CHAPTER 11
REGRESSION MODELING

11.1 THE ESTIMATION TASK

Thus far in the Modeling Phase we have covered the following tasks:

- Classification task
- Clustering task

There remain two tasks left to cover:

- Estimation task
- Association task

In this chapter, we cover the estimation task; later, in Chapter 14, we will cover the association task.

The most widespread method for performing the estimation task is linear regression. Simple linear regression approximates the relationship between a numeric predictor and a continuous target, using a straight line. Multiple regression modeling approximates the relationship between a set of $p > 1$ predictors and a single continuous target, using a p-dimensional plane or hyperplane.

11.2 DESCRIPTIVE REGRESSION MODELING

The usual multiple regression model is a parametric model, defined by the following equation:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \varepsilon$$

where the x's represent the predictor variables, and the β's represent the unknown model parameters, whose values are estimated using the data.[1] Now, estimating

[1]For much more on inferential regression modeling, see *Data Mining and Predictive Analytics*.

Data Science Using Python and R, First Edition. Chantal D. Larose and Daniel T. Larose.
© 2019 John Wiley & Sons, Inc. Published 2019 by John Wiley & Sons, Inc.

model parameters using sample data represents classical statistical inference. The Data Science Methodology outlined in Chapter 1, however, employs cross-validation rather than classical statistical inference to validate model results. Thus, in this book, *we will bypass the parametric regression equation above, in favor of a descriptive approach to regression modeling*, using the following regression equation:

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_p x_p$$

In this regression equation, $\hat{y}$ represents the estimated value of the target variable y, the b's represent the *known* values of the regression coefficients, and the x's represent the predictor variables.

11.3 AN APPLICATION OF MULTIPLE REGRESSION MODELING

To illustrate multiple regression, we turn to the *clothing_sales_training* and *clothing_sales_test* data sets. The client has some data on customer spending and would like to estimate *Sales_per_Visit*, given three predictors:

- *Days between purchases* ("Days," Continuous: Average number of days between purchases.)
- *Credit Card* ("CC," Flag: Does the customer have a store credit card?)
- *Web Account* ("Web," Flag: Does the customer have a web account?)

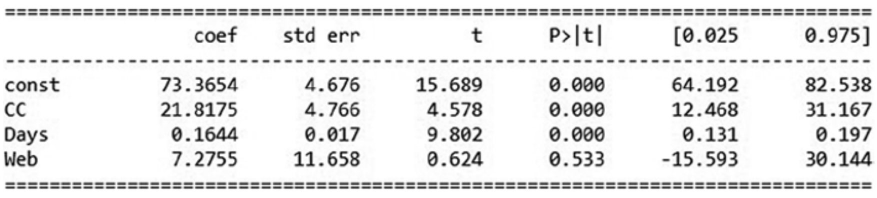
So, our provisional regression equation will be:

$$\begin{aligned} \textit{Sales per Visit} = {} & b_0 + b_1(\textit{Days between purchases}) + b_2(\textit{Credit Card}) \\ & + b_3(\textit{Web Account}) \end{aligned}$$

Because there is only one continuous predictor, it is not necessary to standardize the predictors. The results of the regression of *Sales per Visit* vs the three predictors for the training set are shown in Figure 11.1. We use the p-values as a guide to tell us which variables belong in the model. Note that we are not performing inference as such (the usual domain of p-values), because we will be careful to cross-validate these results with the test data set. The usual p-value cutoff for retaining variables in a regression model is about 0.05, though cutoff values differ from field to field. Variables with p-values lower than the cutoff are retained in the model.

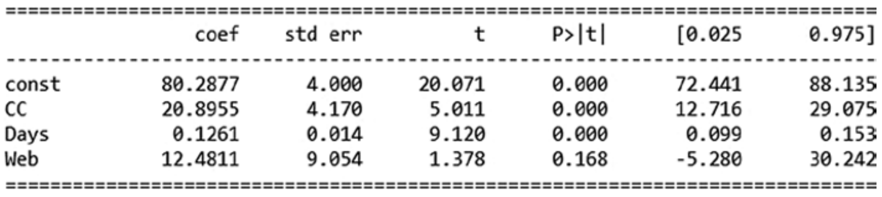
From Figure 11.1, we see that *Web Account*, with a p-value of 0.533, does not belong in the model. The regression results for the test data set in Figure 11.2 concur that *Web Account* does not belong in the model. This leaves us with our regression equation as:

$$\textit{Sales per Visit} = b_0 + b_1(\textit{Days between purchases}) + b_2(\textit{Credit Card})$$



