Prediction of insurance costs

Merthan DOGAN

```r
library(readr)
#we use tidyverse to manipulate data set missing values dublicate etc.
library(tidyverse)
#caTools helps us to split data set into train and test sets.
library(caTools)
#Calculate performance metrics
library(Metrics)
#the most populer ML package
library(caret)
#It provides a set of standard datasets for testing and comparing machine learning algorit
library(mlbench)
```

Determining The Insurance's Contests:

```r
#To export data set into the R
sigorta = read.csv("~/insurance.csv")

#we can check our data set with str() function
str(sigorta)
```

```
'data.frame':   1338 obs. of  7 variables:
 $ age     : int  19 18 28 33 32 31 46 37 37 60 ...
 $ sex     : chr  "female" "male" "male" "male" ...
 $ bmi     : num  27.9 33.8 33 22.7 28.9 ...
 $ children: int  0 1 3 0 0 0 1 3 2 0 ...
 $ smoker  : chr  "yes" "no" "no" "no" ...
 $ region  : chr  "southwest" "southeast" "southeast" "northwest" ...
 $ charges : num  16885 1726 4449 21984 3867 ...
```

Examining these features in the data set and determining what the insurance premium(charges) depends on.

1 - An insurance dataset typically contains information about insurance policies, premiums, and claims. The specific contents of the dataset will depend on the context and source of the data. Some possible features that might be included in an insurance dataset are;

- You can take a look insurance.csv from here.
- Our dataset consists of 1338 rows and 7 columns.(1338 Observations and 7 variables)
- Age: Age of the insurance holder (Numeric)
- Sex: Male or Female (categorical)
- BMI: Body Mass Index of the insurance holder (Numeric)
- Children: How many children does the insurer have (Numeric)
- Smoker: Yes or no (Categorical)
- Region: The region where the insurer lives (Categorical)
- Charges: Insurance premium (Numeric)

Add-ons

- There are no missing observations in the data set. (if it were we could use sigorta = na.exclude(sigorta)
- There are no duplication in data set. (if it were we could use duplicated(sigorta) also we could use distinct(sigorta) command with dplyr)
- Sex, Smoker, and region are categorical variables to make regression analysis we have to transform to factor.

Training a Linear Regression Model.

```r
#We need to transform categorical variables to factor
sigorta$sex = factor(sigorta$sex)
sigorta$smoker = factor(sigorta$smoker)
sigorta$region = factor(sigorta$region)

#Train a linear regression model.
#Firstly, we need to split the data set "test" and "train".
```

```r
#after that,Using the lm() function, we train a linear regression model using the "charges


set.seed(42)
trainIndex = createDataPartition(sigorta$charges, p = .8, list = FALSE)
train = sigorta[trainIndex,]
test = sigorta[-trainIndex,]
#We divide the data set into 80% training set and 20% test set.


#to make regression
regresyon <- lm(charges ~ ., data = train)

#with the summary() function. This summary includes each variable's regression coefficient
summary(regresyon)
```

```

Call:
lm(formula = charges ~ ., data = train)

Residuals:
     Min       1Q   Median       3Q      Max 
-11056.9  -2826.2   -938.3   1518.1  25250.3 

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)    
(Intercept)     -12131.05    1082.64 -11.205  < 2e-16 ***
age                257.14      13.20  19.474  < 2e-16 ***
sexmale           -460.15     367.56  -1.252 0.210878    
bmi                342.50      31.77  10.780  < 2e-16 ***
children           505.73     152.90   3.308 0.000973 ***
smokeryes        23926.95     455.28  52.554  < 2e-16 ***
regionnorthwest   -159.42     525.62  -0.303 0.761720    
regionsoutheast   -820.19     530.02  -1.547 0.122047    
regionsouthwest   -852.17     536.89  -1.587 0.112755    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5987 on 1063 degrees of freedom
Multiple R-squared:  0.7583,	Adjusted R-squared:  0.7564 
F-statistic: 416.8 on 8 and 1063 DF,  p-value: < 2.2e-16
```

```
#if want to plot the fitted model
#can use these command
#plot(charges ~ ., data = train, main = "Linear Regression Model",
#xlab = "Variables", ylab = "Charges")
#abline(regresyon, col = "red")
```

- Model output returns the information about model formula, train data, and the estimated values of model parameters under the coefficient title. You can see the numeric (continuous) features with its name, but the categorical features with name + first category like "sexmale"
- summary() function returns some statistics about the residuals, the model parameters, and also some performance metric values such as the multiple R^2 and the adjusted R^2. These values show the model performance on train data.

Inferences:

- Regression coefficients, we see that age, bmi, and smoker have a positive relationship with charges. This means that as these variables increase, charges also tend to increase. However, we don't observe a significant relationship between charges and sex or region.
- Significant variables: We can examine the p-values to determine the significance of each variable in the model. In this case, the p-values for age, bmi, and smoker are all very low, indicating that these variables are significant in the model. However, the p-values for sex and region are high, indicating that these variables are not significant in the model.
- Overfitting: To check for overfitting, we can test the accuracy of the model on a test dataset that the model has not seen before, and compare it to the accuracy on the train dataset. If the model performs significantly worse on the test dataset, it may be overfitting to the train data.
- Model fit: We can measure model fit using the R-squared value, which indicates how much of the variance in the data is explained by the model. According to the output from the lm function, the R-squared value is 0.75, indicating that the model explains approximately 75% of the data, which is a good fit.

