Chapter 14: Use Case

Ram Gopal, Dan Philps, and Tillman Weyde

2022

Contents

Load packages

```
library(car)
library(caret)
library(ggplot2)
library(leaps)
library(MASS)
library(corrgram)
```

Use Case: Profit Forecasting, Steps for a Safety-first Linear Regression

We have examined profit forecasting using R&D and marketing spend in a parametric context, where we had a good idea what the population distribution of profits was, and a non-parametric approach when we were not sure. We now tackle the same challenge but using a 4-point safety first process:

1. Check the data
2. Check for collinearities
3. Check model fit
4. Check residuals

Check1: Check the data

Check1 is simply data exploration, examining distributions and relationships as we have seen in previous chapters. We also need to check for imbalances in the dataset, particularly in classification problems, where we might be forecasting credit card loan defaults from a dataset where only 5% of the rows represent defaults (we will address this later in the book). Can we take a view on what the population distribution is? If so, our model will always be more accurate if we use tests that assume distributions that most resemble the true population distribution of our data.

```
df_train = read.csv("../../data/50_Startups.csv")
```

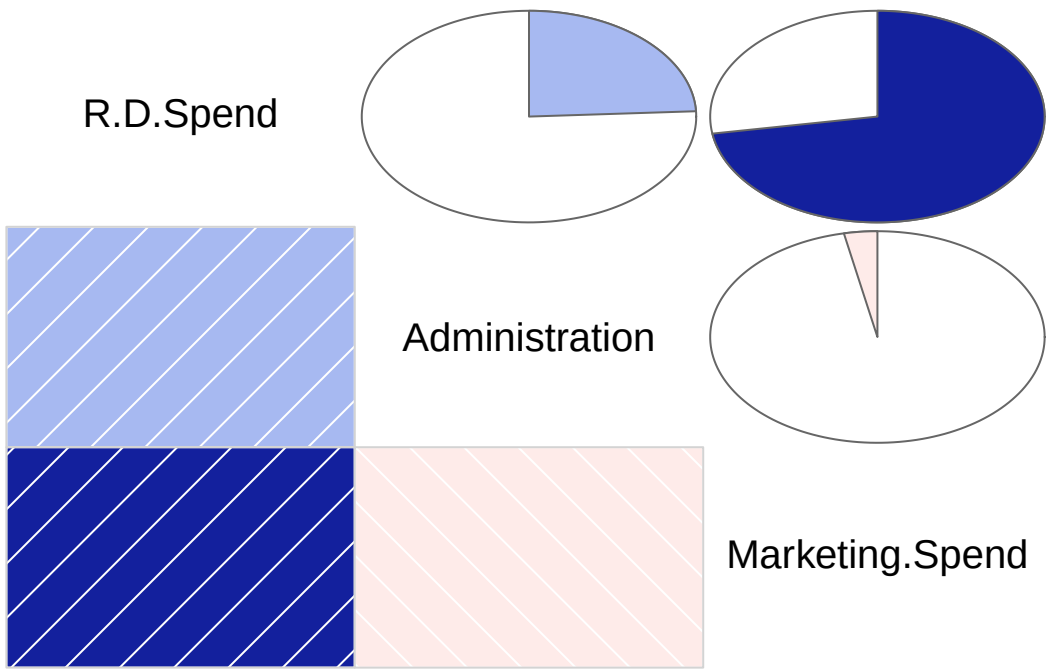
Check2: Check for collinearities

- corrgram package provides a nice visual

```
library(corrgram)
cor(df_train[1:3])
```

```
##                 R.D.Spend Administration Marketing.Spend
## R.D.Spend          1.0000        0.24196         0.72425
## Administration     0.2420        1.00000        -0.03215
## Marketing.Spend    0.7242       -0.03215         1.00000
```

```
corrgram(df_train[1:3],upper.panel = panel.pie)
```



We will use a rule of thumb that no 2 input variables should have a correlation coefficient of >0.5. You can see that R&D Spend and Marketing Spend have a correlation coefficient of 0.72 and so breach our rule of thumb. We will use *differencing* to see if correlation is reduced.

```
df_train_dif = df_train
df_train_dif$Marketing.Spend = df_train_dif$Marketing.Spend - df_train_dif$R.D.Spend
cor(df_train_dif[1:3])
```

```
##                 R.D.Spend Administration Marketing.Spend
## R.D.Spend          1.0000         0.2420          0.4515
## Administration     0.2420         1.0000         -0.1591
## Marketing.Spend    0.4515        -0.1591          1.0000
```

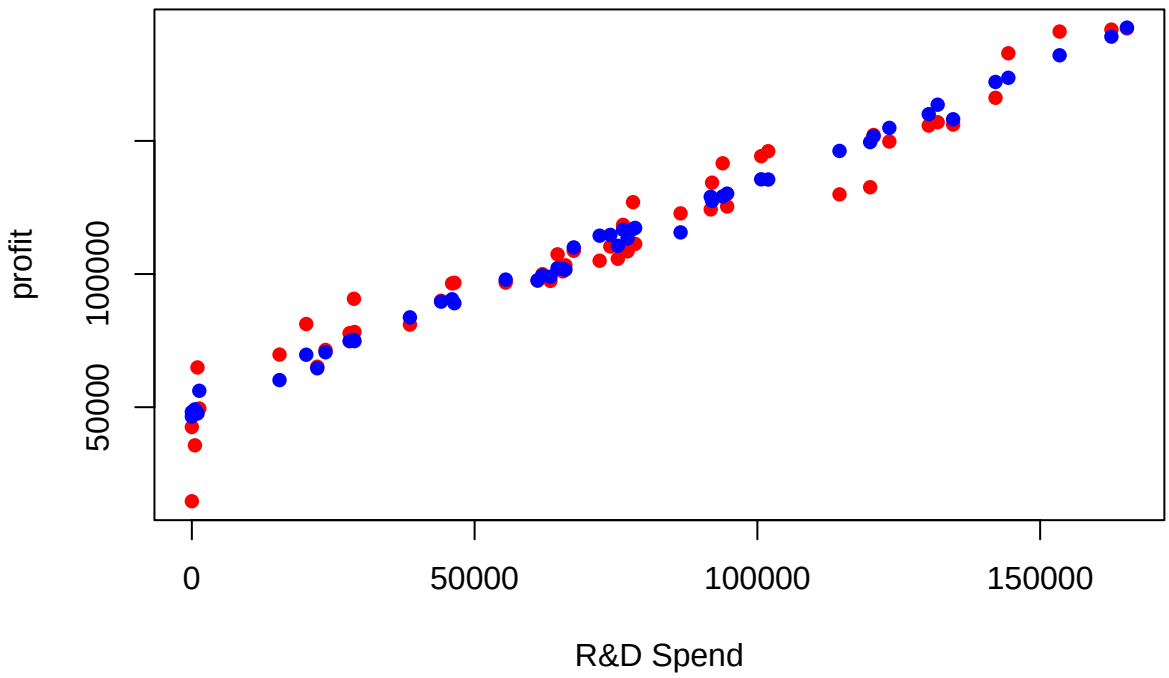
Check3: Check for Model Fit

We can now run the regression and assess the goodness of the model fit:

```
reg1 = lm(Profit ~ .-State, data=df_train_dif)
summary(reg1)
```