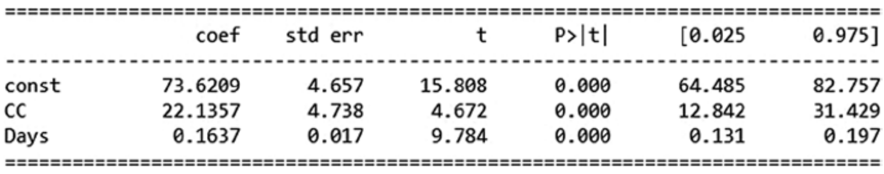
```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         73.3654      4.676     15.689      0.000      64.192      82.538
CC            21.8175      4.766      4.578      0.000      12.468      31.167
Days           0.1644      0.017      9.802      0.000       0.131       0.197
Web            7.2755     11.658      0.624      0.533     -15.593      30.144
==============================================================================
```

Figure 11.1 Python regression results for the training data set.



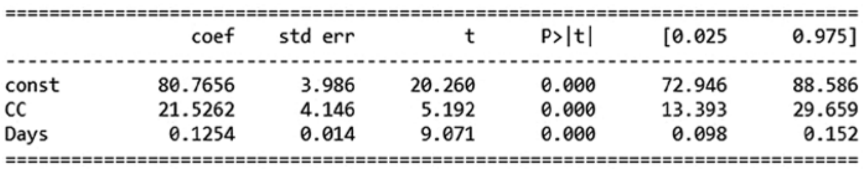
```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         80.2877      4.000     20.071      0.000      72.441      88.135
CC            20.8955      4.170      5.011      0.000      12.716      29.075
Days           0.1261      0.014      9.120      0.000       0.099       0.153
Web           12.4811      9.054      1.378      0.168      -5.280      30.242
==============================================================================
```

Figure 11.2 Python validation of the regression results with the test data set.



```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         73.6209      4.657     15.808      0.000      64.485      82.757
CC            22.1357      4.738      4.672      0.000      12.842      31.429
Days           0.1637      0.017      9.784      0.000       0.131       0.197
==============================================================================
```

Figure 11.3 Final regression model for the training data set using Python.



```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         80.7656      3.986     20.260      0.000      72.946      88.586
CC            21.5262      4.146      5.192      0.000      13.393      29.659
Days           0.1254      0.014      9.071      0.000       0.098       0.152
==============================================================================
```

Figure 11.4 Validating the final regression model with the test data set using Python.

We therefore rerun the regression model, this time omitting *Web Account* from the model. The results for the training set and test set are shown in Figures 11.3 and 11.4. Using the coefficients from the training set, we obtain our final regression model as:

$$\begin{aligned}\textit{Sales per Visit} = 73.6209 + 0.1637(\textit{Days between purchases})\\ + 22.1357(\textit{Credit Card})\end{aligned}$$

That is, the estimated sales per visit for our customer base is \$73.6209 plus \$0.1637 times the number of days between purchases plus \$22.1357 if they have a store credit card. We see that customers tend to spend more if they have a store credit card. Also, the longer it has been between visits, the more customers tend to spend.

We interpret these *regression coefficients* as follows:

- **Credit Card.** The estimated increase in Sales per Visit for a customer with a store credit card (compared to a customer without a store credit card) is \$22.1357, when Days between Purchase is held constant.
- **Days Between Purchase.** For each increase of one day in the average days between purchases, the estimated increase in Sales per Visit is \$0.1637, when Credit Card is held constant. This can be better understood if we compare two shoppers, Customer A and Customer B, where Customer A has an average number of days between purchases one month (30 days) longer than Customer B. Then, Customer A's Sales per Visit is $30 \times \$0.1637 = \$4.91$ greater than Customer B, holding Credit Card constant.

11.4 HOW TO PERFORM MULTIPLE REGRESSION MODELING USING PYTHON

First, as always, we load the required packages.

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
```

Next, we import the *clothing_sales_training* and *clothing_sales_test* data sets as *sales_train* and *sales_test*, respectively.

```
sales_train = pd.read_csv("C:/.../clothing_sales_
training.csv")
sales_test = pd.read_csv("C:/.../clothing_sales_test.csv")
```

For simplicity, we separate the predictor variables and the target variable. We call the data frame of predictor variables **X** and the target variable **y**.

