```

```
## [1] -21788847
```

To jest bardzo złe! Czemu tak się wydarzyło?

```
data.frame(price=diamonds_test$price,
           prediction=exp(predict(diamonds_log.lm2, diamonds_test))) %>%
  ggplot(aes(x=price, y=prediction)) +
  geom_point() + scale_y_log10()
```




```
bad_prediction=which.min(test_residuals)
predict(diamonds_log.lm2, diamonds_test)[c(bad_prediction-1,bad_prediction, bad_prediction+1)]
```

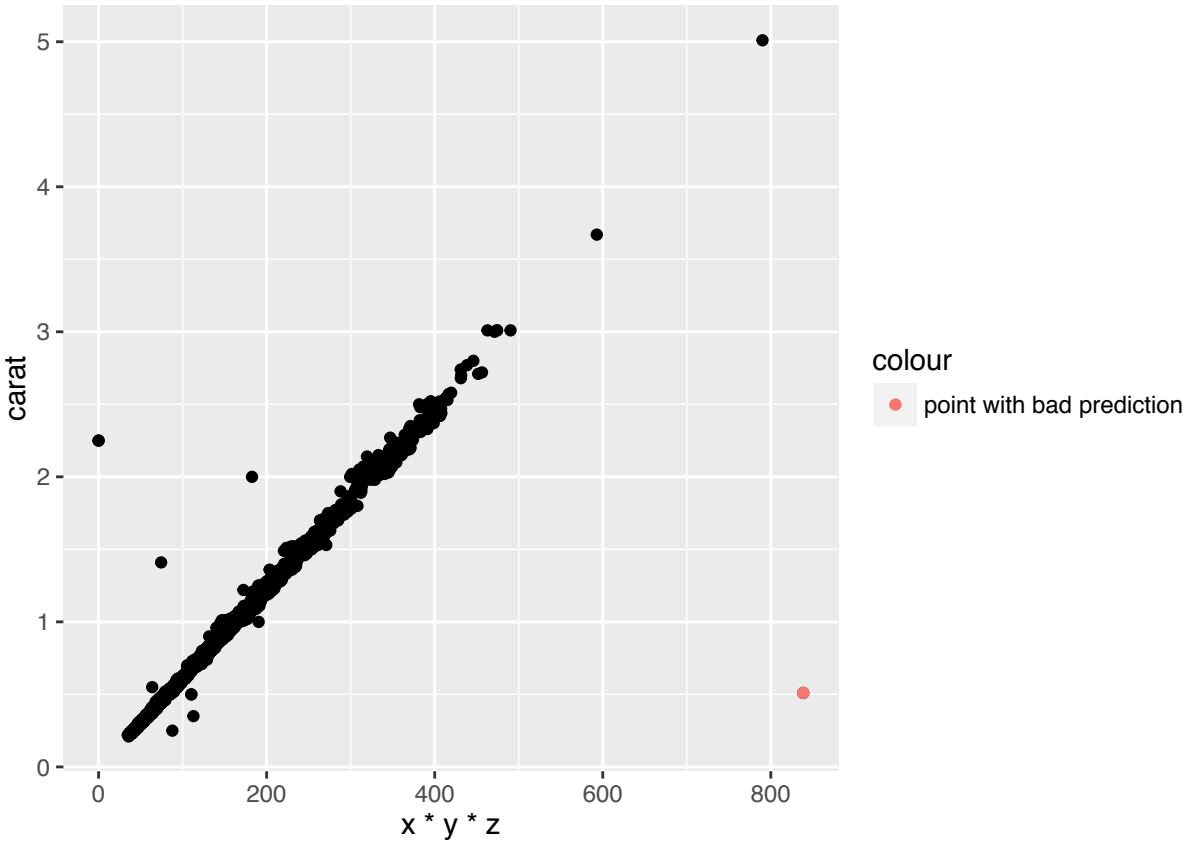
```
##      9796      9797      9798
##  7.663317 21.383501  7.692571
```

```
diamonds_test[c(bad_prediction-1,bad_prediction, bad_prediction+1),]
```

```
## # A tibble: 3 × 10
##   carat   cut color clarity depth table price     x     y     z
##   <dbl> <ord> <ord>   <ord> <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1  0.72 Ideal     F     SI2  61.6    57  2074  5.75  5.68  3.52
## 2  0.51 Ideal     E     VS1  61.8    55  2075  5.15 31.80  5.12
## 3  0.72 Ideal     G     SI2  62.1    54  2079  5.77  5.82  3.60
```

Ilość karatów powinna zależeć od wymiarów diamentu!