Measuring Model Performance With R Squared

- For a continuous variable we can only use R squared, and metrics(MAE,RMSE,MSE), we're going to work "charges" variable which is continuous.

- Since the charges variable is a continuous variable, we cannot use the confusion matrix to evaluate the performance of the regression models. Confusion matrix is a criterion used in classification problems and is based on the numerical values of the predicted and actual classes. Since the variable charges is a continuous variable, it is a suitable criterion for regression problems, not classification. Therefore, it would be more appropriate to use regression performance measures such as RMSE or R-square.
- Regression analysis gives us the R-square. Also, We can calculate the R square by hands, and we can compare the train R square with test R square. If the R squared value in the training set is much higher than the R squared value in the test set, our model can be over-fitting. If the R squared value in the training set is low and the R squared value in the test set is also low, our model can be under-fitting problems.
- In this dataset, I chose the R-squared criterion because it directly measures the explanatory power of the model and is commonly used in regression problems. The R-square measure helps us evaluate the performance of the model by measuring how well the model explains the data.

```r
#To measure the model performance we use predict() command
predicted = predict(regresyon, newdata = test)
rsq = cor(test$charges, predicted)^2

#Measure R Square
rsq_train = summary(regresyon)$r.squared

#Comparing train and test sets.
rsq_train
```

```
[1] 0.7582536
```

```r
rsq
```

```
[1] 0.7203222
```

- The R squared value in the training set is 0.75, and the R squared value in the test set is 0.72.These values are close to each other, our model does not seem to over-fitting or under-fitting, but there is still a little bit probablity of over-fitting. However, other performance measures should also be used for a more comprehensive model, but to be sure we can calculate the metrics aswell.

Measuring Model Performance With Metrics, and Checking Over and Under-fitting:

```r
#Training model and predicts
#The cross-validation method with the trainControl function is chosen.
fitControl = trainControl(method = "cv", number = 5)

model = train(charges ~ ., data = train, method = "lm", trControl = fitControl)
train_pred = predict(model, train)
test_pred = predict(model, test)

#To calculate metrics
train_mse = mean((train_pred - train$charges)^2)
test_mse = mean((test_pred - test$charges)^2)

train_rmse = sqrt(train_mse)
test_rmse = sqrt(test_mse)

train_mae = mean(abs(train_pred - train$charges))
test_mae = mean(abs(test_pred - test$charges))

#Printing the results
cat(paste("Train MSE: ", train_mse, "\n"))
```

```
Train MSE:  35543723.2883605
```

```r
cat(paste("Test MSE: ", test_mse, "\n"))
```

```
Test MSE:  40567253.0521517
```

```r
cat(paste("Train RMSE: ", train_rmse, "\n"))
```

```
Train RMSE:  5961.85569167525
```

```r
cat(paste("Test RMSE: ", test_rmse, "\n"))
```

```
Test RMSE:  6369.24273773199
```

```r
cat(paste("Train MAE: ", train_mae, "\n"))
```

```
Train MAE:  4130.93319460001
```

```r
cat(paste("Test MAE: ", test_mae, "\n"))
```

```
Test MAE:  4293.77536132714
```

```r
msegap = train_mse - test_mse
rmsegap = train_rmse - test_rmse
msegap
```

```
[1] -5023530
```

```r
rmsegap
```

```
[1] -407.387
```

```r
#Let's do it with programming skill
if (test_rmse < train_rmse) {
  cat("The model is not overfitting.\n")
} else {
  cat("The model may be overfitting.\n")
}
```

```
The model may be overfitting.
```

Which metrics should i choose? RMSE and MAE are generally considered a better model performance measure.

- RMSE is a more appropriate error measure for a regression model and MSE for a classification model.
- According to these results, the RMSE in the test set is higher than the RMSE in the train set. This means that the model performed worse on the test set.

- The MSE's performance of the model on the test set is worse than the performance on the train set.
- According to the gaps, i choose lower gap, so i prefer the RMSE metrics with -407.387
- The difference is negative means that the performance of the model is better on test set than train set. This may be a sign for overfitting problem because the model performance is better on test set. So, we may say that the model learn more from the train set that cause the poorer performance on test set. But there is no threshold to conclude that there is any problem related to over or underfitting. We can use this difference just as a sign.
- The solution to the overfitting problem is to use various methods to reduce the complexity of the model and prevent overfitting. For example, methods such as using variable selection techniques, applying regularization techniques, and tuning hyperparameters can be used to collect more data and reduce model complexity.

Creating A New Observation, and predict the value of it's target feature

```
#Creating an example with values similar to properties in the dataset.
#I already made CV into sigorta data set, so i wont use traincontrol() command again.

yenigozlem = data.frame(
  age = 23,
  sex = "male",
  bmi = 25,
  children = 0,
  smoker = "yes",
  region = "southeast"
)
yenigozlem$sex = as.factor(yenigozlem$sex)
yenigozlem$smoker = as.factor(yenigozlem$smoker)
yenigozlem$region = as.factor(yenigozlem$region)

# Train a linear regression model again (not necesserey)
model = train(
  charges ~ .,
  data = sigorta,
  method = "lm"
)
```

```r
#Let's make a prediction for our new observation using thepredict() function
prediction = predict(model, newdata = yenigozlem)
cat("Estimated charge for new observation:", prediction)
```

```
Estimated charge for new observation: 25131.19
```

- There it is estimated charge for me. I predict my charge with my informations.