```
X = pd.DataFrame(sales_train[['CC', 'Days', 'Web']])
y = pd.DataFrame(sales_train[['Sales per Visit']])
```

To have a constant term b_0 in our regression model, we need to add a constant variable to our predictor variables.

```
X = sm.add_constant(X)
```

Running the **add_constant()** command on the **X** variables will add a column to the data frame filled with the value one (1.0).

Finally, we run the multiple regression model.

```
model01 = sm.OLS(y, X).fit()
```

OLS stands for "Ordinary Least Squares," which is the method used to fit this regression model. Note that the two inputs of the **OLS()** command are the target variable **y** and the predictor variables **X**. Save the fitted model as **model01**. To obtain the results of the regression model, run the **summary()** command on **model01**.

```
model01.summary()
```

An excerpt from the output of the **summary()** command is shown in Figure 11.1. The regression coefficients are located in the *coef* column.

To verify the regression model results, we run the same code on the *sales_test* data set. The code is given below, the explanations equivalent to those given earlier in this section.

```
X_test = pd.DataFrame(sales_test[['CC', 'Days', 'Web']])
y_test = pd.DataFrame(sales_test[['Sales per Visit']])
X_test = sm.add_constant(X_test)
model01_test = sm.OLS(y_test, X_test).fit()
model01_test.summary()
```

An excerpt from the results from the **summary()** command run on **model01_test** is given in Figure 11.2. The results validate the results from **model01**.

To remove the variable *Web* from the regression model, we redefine the **X** data frame to include only the remaining two predictor variables. After doing so, we also need to add the constant term back into our predictor variable data frame.

```
X = pd.DataFrame(sales_train[['CC', 'Days']])
X = sm.add_constant(X)
```

Once the predictor variable data frame is ready, we run the **OLS()** and **fit()** commands on the target variable **y** and new **X** data frame again. Note that we did not change the **y** input, since only the **X** input needed to change. Save the new model as **model02** and run the **summary()** command on **model02** to view the results.

```
model02 = sm.OLS(y, X).fit()
model02.summary()
```

An excerpt from the output of the **model02.summary()** command is shown in Figure 11.3.

To verify this smaller model, we run similar code on the test data. Once again, the explanations are similar to those for the preceding example.

```
X_test = pd.DataFrame(sales_test[['CC', 'Days']])
X_test = sm.add_constant(X_test)
model02_test = sm.OLS(y_test, X_test).fit()
model02_test.summary()
```

An excerpt of the output from the **model02_test.summary()** command is shown in Figure 11.4.

11.5 HOW TO PERFORM MULTIPLE REGRESSION MODELING USING R

Load the *clothing_sales_training* and *clothing_sales_test* data sets as *sales_train* and *sales_test*, respectively. Next, make sure the binary variables are factors in both data sets.

```
sales_train$CC <- as.factor(sales_train$CC)
sales_train$Web <- as.factor(sales_train$Web)
sales_test$CC <- as.factor(sales_test$CC)
sales_test$Web <- as.factor(sales_test$Web)
```

Now, run the full model for the training data set.

```
model01 <- lm(formula = Sales.per.Visit ~ Days + Web +
CC, data = sales_train)
```

Notice the two pieces of input that are required: **formula** and **data**. The **formula** takes the same *Target ~ Predictors* form we have seen before. The **data = sales_train** input specifies the data set that our variables come from. We save the results of the regression modeling under the name **model01**. To view a summary of the model results, run the **summary()** command on **model01**.

```
summary(model01)
```

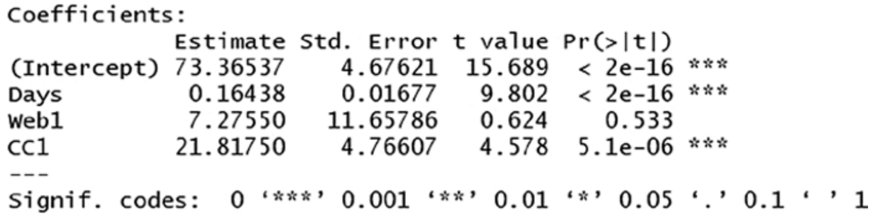
An excerpt of the output generated by **summary(model01)** is shown in Figure 11.5.

To validate the model, change the data input to specify that the variables now come from the *sales_test* data set.

```
model01_test <- lm(formula = Sales.per.Visit ~ Days +
Web + CC, data = sales_test)
```

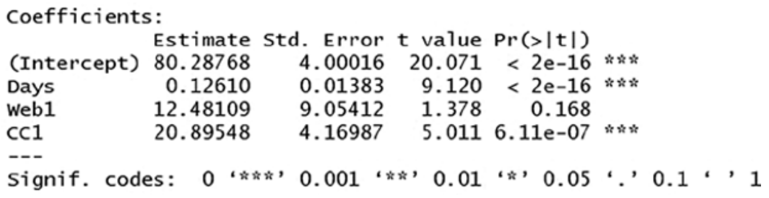
To view the regression summary of this new model, run **summary (model01_test)**. An excerpt of the output generated by this command is shown in Figure 11.6.

To remove variables from the model, remove their names from the series of predictor variables to the right of the tilde within the **lm()** command.



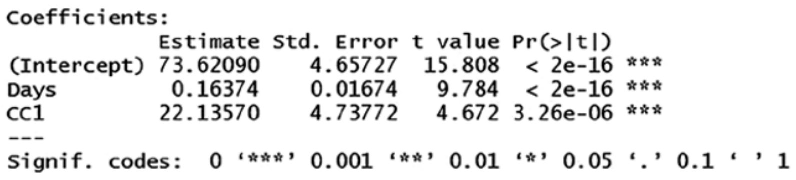
```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 73.36537    4.67621  15.689  < 2e-16 ***
Days         0.16438    0.01677   9.802  < 2e-16 ***
Web1         7.27550   11.65786   0.624    0.533    
CC1         21.81750    4.76607   4.578  5.1e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 11.5 Regression results from R for the training data set.



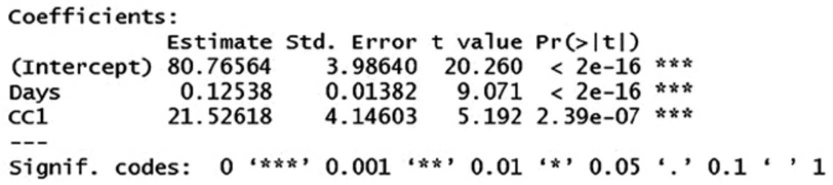
```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 80.28768    4.00016  20.071  < 2e-16 ***
Days         0.12610    0.01383   9.120  < 2e-16 ***
Web1        12.48109    9.05412   1.378    0.168
CC1         20.89548    4.16987   5.011 6.11e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 11.6 Validating the regression results from R with the test data set.



```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 73.62090    4.65727  15.808  < 2e-16 ***
Days         0.16374    0.01674   9.784  < 2e-16 ***
CC1         22.13570    4.73772   4.672 3.26e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 11.7 Final regression results from R for the training data set.



```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 80.76564    3.98640  20.260  < 2e-16 ***
Days         0.12538    0.01382   9.071  < 2e-16 ***
CC1         21.52618    4.14603   5.192 2.39e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 11.8 Validating the final regression results from R with the test data set.

The commands to run the new models and generate the summary output are given below.

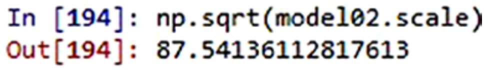
```
model02 <- lm(formula = Sales.per.Visit ~ Days + CC,
data = sales_train)
summary(model02)
model02_test <- lm(formula = Sales.per.Visit ~ Days +
CC, data = sales_test)
summary(model02_test)
```

Notice that excerpts of the output generated by **summary(model02)** and **summary(model02_test)** are shown in Figures 11.7 and 11.8, respectively.

11.6 MODEL EVALUATION FOR ESTIMATION

We can use the regression equation to make predictions (estimates) of sales per visit. For example, consider Customer 1 in the training set, who goes 333 days between purchases and who does not have a store credit card (*Credit Card* = 0). Plugging these values into the regression equation, we obtain:

$$\textit{Sales per Visit} = 73.62 + 0.1637(333) + 22.14(0) = \$128.13$$



```
In [194]: np.sqrt(model02.scale)
Out[194]: 87.54136112817613
```

Figure 11.9 The standard error of the estimate from Python.

That is, using the regression model, we would estimate the average sales per visit for this customer to be $\hat{y} = \$128.13$. However, the *actual* sales per visit for this customer is $y = \$184.23$. So, the *prediction error* (residual) for this customer is:

$$\textit{prediction error} = \left(y - \hat{y}\right) = 184.23 - 128.13 = \$56.10$$

So, this customer spends \$56.10 more than expected, given his or her days between visits and credit card status.

The typical size of the prediction error is given by the statistic *s*, the standard error of the estimate.

$$s = \sqrt{MSE} = \sqrt{\frac{SSE}{n-p-1}} = \sqrt{\frac{\sum\left(y-\hat{y}\right)^2}{n-p-1}}$$

Here, $s = \$87.54$, meaning that the size of the model's typical prediction error is about \$87.54, as can be seen in Figure 11.9. This large value is due to the fact that our data is excerpted from a much larger data set, including dozens more predictors, useful for making our model's estimates more precise. Usually, however, *s* is a very important metric for measuring the efficacy of a regression model.

In its derivation, *s* squares the prediction errors, thereby possibly endowing outliers with undue influence in the magnitude of the statistic. Data scientists should therefore compare *s* with the *Mean Absolute Error* (MAE), given by

$$\textit{Mean Absolute Error} = \frac{\sum\left|y-\hat{y}\right|}{n}$$

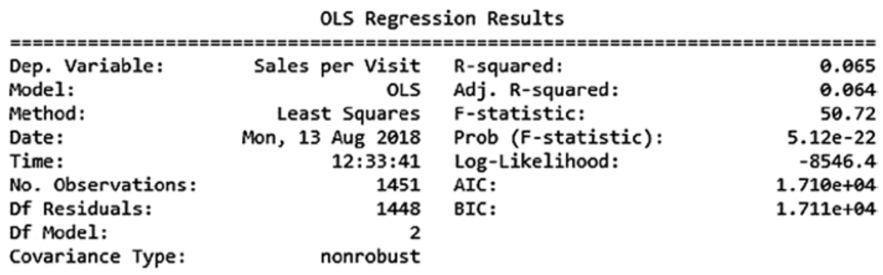
The MAE takes the *distance* between the actual and predicted values of *y* and finds the average of these distances. There is no squaring going on. As for any model evaluation statistics, we should do the following:

1. Develop the regression model using the training data set.
2. Calculate the *MAE* by passing the test data set through the model trained on the training data set.

For the training data set, we have $MAE = \$53.39$.

Estimation Model Metrics

When evaluating estimation models, always report both *s* and *MAE*.



```
                            OLS Regression Results
==============================================================================
Dep. Variable:         Sales per Visit   R-squared:                       0.065
Model:                             OLS   Adj. R-squared:                  0.064
Method:                  Least Squares   F-statistic:                     50.72
Date:                 Mon, 13 Aug 2018   Prob (F-statistic):           5.12e-22
Time:                         12:33:41   Log-Likelihood:                -8546.4
No. Observations:                 1451   AIC:                         1.710e+04
Df Residuals:                     1448   BIC:                         1.711e+04
Df Model:                            2
Covariance Type:             nonrobust
```

Figure 11.10 R^2_{adj} from Python for the final regression model.

Finally, R^2 is a well-known regression metric. It is interpreted as the proportion of the variability in the response that is accounted for by the predictors in the model. For multiple regression models, analysts should use R^2_{adj}, which penalizes R^2 for having too many unhelpful predictors in the model. Our regression model has $R^2_{adj} = 0.064$, as seen in Figure 11.10. That is, 6.4% of the variability in *Sales per Visit* is accounted for by the predictors *Days since Purchase* and *Credit Card*. This small proportion is not surprising, since there are many other factors affecting how much customers spend.

11.6.1 How to Perform Estimation Model Evaluation Using Python

To use the regression model to predict customer sales per visit, we first need to specify the variable values for the first customer for the Python regression model. As the variables in the model are in the order *Constant, CC, Days*, this is the order in which we specify the values.

```
cust01 = np.column_stack((1, 0, 333))
```

The first input in the **column_stack()** command is 1, for the constant term in the model.

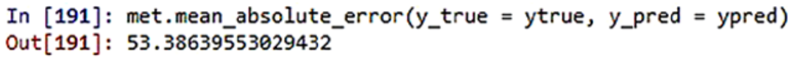
Once you have constructed the customer in question, run the **predict()** command on **cust01**. Since we are predicting the sales using the results stored in **model02**, we use **model02.predict()**.

```
model02.predict(cust01)
```

To obtain the predicted values of all customers in the test data set, change the input of the **predict()** command to the test data predictor variable data frame, **X_test**.

```
ypred = model02.predict(X_test)
```

The result is a column of predictions, one for each record in the test data set. These values will allow us to calculate the MAE later in this section.



```
In [191]: met.mean_absolute_error(y_true = ytrue, y_pred = ypred)
Out[191]: 53.38639553029432
```

Figure 11.11 The MAE from Python.

Python does not automatically supply the standard error of the estimate. However, it can be calculated using the square root of the scale parameter of the model.