```
bad_df=diamonds_test[bad_prediction,]
ggplot(diamonds_test, aes(x=x*y*z, y=carat)) +
  geom_point() +
  geom_point(data = bad_df, aes(color="point with bad prediction"))
```



Można by zmienić 31.80 na 3.18, ale lepiej jest usunać obserwację (nie wiadomo na pewno, co się popsuło).

```
test_residuals=diamonds_test$price-exp(predict(diamonds_log.lm2, diamonds_test))
1-var(test_residuals[-9797])/var(diamonds_test$price[-9797])
```

```
## [1] 0.9560341
```

Referencje:

http://r4ds.had.co.nz/model-intro.html

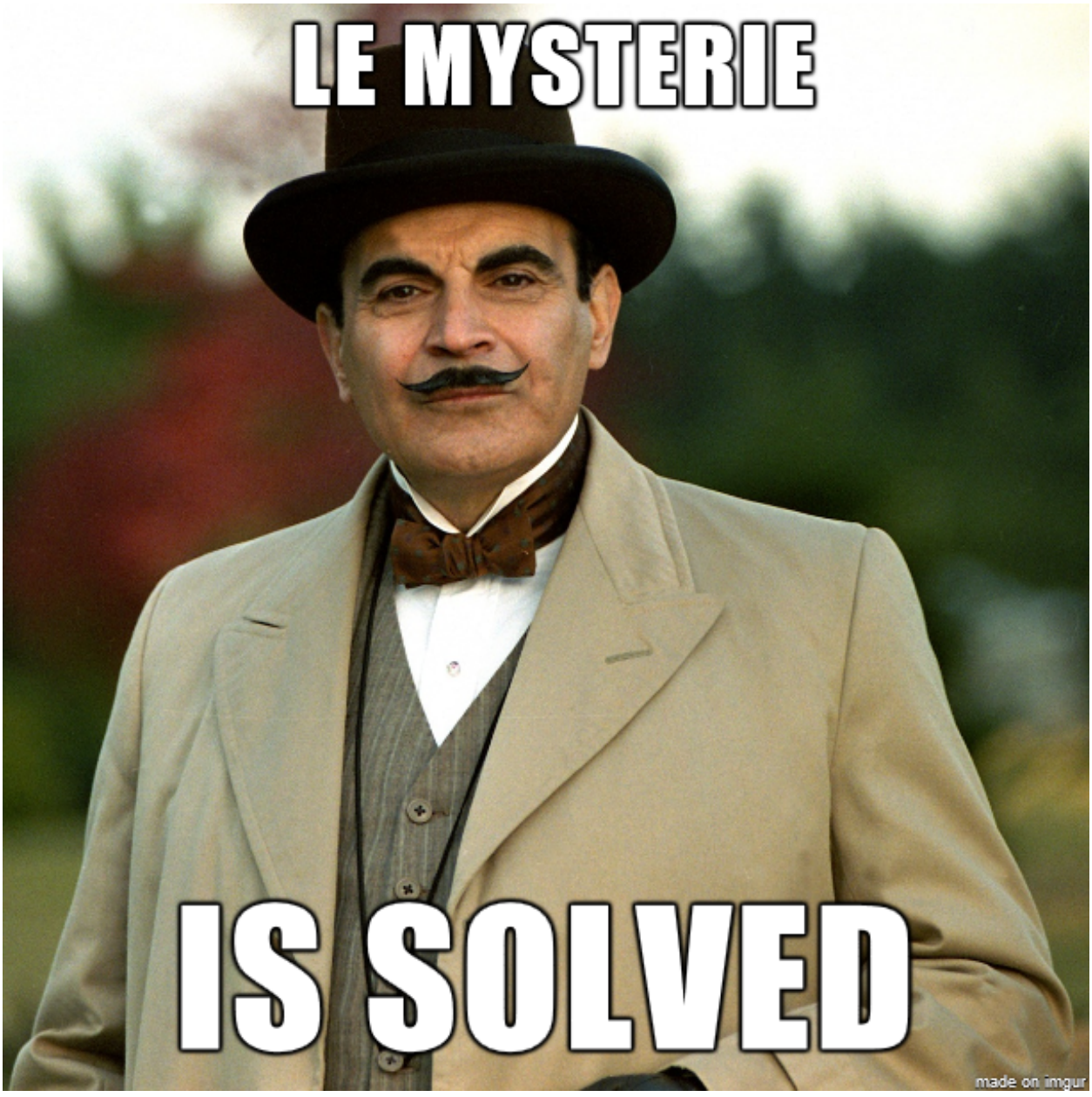


Figure 1: