Supervised Learning - Regression

Fabio Carvalho Lima

25/01/2020

Predicting medical expenses using linear regression

Introduction

The goal of this analysis is to use patient data do forecast the average medical care expenses for such population segments. These estimates could be used to create actuarial tables that set the price of yearly premiuns higher or lower according to the expected treatment costs.

For this analysis, we will use simulated dataset containing hypothetical medical expenses for patients in the United States.

The *insurance.csv* file has examples of beneficiaries currently enrolled in the insurance plan, with features indicating characteristics of the patient as well as the total medical expenses charged to the plan for the calendar year. The features are:

- age: An integer indicating the age of the primary beneficiary (excluding those above 64 years, as they are generally covered by the government).
- sex: The policy holder's gender: either male or female.
- bmi: The body mass index (BMI), which provides a sense of how over or underweight a person is relative to their height. BMI is equal to weight (in kilograms) divided by height (in meters) squared. An ideal BMI is within the range of 18.5 to 24.9.
- children: An integer indicating the number of children/dependents covered by the insurance plan.
- smoker: A yes or no categorical variable that indicates whether the insured regularly smokes tobacco.
- region. The beneficiary's place of residence in the US, divided into four geographic regions: northeast, southeast, southwest, or northwest.

It is important to give some thought to how these variables may relate to billed medical expenses. For example, we might expect that older people, smokers and with high bmi are at higher risk of large medical expenses.

Exploring and preparing the data

```r
options(digits = 3)
# read the data and inspect the 6 first lines
insurance <- read.csv("./data/insurance.csv", stringsAsFactors = TRUE)
head(insurance)
```

```
##   age    sex  bmi children smoker    region expenses
## 1  19 female 27.9        0    yes southwest    16885
## 2  18   male 33.8        1     no southeast     1726
## 3  28   male 33.0        3     no southeast     4449
## 4  33   male 22.7        0     no northwest    21984
## 5  32   male 28.9        0     no northwest     3867
## 6  31 female 25.7        0     no southeast     3757
```

```
str(insurance)
```

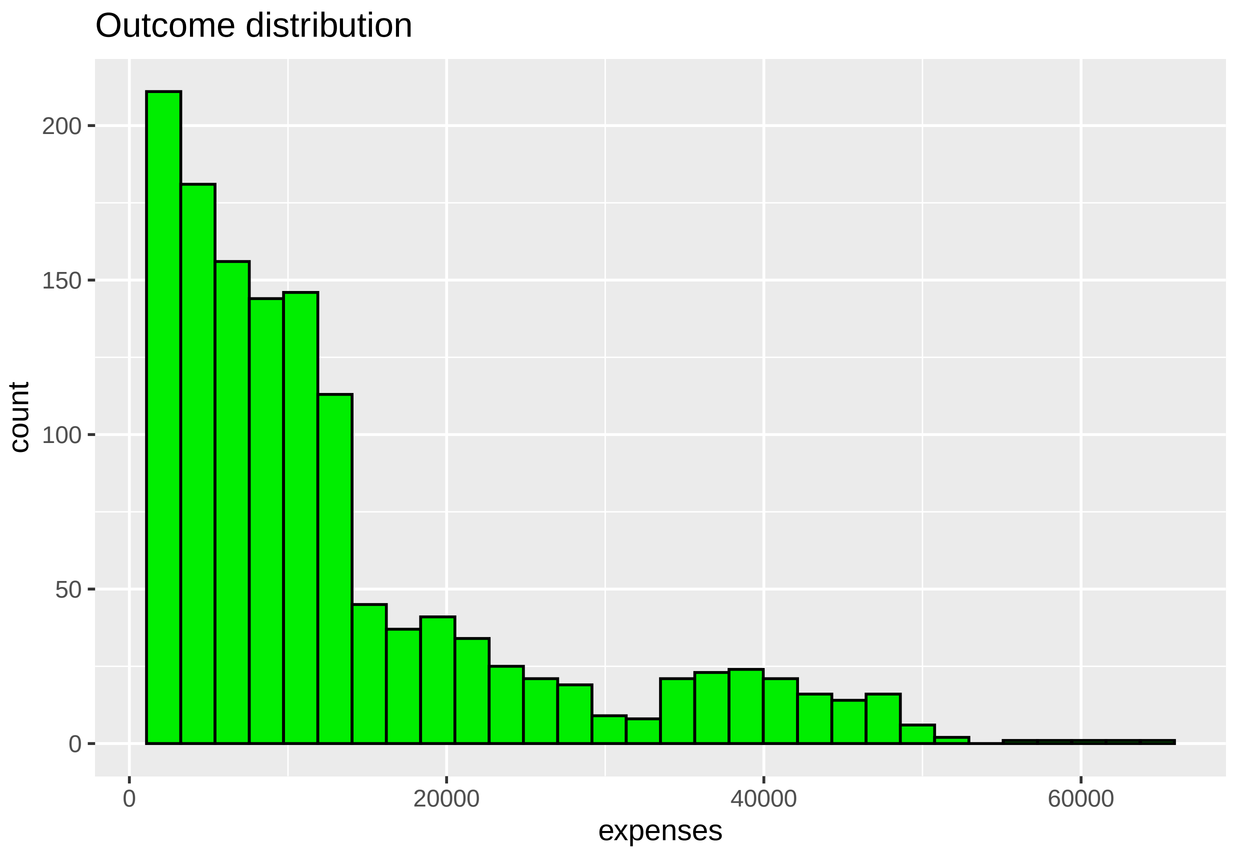
```
## 'data.frame':    1338 obs. of  7 variables:
##  $ age     : int  19 18 28 33 32 31 46 37 37 60 ...
##  $ sex     : Factor w/ 2 levels "female","male": 1 2 2 2 2 1 1 1 2 1 ...
##  $ bmi     : num  27.9 33.8 33 22.7 28.9 25.7 33.4 27.7 29.8 25.8 ...
##  $ children: int  0 1 3 0 0 0 1 3 2 0 ...
##  $ smoker  : Factor w/ 2 levels "no","yes": 2 1 1 1 1 1 1 1 1 1 ...
##  $ region  : Factor w/ 4 levels "northeast","northwest",..: 4 3 3 2 2 3 3 2 1 2 ...
##  $ expenses: num  16885 1726 4449 21984 3867 ...
```

Let's check for missing values:

```
sapply(insurance, function(x) sum(is.na(x)))
```

```
##      age      sex      bmi children   smoker   region expenses
##        0        0        0        0        0        0        0
```

Our model's dependent variable is **expense**, which measures the medical costs each person charged to the insurance plan for a year. Prior to building a multiple regression model, it is often helpful to check for normality. Although linear regression does not stricly require a normally distributed dependent variable, the model often fits better when this is true.

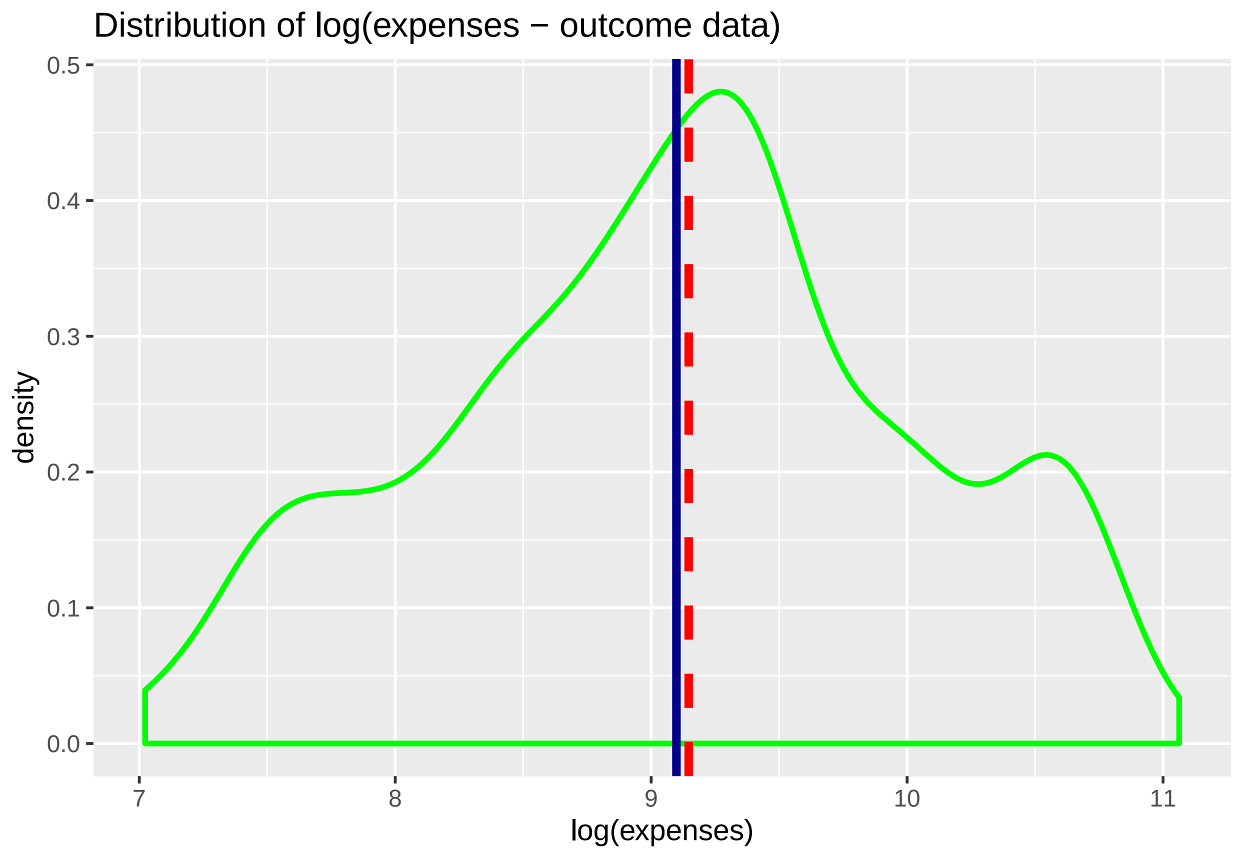


```
summary(insurance$expenses)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1122    4740    9382   13270   16640   63770
```

As we can see, the outcome distribution isn't normal at all. Most of the people in your data have yearly medical expenses between zero and $\$15,000$. As we discussed before, this distribution is not ideal for a linear regression, because regression algoritms usually predict the expected, or mean value of the output. This means that predicting expenses directly will tend to overpredict the typical expenses for subjects with a given set of characteristics. If you take the log of lognormally distributed data, the resulting data is normally distributed. This means the mean tracks the median, and the dynamic range of the data is more modest.

```
insurance %>% ggplot(aes(x = log(expenses))) +
  geom_density(lwd = 1, color = "green") +
  geom_vline(xintercept = median(log(insurance$expenses)),
             color = "red",
             linetype = "dashed",
             size  = 1.5) +
  geom_vline(xintercept = mean(log(insurance$expenses)),
             color = "darkblue",
             size = 1.5) +
  ggtitle("Distribution of log(expenses - outcome data)")
```

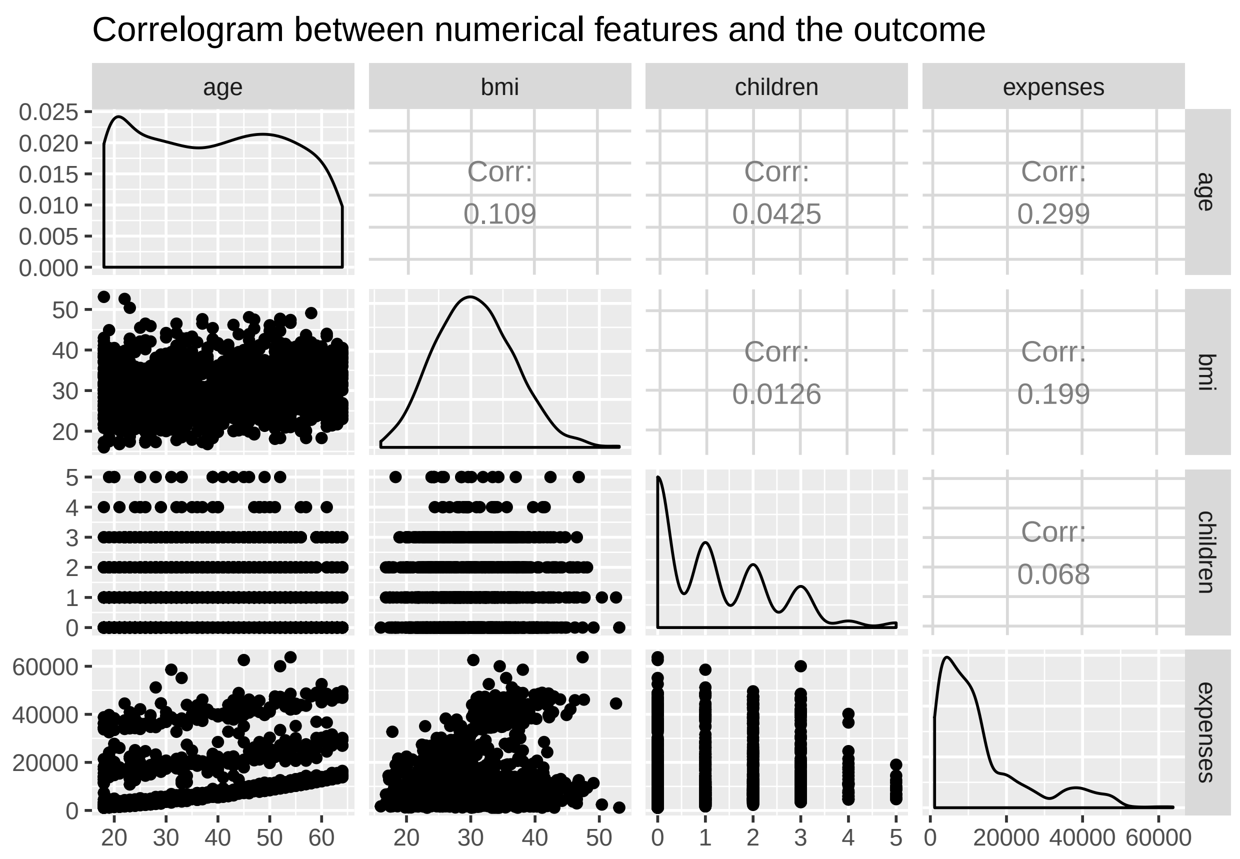


As we can see in the plot above, the median and the mean is close together and the distribution look like the normal distribution.

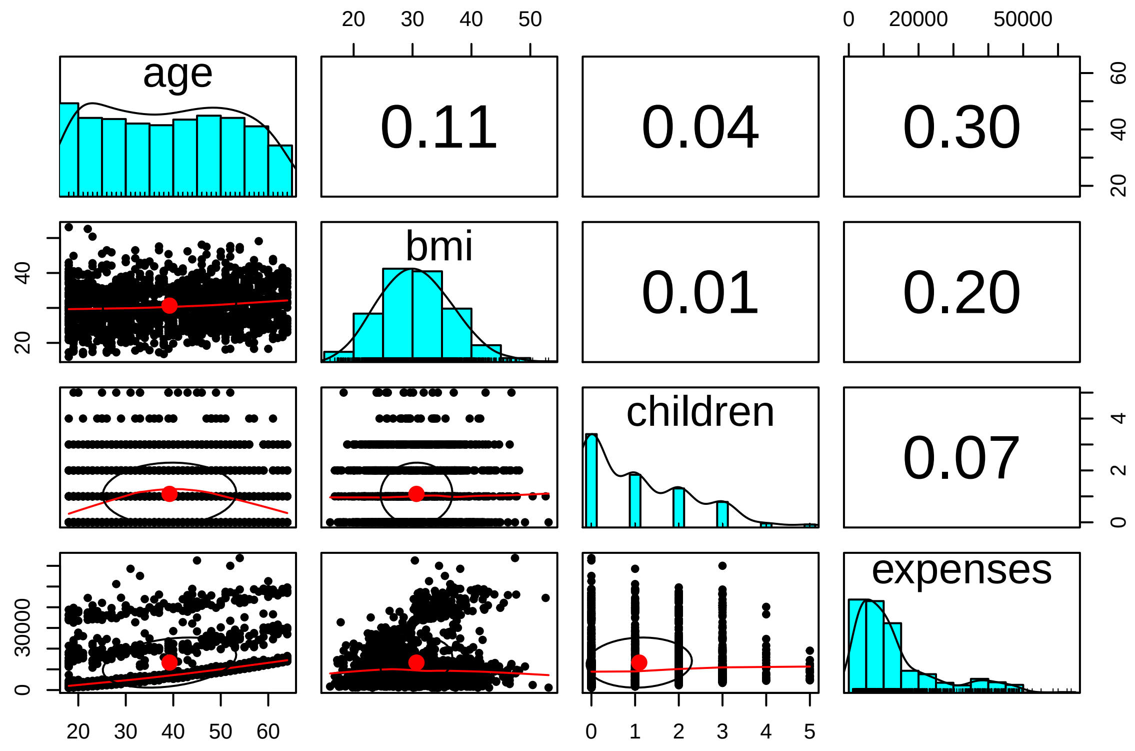
As a process of exploratory data analysis, we must explore the relationship among the predictors (or features), to perform that we will create the correlation plot.

A correlogram or correlation matrix allows us to analyse the relationship between each pair of a numeric variables in a dataset. It gives a quick overview of numerical variables. It is more used for *exploratory* purpose than explanatory.

```r
library(GGally)

insurance %>%
        select(c("age", "bmi", "children", "expenses")) %>%
        ggpairs(., title = "Correlogram between numerical features and the outcome")
```



In the scatterplot matrix above, we can find a pattern between age feature and expenses, they looks like straight lines, any other plots it is difficult to detected trends. The plot between bmi and expenses has two groups. If we adding more information to the matrix plot above, we can have more useful insights about the relationships. To create an better version of matrix plot we can use a function *pairs.panels()* from the psych package.



As we can see, the plot is more informative than before, the scaterplots above are presented with additional visual information.

The oval-shaped object on each scatterplot is a correlation ellipse. It provides a visualization of correlation strength. The more the ellipse is stretched, the stronger the correlation.

The curve drawn on the scatterplot is called a **loess curve**[1]. It is best understood by example. The curve for age and children is an upside-down U, peaking around middle age. This means tha the oldest and youngest people in the sample have fewer children on the insurance plan than those around middle age. Because this trend is nonlinear, this finding could not have been inferred from the correlations alone. For the loess curve between age and bmi is a line sloping up gradually, implying that body mass increases with age, but we had already inferred this from the correlation matrix.

Building a multiple linear regression model

To fit a linear regression model to data with R, we will the *caret* package for automated parameter tuning. Caret is the short for *Classification And Regression Training*. It is a complete package that covers all the stages of a pipeline for creating a machine learning predictive model.

Rather than choosing arbitrary values for each of the model's parameters - a task that is not only tedious but also somewhat unscientific - it is better to conduct a search through many possible parameter values to find the best combination.

The caret package, provide tools to assist with automated parameter tuning. The core functionality is provide by a train() function, as its name suggests, it is used to train a model and serves a standardized interface for over 200 different machine learning models for both classification and regression tasks.

[1]LOWESS (Locally Weighted Scatterplot Smoothing), sometimes called LOESS (locally weighted smoothing), is a popular tool used in regression analysis that creates a smooth line through a timeplot or scatter plot to help you to see relationship between variables and foresee trends. - (source: wikipedia)

As a necessary step for any modeling process, we have to split the data in training and test. In the training set, we will use to develop the model and the test set we use only for evaluation of the model. Caret has a function to help us to perform the split the data.

```r
str(insurance)
```

Machine Learning with caret - splitting data

```
## 'data.frame':    1338 obs. of  7 variables:
##  $ age     : int  19 18 28 33 32 31 46 37 37 60 ...
##  $ sex     : Factor w/ 2 levels "female","male": 1 2 2 2 2 1 1 1 2 1 ...
##  $ bmi     : num  27.9 33.8 33 22.7 28.9 25.7 33.4 27.7 29.8 25.8 ...
##  $ children: int  0 1 3 0 0 0 1 3 2 0 ...
##  $ smoker  : Factor w/ 2 levels "no","yes": 2 1 1 1 1 1 1 1 1 1 ...
##  $ region  : Factor w/ 4 levels "northeast","northwest",..: 4 3 3 2 2 3 3 2 1 2 ...
##  $ expenses: num  16885 1726 4449 21984 3867 ...
```

The advantage of using a train/test split rather than just validating your model in-sample on the training set. It gives us an estimate of how well your model performs on new data. Because, we have after this split process one sample (test set) to perform its evaluation out-of-sample performance and calculate the root-mean-square (RMSE). The test set is used for predict and determine the out-of-sample error for the model.

Just for checking the split process:

```
## insurance.train
##
##  7  Variables      938  Observations
## ---------------------------------------------------------------------------------
## age
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      938        0       47    0.999    39.37    16.21    18.85    20.00
##      .25      .50      .75      .90      .95
##    27.00    40.00    51.00    59.00    62.00
##
## lowest : 18 19 20 21 22, highest: 60 61 62 63 64
## ---------------------------------------------------------------------------------
## sex
##        n  missing distinct
##      938        0        2
##
## Value      female   male
## Frequency     462    476
## Proportion  0.493  0.507
## ---------------------------------------------------------------------------------
## bmi
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      938        0      257        1    30.76     7.02    21.09    22.90
##      .25      .50      .75      .90      .95
##    26.20    30.50    34.90    38.93    41.20
##
## lowest : 17.2 17.3 17.4 17.7 17.8, highest: 48.1 49.1 50.4 52.6 53.1
## ---------------------------------------------------------------------------------
## children
##        n  missing distinct     Info     Mean      Gmd
##      938        0        6    0.891    1.052     1.25
```

```
##
## lowest : 0 1 2 3 4, highest: 1 2 3 4 5
##
## Value          0     1     2     3     4     5
## Frequency    417   226   165   102    15    13
## Proportion 0.445 0.241 0.176 0.109 0.016 0.014
## ---------------------------------------------------------------------------
## smoker
##        n  missing distinct
##      938        0        2
##
## Value         no   yes
## Frequency    746   192
## Proportion 0.795 0.205
## ---------------------------------------------------------------------------
## region
##        n  missing distinct
##      938        0        4
##
## Value      northeast northwest southeast southwest
## Frequency        222       230       262       224
## Proportion     0.237     0.245     0.279     0.239
## ---------------------------------------------------------------------------
## expenses
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      938        0      938        1    13303    12393     1768     2317
##      .25      .50      .75      .90      .95
##     4741     9382    16640    34908    41741
##
## lowest :  1122  1132  1136  1137  1137, highest: 55135 58571 60021 62593 63770
## ---------------------------------------------------------------------------

## insurance.test
##
##  7  Variables      400  Observations
## ---------------------------------------------------------------------------
## age
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      400        0       47    0.999    38.82    16.24    18.00    19.00
##      .25      .50      .75      .90      .95
##    26.75    38.00    51.00    59.00    61.05
##
## lowest : 18 19 20 21 22, highest: 60 61 62 63 64
## ---------------------------------------------------------------------------
## sex
##        n  missing distinct
##      400        0        2
##
## Value      female   male
## Frequency     200    200
## Proportion    0.5    0.5
## ---------------------------------------------------------------------------
## bmi
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      400        0      186        1    30.43    6.601    21.50    23.20
##      .25      .50      .75      .90      .95
##    26.30    30.20    34.23    38.31    40.33
##
## lowest : 16.0 16.8 17.3 17.5 18.3, highest: 44.7 45.9 46.2 46.5 47.6
## ---------------------------------------------------------------------------------
## children
##        n  missing distinct     Info     Mean      Gmd
##      400        0        6    0.916    1.195    1.328
##
## lowest : 0 1 2 3 4, highest: 1 2 3 4 5
##
## Value          0     1     2     3     4     5
## Frequency    157    98    75    55    10     5
## Proportion 0.392 0.245 0.188 0.138 0.025 0.012
## ---------------------------------------------------------------------------------
## smoker
##        n  missing distinct
##      400        0        2
##
## Value         no   yes
## Frequency    318    82
## Proportion 0.795 0.205
## ---------------------------------------------------------------------------------
## region
##        n  missing distinct
##      400        0        4
##
## Value      northeast northwest southeast southwest
## Frequency        102        95       102       101
## Proportion     0.255     0.238     0.255     0.252
## ---------------------------------------------------------------------------------
## expenses
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      400        0      400        1    13193    12112     1747     2466
##      .25      .50      .75      .90      .95
##     4740     9396    16655    34683    41037
##
## lowest :  1136  1141  1242  1243  1256, highest: 46889 47056 48674 48676 49578
## ---------------------------------------------------------------------------------
```

The train() function has a few arguments and its the core of caret's package. This function allow us hyperparameter the processing of training (or fitting) the model and evaluate their out-of-sample performance using cross-validation and extract metrics for both classification and regression problems.

We can check any hyperparameters for any particular model, by the *modelLookup()* function.

```r
# check for the hyperparameters for lm model
modelLookup("lm")
```

```
##   model parameter     label forReg forClass probModel
## 1    lm intercept intercept   TRUE    FALSE     FALSE
```

We start the process of fit the model.

A better approach than a simple train/test split is using multiple test sets and averaging out-of-sample error, which gives us a more precise estimate of true out-of-sample error. One of the most common approaches for multiple test sets is known as "cross-validation", in which we split our data into ten "folds" or train/test splits.This is one of the best ways to estimate out-of-sample error for predictive models. The *trainControl()* function is used to create a set of configuration options known as a **control object**. This object guides the train() function and allows for the selection of model evaluation criteria, such as the resampling strategy and the measure used for choosing the best model. Caret supports many types of cross-validation, and you can specify which type of cross-validation and the number of cross-validation folds with the *trainControl()* function, which you pass to the *trControl* argument in *train()*.

It's important to note that you pass the method for modeling to the main train() function and the method for cross-validation to the trainControl() function.

```
## + Fold01: intercept=TRUE
## - Fold01: intercept=TRUE
## + Fold02: intercept=TRUE
## - Fold02: intercept=TRUE
## + Fold03: intercept=TRUE
## - Fold03: intercept=TRUE
## + Fold04: intercept=TRUE
## - Fold04: intercept=TRUE
## + Fold05: intercept=TRUE
## - Fold05: intercept=TRUE
## + Fold06: intercept=TRUE
## - Fold06: intercept=TRUE
## + Fold07: intercept=TRUE
## - Fold07: intercept=TRUE
## + Fold08: intercept=TRUE
## - Fold08: intercept=TRUE
## + Fold09: intercept=TRUE
## - Fold09: intercept=TRUE
## + Fold10: intercept=TRUE
## - Fold10: intercept=TRUE
## Aggregating results
## Fitting final model on full training set
```

```
# Print model to console
insurance_model1
```

```
## Linear Regression
##
## 938 samples
##   6 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 842, 844, 846, 844, 844, 845, ...
## Resampling results:
##
##   RMSE  Rsquared  MAE
##   6214  0.746     4282
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

Understanding the regression coefficients is fairly straightforward. The intercept is the predicted value of *expenses* when the independent variables are equal to zero. It is often impossible to have values of zero for all of the features, and consequently the intercept has no real-world interpretation.

The beta coefficients indicate the estimated increase in expenses for an increase of one unit in each of the features, assuming all other values are held constant. For example, for each additional year of age, we would expect $258.26 higher medical expenses on average, assuming everything elseis held equal.

The results of the linear regression model make logical sense: old age, smoking, and obesity tend to be conected to additional health issues, while additional family member dependents may result in an increase in physician visits and preventive care such vaccinations and yearly physical exams.

Evaluating model performance

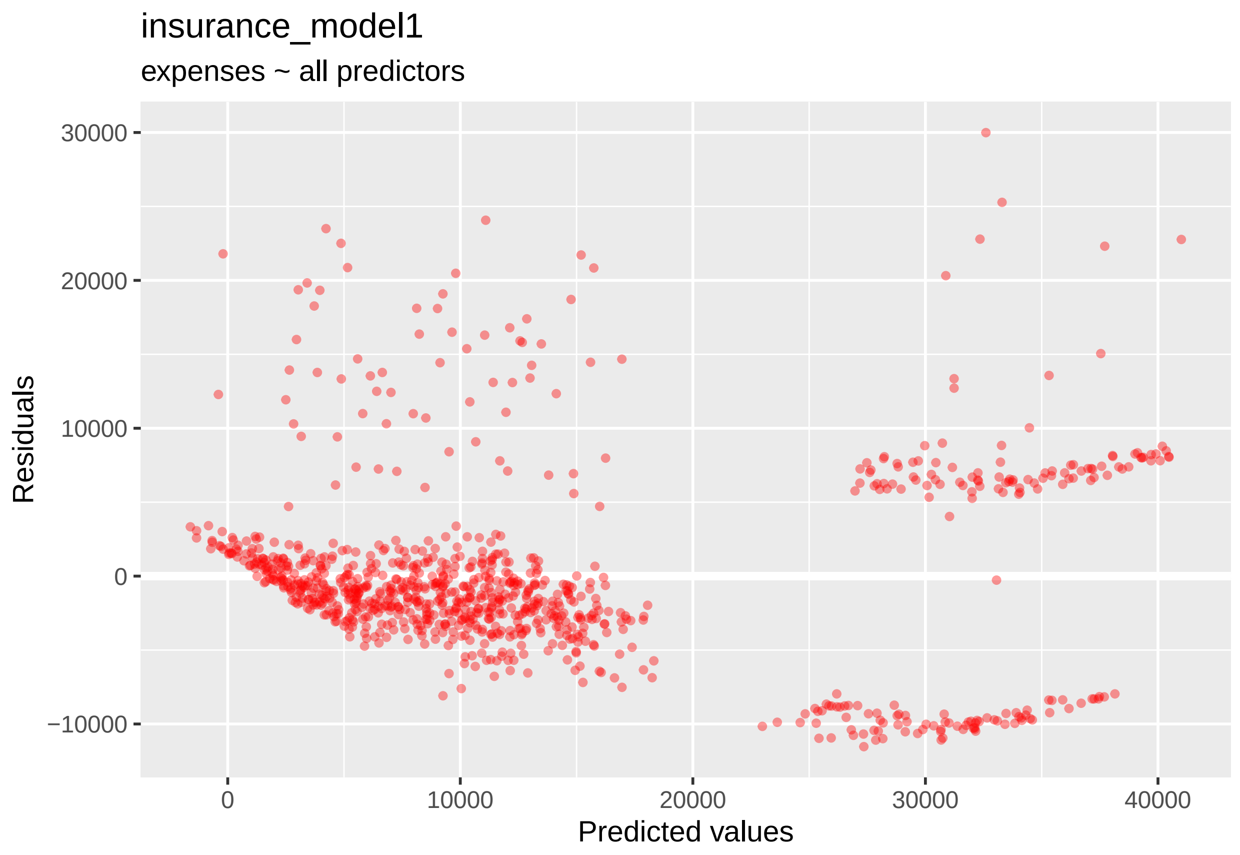
The parameter estimates we obtained by **summary(insurance_model1)** tell us about how the independent variables are related to the dependent variable, but they tell us nothing about how well the model fits out data.

The first key to evaluate the performance of our model is the residuals.

```r
summary(insurance_model1)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -11532  -2924   -977   1300  29991
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -11782.7     1182.7   -9.96   <2e-16 ***
## age                258.3       14.6   17.67   <2e-16 ***
## sexmale           -401.4      407.2   -0.99     0.32
## bmi                333.9       34.7    9.62   <2e-16 ***
## children           397.5      170.2    2.34     0.02 *
## smokeryes        23952.8      503.3   47.59   <2e-16 ***
## regionnorthwest   -200.0      583.3   -0.34     0.73
## regionsoutheast   -939.4      585.6   -1.60     0.11
## regionsouthwest   -674.6      588.6   -1.15     0.25
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6190 on 929 degrees of freedom
## Multiple R-squared:  0.746,  Adjusted R-squared:  0.744
## F-statistic:  341 on 8 and 929 DF,  p-value: <2e-16
```

The Residuals section provides summary statistics for the errors in your predictions, some of which are apparently quite large. Since a residual is equal to the true value minus the predicted value, the maximum error of $2999.1 suggests that the model under-predicted expenses by nearly $30,000 for at least one observation. On the other rand, 50% of error fall within the first and third quartile.



Another information from the residuals is quite important is constant variance among residuals. Linear regression assumes the variance among error terms are constant (this assumption is referred to as homoscedasticity). If the error variance is not constant, the p-values and confidence intervals for the coefficients will be invalid. Similar to the linear relationship assumption, non-constant variance can often be resolved with variable transformations or by including additional predictors. It seems we have some parabolics trends, it should be related to a specific feature. We will dive in some those detaills later on.

```r
summary(insurance_model1) %>%
  tidy()
```

```
## # A tibble: 9 x 5
##   term            estimate std.error statistic   p.value
##   <chr>              <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)      -11783.    1183.     -9.96  2.77e- 22
## 2 age                 258.      14.6    17.7   1.84e- 60
## 3 sexmale            -401.     407.     -0.986 3.24e-  1
## 4 bmi                 334.      34.7     9.62  5.91e- 21
## 5 children            397.     170.      2.34  1.97e-  2
## 6 smokeryes         23953.     503.     47.6   2.32e-251
## 7 regionnorthwest    -200.     583.     -0.343 7.32e-  1
## 8 regionsoutheast    -939.     586.     -1.60  1.09e-  1
## 9 regionsouthwest    -675.     589.     -1.15  2.52e-  1
```

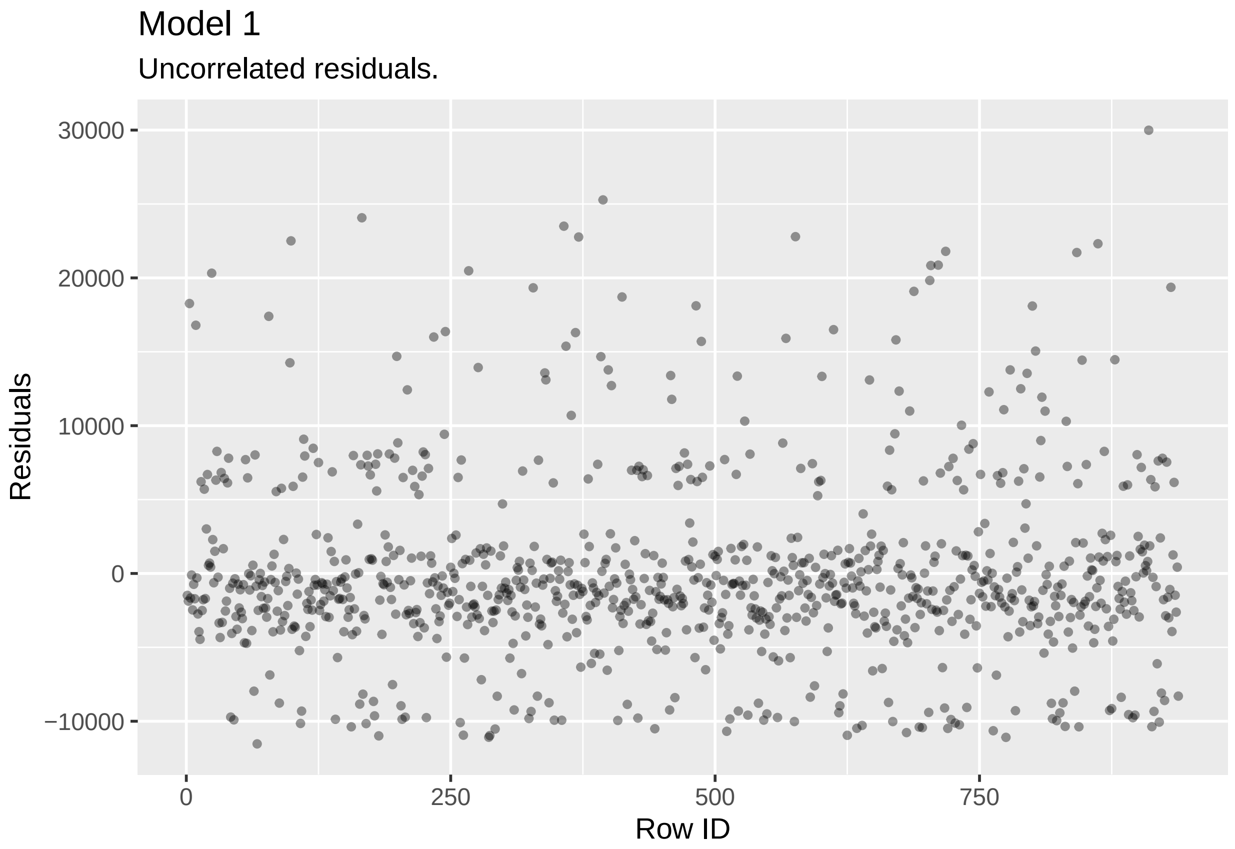
Another usual information from summary is about the regression coefficients, and the **p-value**, denoted Pr (>|t), provides an estimate of the probability that the true coefficient is zero given the value of the estimate. Small p-values suggest that the true coefficient is very unlikely to be zero, which means that the feature is extremely unlikely to have no relationship with the outcome variable. The p-values less than the significance level are considered **statistic significant**. Our model has several highly significant variables, and they seem to be related to the outcome in logical ways.

```
summary(insurance_model1) %>%
  glance()
```

```
## # A tibble: 1 x 6
##   r.squared adj.r.squared sigma statistic   p.value    df
##       <dbl>         <dbl> <dbl>     <dbl>     <dbl> <int>
## 1     0.746         0.744 6193.      341. 2.56e-270     9
```

The **Multiple R-squared** value (also called the coefficient of determination) provides a measure of how well our model as a whole explains the values of the dependent variable. It is similar to the correlation coefficient, closer to 1 better the model explains the data. It means the model explains nearly 75% of the variation in the dependent variable.

Linear regression assumes the errors are independent and uncorrelated. If in fact, there is correlation among the errors, then the estimated standard errors of the coefficients will be biased leading to prediction intervals being narrower than they should be.



Given all the perfomance indicators, our model is performing fairly well. The size of some of the errors is a bit concerning, but not surprising given the nature of medical expense data. Up to know we only check the results of the model in the training set. We don't to overfit, however before we test the model in the test set, lets try to improve the model performance, by add feature related to the outcome and transform the outcome by log10 transformation.

Model specification - adding nonlinear relationships

In the linear regression, the relationship between an independent variable and the dependent variable is assumed to be linear, yet this may not necessarily be true. For example, for old ages the treatment may become disproportionately expensive. A typical regression equation follows a form similar to this :

$$y = \alpha + \beta_1 x$$

To account for a nonlinear relantionship, such x^2, the model become a polynomial and the relationship between the outcome and predictors can be modeling like this:

$$y = \alpha + \beta_1 x + \beta_2 x^2$$

To add the nonlinear age to model, we write the follow code:

```r
insurance <- insurance %>% mutate(age2 = age^2)
```

Transformation - converting a numeric variable to a binary indicator.

BMI may have higher impact for individuals are fat, which should increase the cost for such individuals. To model this relationship, we create a binary obesity indicator that is 1 if the BMI is at least 30 and 0 if it is less than 30. The estimated beta for this binary feature would then inficate the average net impact on medical expenses for individuals with BMI of 30 or above, relative to those with BMI less than 30.

```r
insurance$bmi30 <- ifelse(insurance$bmi >= 30, 1, 0)
```

Model specification - adding interaction effects

So far, we have only considered each feature's individual contribution to the outcome. What if certain features have a combined impact on the dependent variable? For instance, smoking and obesity may have harmful effects separately, but it is reasonable to assume that their combined effect may be worse than the sum of each one alone.

When two features have a combined effect, this is known as an **interation**. To interact the obesity indicator (bmi30) with the smoking indicator (smoker) we would write a formula in the form expenses - bmi30*smoker

```r
set.seed(42)

# create partition  index

index <- createDataPartition(insurance$expenses, p = 0.70, list = FALSE)

# subset insurance data with index
insurance.train <- insurance[index, ]# use to fit the model
insurance.test <- insurance[-index,] # are essential for making sure your models


# Train model using 10-fold cross-validation
# trainControl() function - method parameter is used to set the resampling
# method, such as holdout sampling or k-fold CV
#
set.seed(123)    # for reproducibility

fitControl <- trainControl(
        method = "cv",
        number = 10,
        selectionFunction = "oneSE",
        verboseIter = TRUE
)

insurance_model2 <- train(
        expenses ~ age + age2 + children + bmi + sex + bmi30*smoker + region,
        data = insurance.train,
        method = "lm",
        trControl = fitControl
)
```