```
np.sqrt(model02.scale)
```

To calculate the MAE, we need both the predicted and actual values of y. The actual values are the values of the target variable, renamed below as **ytrue** for clarity. The **ypred** values come from the code above.

```
ytrue = sales_train[['Sales per Visit']]
met.mean_absolute_error(y_true = ytrue, y_pred = ypred)
```

The final output from the code is about 53.39, the value of the MAE. The code, with this result, is shown in Figure 11.11.

To obtain the value of R^2_{adj}, examine the output from the **summary()** command demonstrated in the previous Python section, and shown in Figure 11.10.

11.6.2 How to Perform Estimation Model Evaluation Using R

To use our model to predict the sales per visit of a particular customer, we build a data frame containing that customer's information.

```
cust01 <- data.frame(CC = as.factor(0), Days = 333)
```

The command **data.frame()** will create a data frame using the input contents. The variables' names must exactly match the names of the predictor variables in the model. Since the credit card variable were factors when we built the model, make sure they are factors when creating this new customer. Save this new customer data as **cust01**. Note that we did not include the target variable.

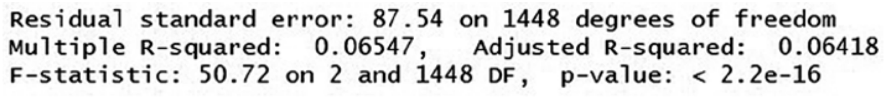
```
predict(object = model02, newdata = cust01)
```

When we run the **predict()** command using **object = model02** and **newdata = cust01**, the output is the predicted number of sales per visit.

The standard error of the estimate is given as part of the output generated by the **summary()** command. Figure 11.12 shows an excerpt of the output generated by **summary(model02)**. The important statistic *s* is called the "Residual standard error" in this output, and is reported for this model as 87.54 in Figure 11.12, along with the value of R^2_{adj}, called "Adjusted R-squared."

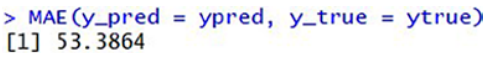
To calculate the MAE, we need the actual and predicted values for all records in the test data set using the training data model.

```
X_test <- data.frame(Days = sales_test$Days, CC = sales_
test$CC)
ypred <- predict(object = model02, newdata = X_test)
```



```
Residual standard error: 87.54 on 1448 degrees of freedom
Multiple R-squared:  0.06547,   Adjusted R-squared:  0.06418
F-statistic: 50.72 on 2 and 1448 DF,  p-value: < 2.2e-16
```

Figure 11.12 Standard error and Adjusted R-squared from R.



```
> MAE(y_pred = ypred, y_true = ytrue)
[1] 53.3864
```

Figure 11.13 The MAE from R.

```
ytrue <- sales_test$Sales.per.Visit
```

We also need to install and open the *MLmetrics* package.

```
install.packages("MLmetrics"); library(MLmetrics)
```

Once the package is open, you can calculate the MAE.

```
MAE(y_pred = ypred, y_true = ytrue)
```

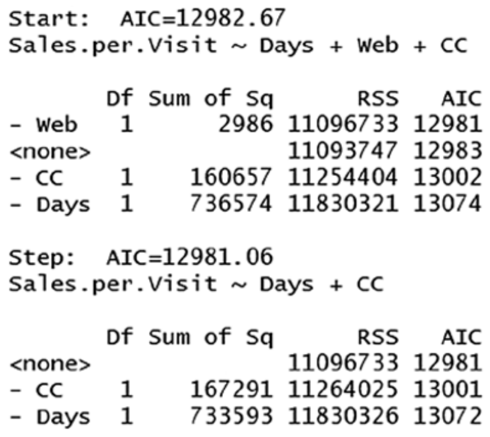
The two inputs of the **MAE()** command are **y_pred** and **y_true**. Set **y_pred = ypred**, the values you obtained from the regression model; and set **y_true = ytrue**, the target variable from the training data set. The result of running this command is the MAE, the code and output for which are shown in Figure 11.13.

11.7 STEPWISE REGRESSION

In this small example, we had only three predictors. But, most data science projects use dozens if not hundreds of predictors. We therefore need a method to ease the selection of the best regression model. This method is called *stepwise regression*. In stepwise regression, helpful predictors are entered into the model one at a time, starting with the most helpful predictor. Because of multicollinearity or other effects, when several helpful variables are entered, one of them may no longer be considered helpful any more, and should be dropped. For this reason, stepwise regression adds the most helpful predictors into the model one at a time and then checks to see if they all still belong. Finally, the stepwise algorithm can find no further helpful predictors and converges to a final model.

The application of stepwise regression (not shown) to the *clothing_sales_training* and *clothing_sales_test* data sets converged on the final models displayed in Figures 11.3 and 11.4. It is important to understand that stepwise regression is not guaranteed to uncover the optimal model, as its search algorithm does not perform all possible regressions. To guarantee the optimal model, you can use *best subsets regression*,[2] though the software may limit the number of predictors to the best subsets algorithm.

[2]See *Data Mining and Predictive Analytics*.