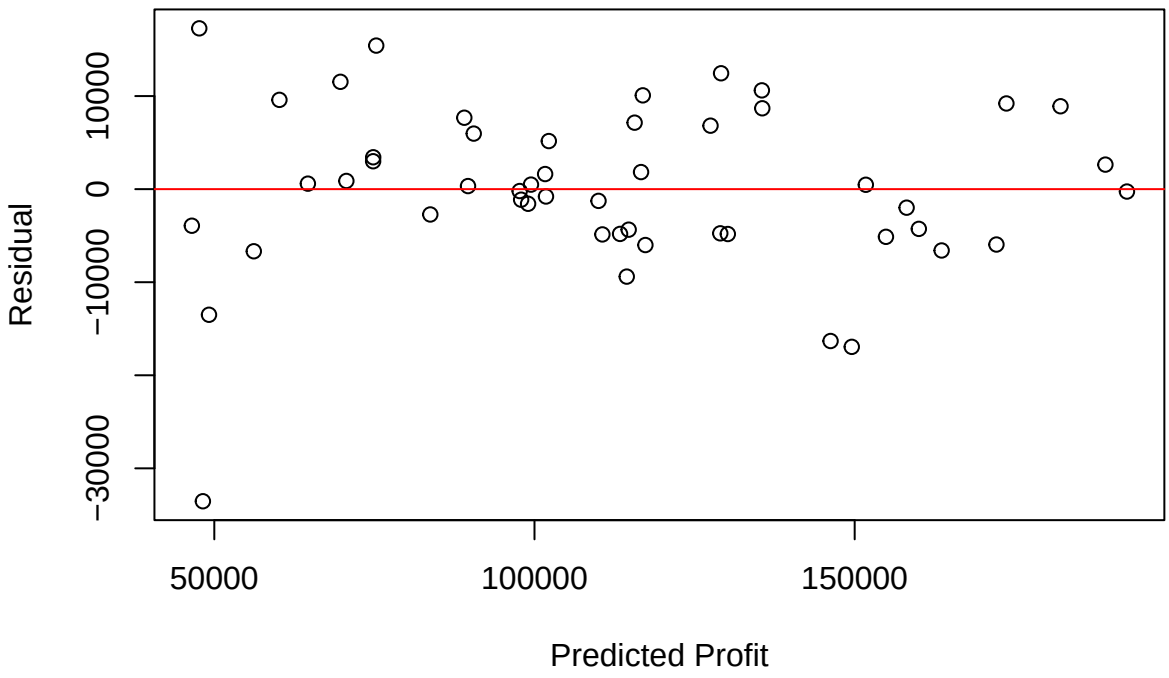
```
##
## Call:
## lm(formula = Profit ~ . - State, data = df_train_dif)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -33534  -4795     63   6606  17275
##
## Coefficients:
##                   Estimate Std. Error t value             Pr(>|t|)
## (Intercept)     50122.1930  6572.3526    7.63         0.0000000011 ***
## R.D.Spend           0.8329     0.0345   24.17 < 0.0000000000000002 ***
## Administration     -0.0268     0.0510   -0.53                  0.6
## Marketing.Spend     0.0272     0.0165    1.66                  0.1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9230 on 46 degrees of freedom
## Multiple R-squared:  0.951,  Adjusted R-squared:  0.948
## F-statistic:  296 on 3 and 46 DF,  p-value: <0.0000000000000002
```

```
plot(df_train_dif$R.D.Spend,df_train_dif$Profit,col="red",pch=16,xlab="R&D Spend",ylab="profit")
points(df_train_dif$R.D.Spend,reg1$fitted.values,col="blue",pch=16)
```

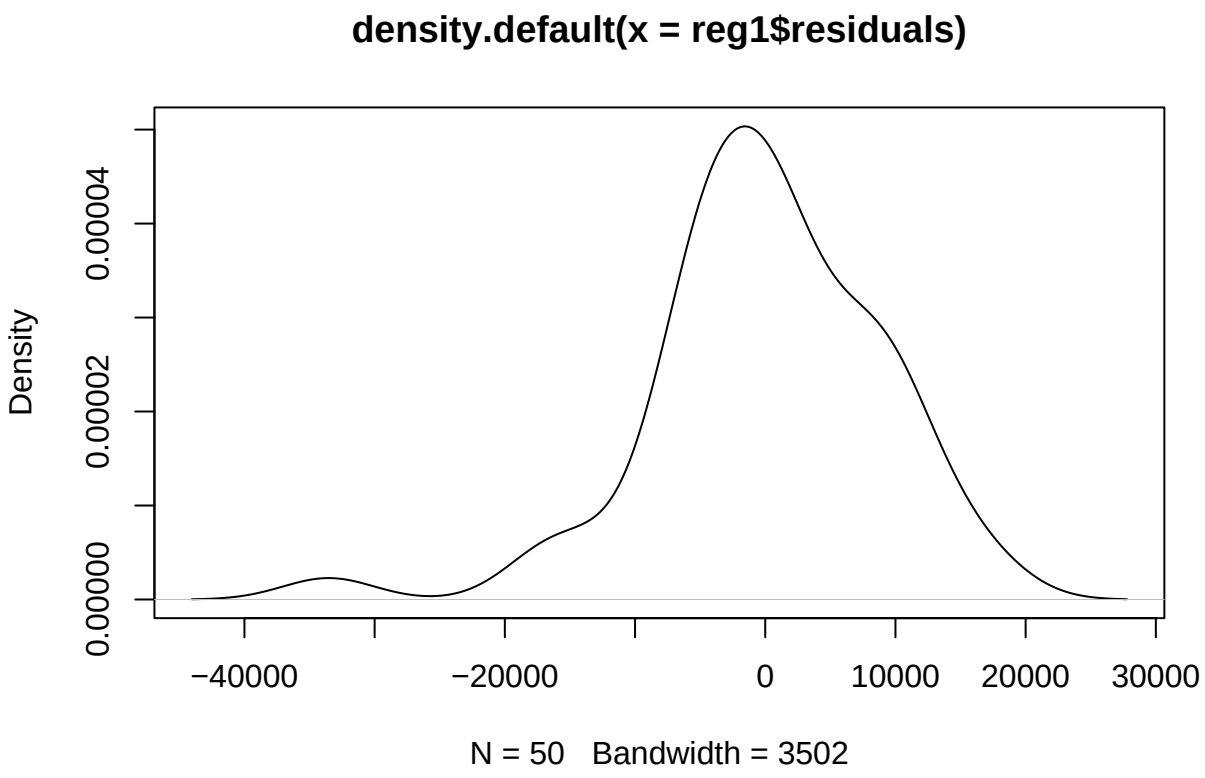


Check4: Check Residuals

```
plot(reg1$fitted.values,reg1$residuals,
     xlab="Predicted Profit",
     ylab = "Residual")
abline(h=0,col="red")
```




```
plot(density(reg1$residuals))
```




```
shapiro.test(reg1$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  reg1$residuals
## W = 0.94, p-value = 0.01
```

The plot and the normality test indicate that the normality assumption is a bit weak.

Automating Model Construction

In the R code below, we will run the subsets regression to select the right input variables:

```
library(leaps)
bestsub1 =  regsubsets(Profit ~ . - State, data = df_train_dif,nvmax = 12)
summary(bestsub1)
```

```
## Subset selection object
## Call: regsubsets.formula(Profit ~ . - State, data = df_train_dif, nvmax = 12)
## 3 Variables  (and intercept)
##                Forced in Forced out
## R.D.Spend          FALSE      FALSE
```

```
## Administration      FALSE      FALSE
## Marketing.Spend     FALSE      FALSE
## 1 subsets of each size up to 3
## Selection Algorithm: exhaustive
##          R.D.Spend Administration Marketing.Spend
## 1  ( 1 ) "*"       " "            " "
## 2  ( 1 ) "*"       " "            "*"
## 3  ( 1 ) "*"       "*"            "*"
```

```
names(summary(bestsub1))
```

```
## [1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
```

```
round(cbind(
    Cp     = summary(bestsub1)$cp,
    r2     = summary(bestsub1)$rsq,
    Adj_r2 = summary(bestsub1)$adjr2,
    BIC    = summary(bestsub1)$bic
),3)
```

```
##         Cp    r2 Adj_r2    BIC
## [1,] 3.932 0.947  0.945 -138.6
## [2,] 2.276 0.950  0.948 -138.5
## [3,] 4.000 0.951  0.948 -134.9
```

The results recommend using only "R&D Spend" and "Marketing Spend" as the input variables. Rerunning the regression model yields the following outcome which is an improvement over the initial regression model (for example, based on Adjusted R^2 values):

```
reg2 = lm(Profit ~ R.D.Spend+Marketing.Spend, data=df_train_dif)
summary(reg2)
```

```
##
## Call:
## lm(formula = Profit ~ R.D.Spend + Marketing.Spend, data = df_train_dif)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -33645  -4632   -414   6484  17097
##
## Coefficients:
##                   Estimate Std. Error t value            Pr(>|t|)
## (Intercept)     46975.8642  2689.9329   17.46 <0.0000000000000002 ***
## R.D.Spend           0.8265     0.0320   25.87 <0.0000000000000002 ***
## Marketing.Spend     0.0299     0.0155    1.93                0.06 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9160 on 47 degrees of freedom
## Multiple R-squared:  0.95,   Adjusted R-squared:  0.948
## F-statistic:  451 on 2 and 47 DF,  p-value: <0.0000000000000002
```