```
## + Fold01: intercept=TRUE
## - Fold01: intercept=TRUE
## + Fold02: intercept=TRUE
## - Fold02: intercept=TRUE
## + Fold03: intercept=TRUE
## - Fold03: intercept=TRUE
## + Fold04: intercept=TRUE
## - Fold04: intercept=TRUE
## + Fold05: intercept=TRUE
## - Fold05: intercept=TRUE
## + Fold06: intercept=TRUE
## - Fold06: intercept=TRUE
## + Fold07: intercept=TRUE
## - Fold07: intercept=TRUE
## + Fold08: intercept=TRUE
## - Fold08: intercept=TRUE
## + Fold09: intercept=TRUE
## - Fold09: intercept=TRUE
## + Fold10: intercept=TRUE
## - Fold10: intercept=TRUE
## Aggregating results
## Fitting final model on full training set
```

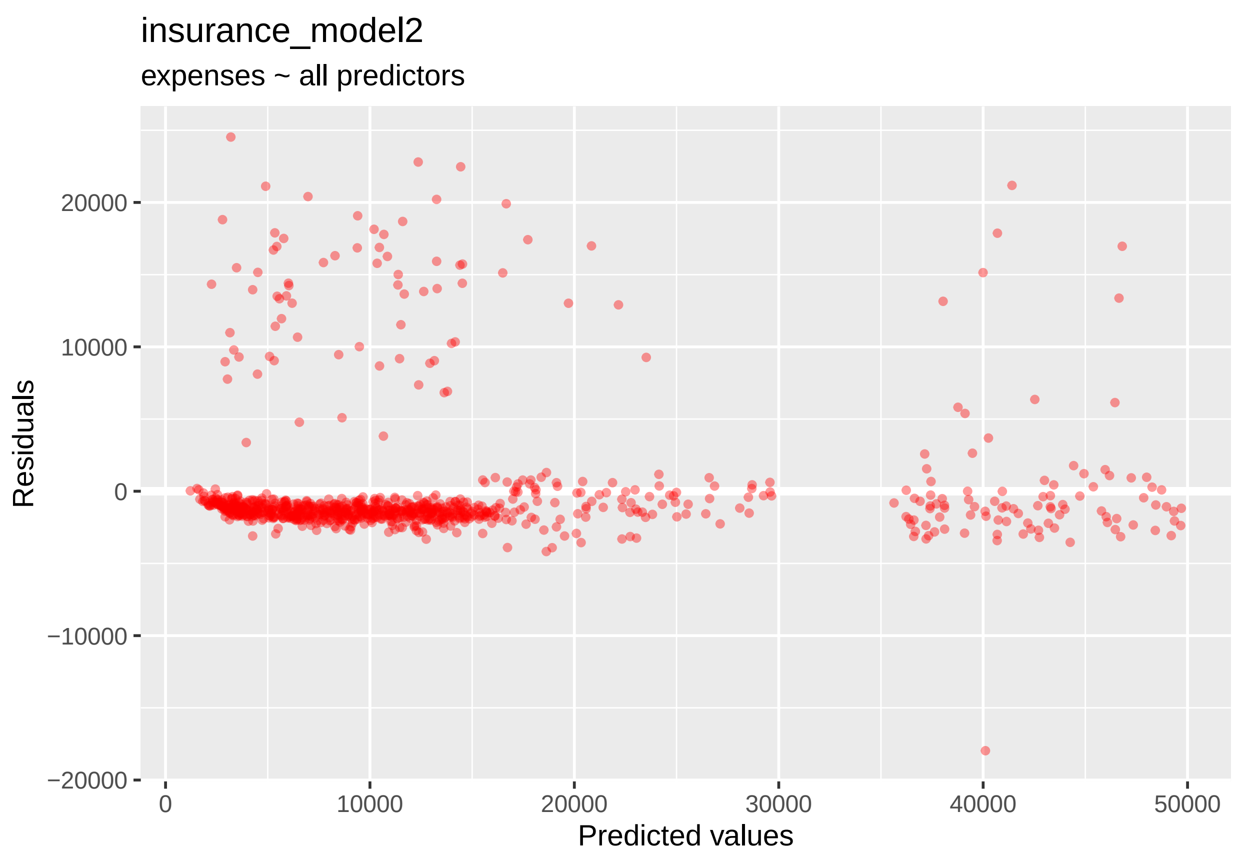
```r
# Print model 2 to console
insurance_model2
```

```
## Linear Regression
##
## 938 samples
##   8 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 842, 844, 846, 844, 844, 845, ...
## Resampling results:
##
##   RMSE  Rsquared  MAE
##   4528  0.864     2495
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
summary(insurance_model2)
```

```
##
```

```
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -17969  -1675  -1274   -764  24529
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -1065.198   1673.039   -0.64   0.5245
## age                  45.459     73.621    0.62   0.5371
## age2                  2.835      0.919    3.09   0.0021 **
## children            623.840    132.001    4.73  2.6e-06 ***
## bmi                 112.244     41.518    2.70   0.0070 **
## sexmale            -549.541    299.577   -1.83   0.0669 .
## bmi30              -917.123    520.754   -1.76   0.0785 .
## smokeryes         13418.445    536.611   25.01  < 2e-16 ***
## regionnorthwest    -242.185    429.486   -0.56   0.5730
## regionsoutheast    -902.202    431.762   -2.09   0.0369 *
## regionsouthwest   -1280.936    433.880   -2.95   0.0032 **
## `bmi30:smokeryes` 20230.285    740.978   27.30  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4560 on 926 degrees of freedom
## Multiple R-squared:  0.863,  Adjusted R-squared:  0.861
## F-statistic:  530 on 11 and 926 DF,  p-value: <2e-16
```



From the plot above there is no more parabolic pattern on the plot, it seems a improve from the model 1.

```r
summary(insurance_model2) %>%
  tidy()
```

```
## # A tibble: 12 x 5
##    term               estimate std.error statistic   p.value
##    <chr>                 <dbl>     <dbl>     <dbl>     <dbl>
##  1 (Intercept)         -1065.     1673.       -0.637 5.24e-  1
##  2 age                    45.5      73.6       0.617 5.37e-  1
##  3 age2                    2.83      0.919     3.09  2.09e-  3
##  4 children              624.      132.        4.73  2.65e-  6
##  5 bmi                   112.       41.5       2.70  6.99e-  3
##  6 sexmale              -550.      300.       -1.83  6.69e-  2
##  7 bmi30                -917.      521.       -1.76  7.85e-  2
##  8 smokeryes           13418.      537.       25.0   7.32e-106
##  9 regionnorthwest      -242.      429.       -0.564 5.73e-  1
## 10 regionsoutheast      -902.      432.       -2.09  3.69e-  2
## 11 regionsouthwest     -1281.      434.       -2.95  3.23e-  3
## 12 `bmi30:smokeryes`   20230.      741.       27.3   7.03e-121
```

```r
summary(insurance_model2) %>%
  glance()
```

```
## # A tibble: 1 x 6
##   r.squared adj.r.squared sigma statistic p.value    df
##       <dbl>         <dbl> <dbl>     <dbl>   <dbl> <int>
## 1     0.863         0.861 4556.      530.       0    12
```

Let's summarize your first two models before we try the last one (should transforme to log10(expenses)).

```
# Extract out of sample performance measures
summary(resamples(list(
  model1 = insurance_model1,
  model2 = insurance_model2
)))
```

```
##
## Call:
## summary.resamples(object = resamples(list(model1 = insurance_model1, model2
##  = insurance_model2)))
##
## Models: model1, model2
## Number of resamples: 10
##
## MAE
##        Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
## model1 3920    4088   4225 4282    4364 4795    0
## model2 2212    2285   2481 2495    2618 3038    0
##
## RMSE
##        Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
## model1 5654    5839   6151 6214    6458 7265    0
## model2 3652    4095   4417 4528    4941 5630    0
##
## Rsquared
##         Min. 1st Qu. Median  Mean 3rd Qu.  Max. NA's
## model1 0.671   0.729  0.756 0.746   0.772 0.791    0
## model2 0.813   0.844  0.868 0.864   0.883 0.903    0
```

For models predicting a numeric outcome (expenses) some measure of accuracy (RMSE)is typically used to evaluate the effectiveness of the model. From the cross-validation, we concluded the second model present a better results and it is safe to infer that by adding extra predictors (features) we improved **RMSE** substantially. To understand the strengths and weakness of this model, it is not reliable only a single metric. Visualizations of the model fit, particularly residual plots, are critical to understanding whether the model is fit for purpose.

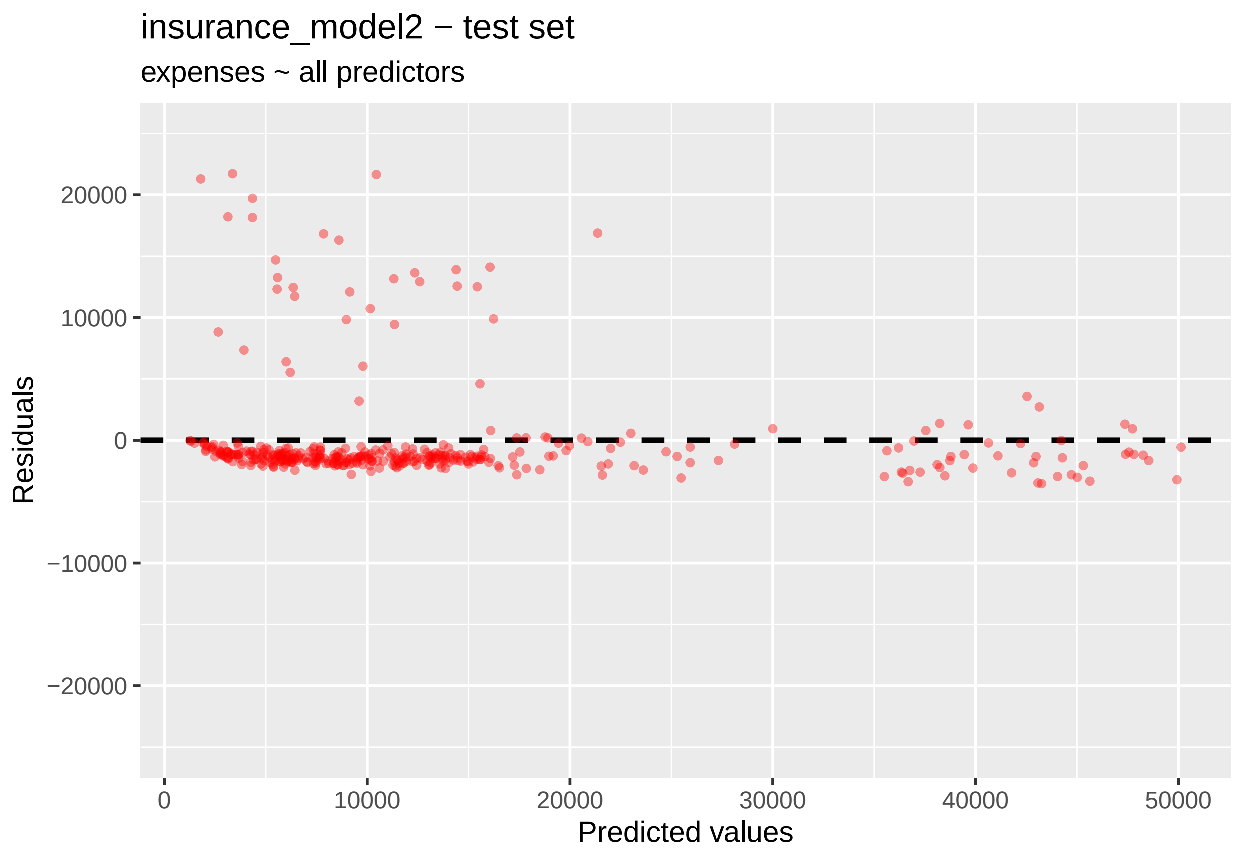
```
# Making predictions on the test set for model 2