```
Start:  AIC=12982.67
Sales.per.Visit ~ Days + Web + CC

       Df Sum of Sq      RSS   AIC
- Web   1      2986 11096733 12981
<none>              11093747 12983
- CC    1    160657 11254404 13002
- Days  1    736574 11830321 13074

Step:  AIC=12981.06
Sales.per.Visit ~ Days + CC

       Df Sum of Sq      RSS   AIC
<none>              11096733 12981
- CC    1    167291 11264025 13001
- Days  1    733593 11830326 13072
```

Figure 11.14 The output from stepwise regression in R.

11.7.1 How to Perform Stepwise Regression Using R

To run stepwise regression, you first need to install and open the *MASS* package.

```
install.packages("MASS"); library(MASS)
```

Run the regression model, including all variables under consideration. Save the model under a name. For this example, we will use **model01**, which we know to have a variable, *Web*, that does not belong in the model.

Once you have saved the model, it is time for stepwise regression.

```
model01_step <- stepAIC(object = model01)
```

The **stepAIC()** command will run stepwise regression on the object specified. For our example, we want to run stepwise regression on **model01**. We save the result under the name **model01_step**.

Even saving the **stepAIC()** output under the name will show some output, given in Figure 11.14. The output shows the steps taken to converge on a model. Moving from the top half to the bottom half of the output shows that the stepwise algorithm took one step. Namely, it removed the variable *Web*.

If you run the name **model01_step** by itself, it will give you the regression coefficients of the final model. If you run **summary(model01_step)**, it will give the full summary of the final model, which will match the output given by **summary(model02)**, since the final model converged to by stepwise is the regression model we saved as **model02**.

11.8 BASELINE MODELS FOR REGRESSION

The usual baseline model to compare your regression model against is the simple $y = \bar{y}$ model. If any of the predictors are helpful at all in estimating the response, then the model will beat the $y = \bar{y}$ model. Nevertheless, we should still formally verify that our regression model (or any estimation model) is outperforming the $y = \bar{y}$ model, as follows:[3]

Baseline Model Comparison for Estimation Models

1. Calculate the errors made by the baseline model. These take the form $Error = y - \bar{y}$.
2. Compute the MAE for the baseline model, as follows:
$$MAE_{Baseline} = \frac{\sum |y - \bar{y}|}{n}$$
3. Compare $MAE_{Baseline}$ to the MAE for the estimation model.
$$MAE_{Regression} = \frac{\sum |y - \hat{y}|}{n}$$
4. The estimation model outperforms the baseline model when
$$MAE_{Regression} < MAE_{Baseline}$$

We apply the Baseline Model Comparison to our final regression model (Figure 11.3) as follows:

1. We calculate the errors for the baseline model using the $\bar{y} = \$112.57$ provided by the test data set.
2. We compute $MAE_{Baseline} = \$55.53$.
3. The $MAE_{Regression}$ we obtained by passing the test data set through the model developed by the training data set is \$53.39.
4. Since $\$53.39 = MAE_{Regression} < MAE_{Baseline} = \$55.53$, our regression model did beat the baseline model.

REFERENCES

The *MASS* package shares the same core publication that we have seen for the *nnet* package: W. N. Venables and B. D. Ripley, *Modern Applied Statistics with S*, Fourth Edition, Springer, New York, 2002.

The *MLmetrics* package, which allowed us to obtain the MAE in R, can be further explored here: Yachen Yan, MLmetrics: Machine Learning Evaluation Metrics. R package version 1.1.1. 2016. https://CRAN.R-project.org/package=MLmetrics

[3]Some data scientists may prefer to compare $MSE_{Regression}$ with $MSE_{Baseline}$.

EXERCISES

CLARIFYING THE CONCEPTS

1. How does multiple regression approximate the relationship between a set of two predictors and a single numeric target?
2. Explain how we are bypassing the classical statistical inference approach to regression.
3. Explain why it is not necessary to standardize the predictors when there is only one continuous predictor and the others are flags?
4. True or false: Our use of p-values as guides for determining inclusion in the model means that we are using statistical inference. If false, explain why not.
5. For the training set results in Figure 11.3, suppose two customers both had a store credit card, but Customer A had 100 more days between purchases than Customer B. Describe the difference in the two customers' estimated sales per visit.