insurance.test2 <- insurance.test %>%
        mutate(predictions = predict(insurance_model2, insurance.test))
```

```
# Residual for R.squared
insurance.test2 <- insurance.test2 %>%
        mutate(residualValues = expenses - predictions)
summary(insurance.test2$residualValues)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -3532   -1718   -1330    -170    -843   21721
```

It is important to check the plot of the residuals versus the predicted values can help uncover systematic patterns in the model predictions.

```
ggplot(data = insurance.test2, aes(x = predictions, y = residualValues)) +
  geom_hline(yintercept = 0, lwd = 1, color = "black",linetype = 2) +
  geom_point(size = 1, alpha = 0.4, color = "red") +
  xlab("Predicted values") +
  ylab("Residuals") +
  scale_y_continuous(limits = c(-25000,25000)) +
  ggtitle("insurance_model2 - test set", subtitle = "expenses ~ all predictors")
```



The caret package contains functions for calculating the RMSE and R^2 value:

```
# Rsquared and RMSE for test set
observed <- insurance.test2$expenses
predicted <- insurance.test2$predictions

cat("Correlation calculated by caret = ", R2(predicted, observed), '\n')
```

```
## Correlation calculated by caret =  0.875
```

```
cat("Simple correlation = ", cor(predicted, observed), '\n')
```

```
## Simple correlation =  0.936
```

```
cat("RMSE = ", RMSE(predicted, observed))
```

```
## RMSE =  4199
```

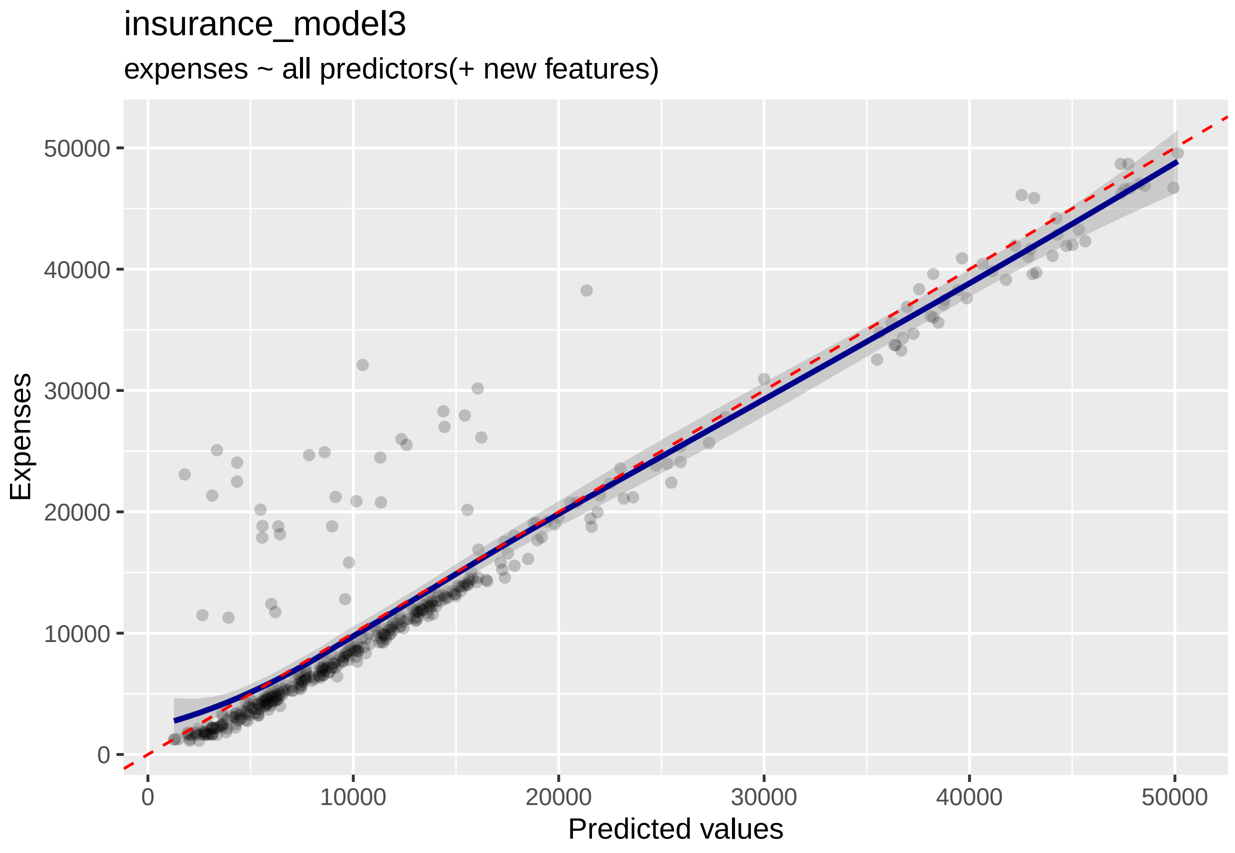
From the both results above and in agreement with other analysis is secure to say, the model 2 is accurate, R^2 is high enough and RMSE seems to be small enough. We can check up whether is small or large by:

```
# A RMSE much smaller than the outcome's standard deviation
# suggests a model that predict well
```

```
sd_outcome <- sd(insurance.test2$expenses)
RMSE <- RMSE(predicted, observed)
cat("RMSE/sd_outcome", (RMSE/sd_outcome)*100)
```

```
## RMSE/sd_outcome 35.5
```

As the number above shows the RMSE is small enough which means the model predicts well the future outcome or unseen data. It is useful to plot the observed values against the predicted values helps one to understand how well the model fits.



The previous results are almost the same we got in the cross-validation on the model2 as we can check up above. And as we can expected this result it better on the test and it is not overfitted, and can be use to forecast the expenses for potential new enrollees on the insurance plan.

From the analysis above we would say the linear regression is less prone to overfitting, which means the performance on the new data is usually quite similar to its performance on the training data. Regression modeling makes some strong assumptions about the data. These assumptions are not as important for numeric forecasting, as the model's worth is not based upon whether it truly captures the underlying process - we simply care about the accuracy of its predictions.

The last model its a plus in this analysis, we are going to check if we transform the outcome, it will improve the result in anyway.

The procedure we are going to perform log transformation (model 3) has 3 steps:

1 - Log the outome and fit a model

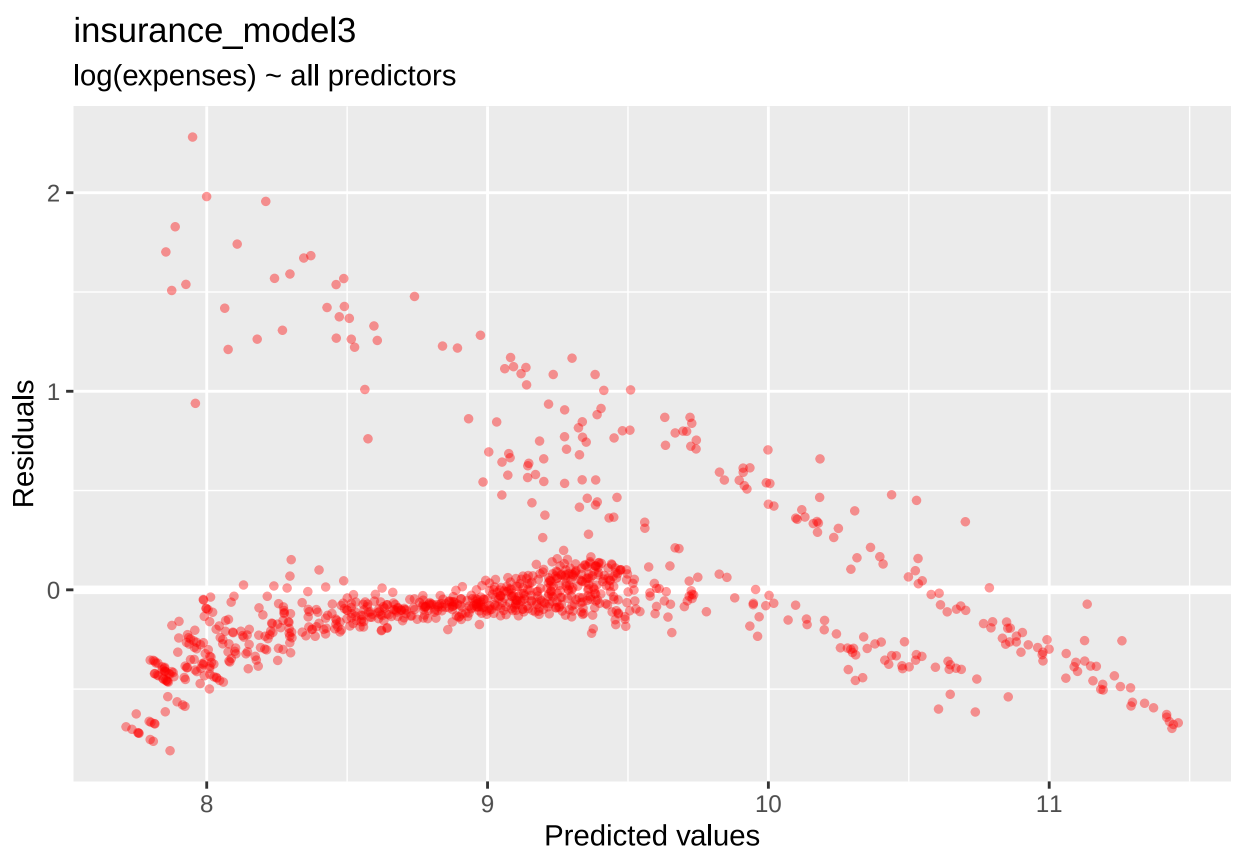
```
# Train model using 10-fold cross-validation
# trainControl() function - method parameter is used to set the resampling
```

```r
# method, such as holdout sampling or k-fold CV
#
set.seed(123)   # for reproducibility

fitControl <- trainControl(
        method = "cv",
        number = 10,
        selectionFunction = "oneSE",
        verboseIter = FALSE
)

insurance_model3 <- train(
        log(expenses) ~ age + age2 + children + bmi + sex + bmi30*smoker +  region,
        data = insurance.train,
        method = "lm",
        trControl = fitControl
)
```

```r
print(insurance_model3)
```

```
## Linear Regression
##
## 938 samples
##   8 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 842, 844, 846, 844, 845, 845, ...
## Resampling results:
##
##   RMSE   Rsquared  MAE
##   0.418  0.797     0.264
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
df3 <- augment(insurance_model3$finalModel, data = insurance.train)
df3 %>%
  ggplot(aes(x = .fitted, y = .resid)) +
  geom_hline(yintercept = 0, lwd = 1.5, color = "white") +
  geom_point(size = 1, alpha = 0.4, color = "red") +
  xlab("Predicted values") +
  ylab("Residuals") +
  ggtitle("insurance_model3", subtitle = "log(expenses) ~ all predictors")
```



2 - Making predictions on the test set:

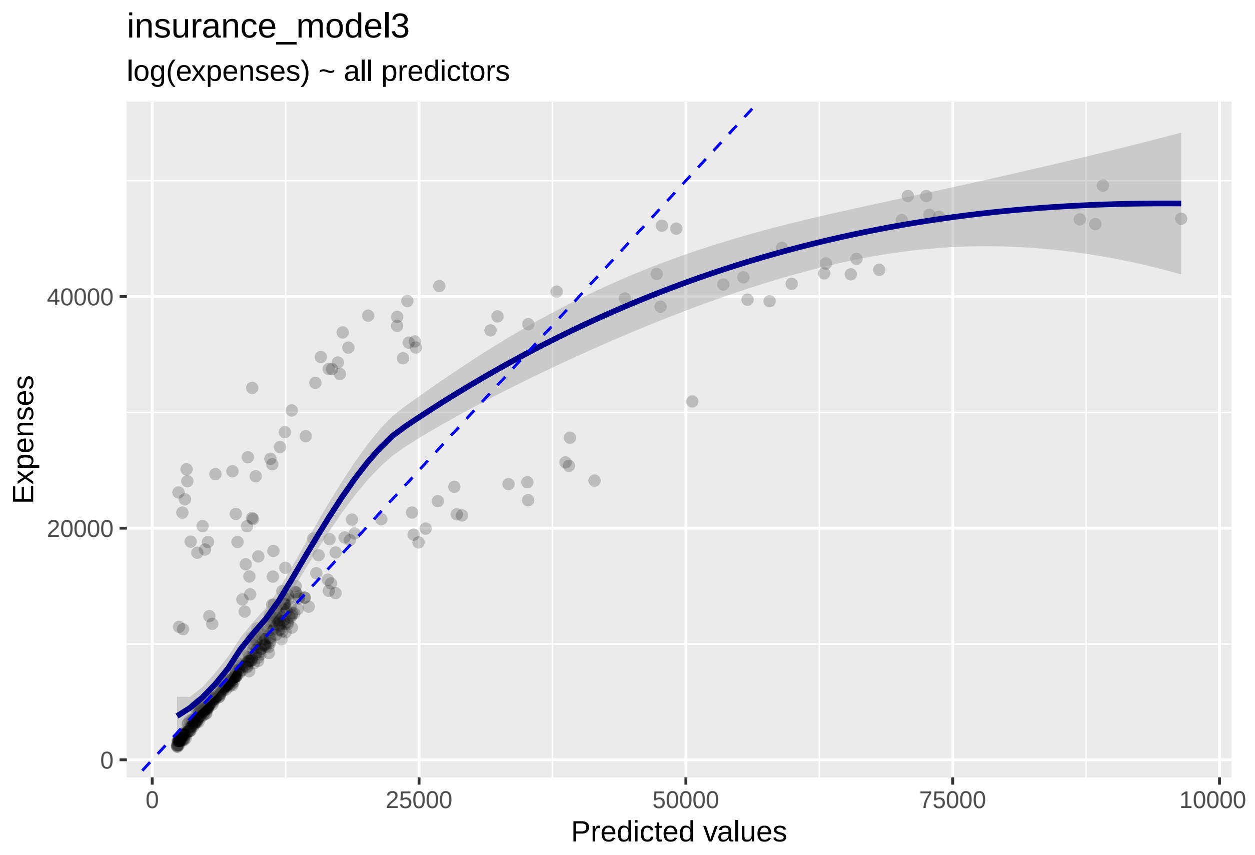
```
insurance.test <- insurance.test %>%
        mutate(logpredictions = predict(insurance_model3, insurance.test))
summary(insurance.test$logpredictions)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    7.75    8.50    9.08    9.09    9.44   11.48
```

3 - Convert the predictions to expenses units

```
insurance.test <- insurance.test %>%
        mutate(predictions = exp(logpredictions))
summary(insurance.test$predictions)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2326    4921    8815   13225   12554   96408
```



We can see in the plot above there is a problem in the predictions after $\$12,000$, so even with the transformation of the dependent variable to meet the normalization requirement we could not get a better predictions from model 2

```
# correlation between predictions and expenses values
cor(insurance.test$predictions, insurance.test$expenses)
```

```
## [1] 0.845
```