6. For the training set results in Figure 11.3, suppose two customers both had the same days between purchases, but Customer C had a store credit card and Customer D did not. Describe the difference in the two customers' estimated sales per visit.
7. Calculate the prediction error for Customer 2 in the training set.
8. Calculate s for the test data set.
9. Calculate the MAE for the test data set.
10. True or false: Stepwise regression always finds the optimal set of predictors.

WORKING WITH THE DATA

For the following exercises, work with the *clothing_sales_training* and *clothing_sales_test* data sets. Use either Python or R to solve each problem.

11. Use the training set to run a regression model to predict Sales per Visit using Days between purchases, Credit card, and Web account. Identify which predictor variable should not be in the model.
12. Validate the model from the previous exercise, by running the regression using the test data set.
13. Suppose someone said, "There is no evidence for a relationship between *Sales per Visit* and whether the customer has a store credit card." How would you respond?
14. Suppose someone said, "There is no evidence for a relationship between *Sales per Visit* and whether the customer has a store web account." How would you respond?
15. Run a regression model to predict Sales per Visit, using only the variables found to be significant in the previous regression model.
16. Validate the model from the previous exercise.
17. Use the regression equation to complete this sentence: "The estimated Sales per Visit equals…."

18. Calculate and interpret the standard error of the estimate for the regression.

19. Find and interpret R^2_{adj}.

20. Calculate and interpret the MAE for the regression model. Compare it to the standard error.

21. Perform stepwise regression on the model in Exercise 11. Confirm that it converges to the model in Exercise 13.

22. Calculate $MAE_{Baseline}$.

23. Compute $MAE_{Regression}$.

24. Determine whether the regression model outperformed its baseline model.

HANDS-ON ANALYSIS

For the following exercises, work with the *adult* data set. Use either Python or R to solve each problem.

25. Partition the data set into a training set and a test set, each containing about half of the records.

26. Run a regression model to predict Hours per Week using Age and Education Num. Obtain a summary of the model. Are there any predictor variables that should not be in the model?

27. Validate the model from the previous exercise.

28. Use the regression equation to complete this sentence: "The estimated Hours per Week equals…."

29. Interpret the coefficient for *Age*.

30. Interpret the coefficient for *Education Num*.

31. Find and interpret the value of s.

32. Find and interpret R^2_{adj}.

33. Find $MAE_{Baseline}$ and $MAE_{Regression}$, and determine whether the regression model outperformed its baseline model.

For the following exercises, work with the *bank_reg_training* and the *bank_reg_test* data sets. Use either Python or R to solve each problem.

34. Use the training set to run a regression predicting *Credit Score*, based on *Debt-to-Income Ratio* and *Request Amount*. Obtain a summary of the model. Do both predictors belong in the model?

35. Validate the model from the previous exercise.

36. Use the regression equation to complete this sentence: "The estimated Credit Score equals…."

37. Interpret the coefficient for *Debt-to-Income Ratio*.

38. Interpret the coefficient for *Request Amount*.

39. Find and interpret the value of *s*.

40. Find and interpret R^2_{adj}. Comment.

41. Find $MAE_{Baseline}$ and $MAE_{Regression}$, and determine whether the regression model outperformed its baseline model.

42. Construct a regression model for predicting *Interest*, using *Request Amount*. Obtain a summary of the model.

43. Explain what is unusual with your results from the previous exercise.

44. Construct a scatterplot of *Interest* against *Request Amount*. Describe the relationship between the variables. Explain how this relationship explains the unusual results from your regression model.

For the following exercises, work with the *Framingham_training* and the *Framingham_test* data sets. Reexpress *Sex* so that it is a flag variable with 0 for males and 1 for females. Use either Python or R to solve each problem.

45. Use the training set to run a regression predicting *Age*, based on *Sex* and *Education*. Obtain a summary of the model. Do both predictors belong in the model?

46. Validate the model from the previous exercise.

47. Use the regression equation to complete this sentence: "The estimated Age equals…."

48. Interpret the coefficient for *Sex*.

49. Interpret the coefficient for *Education*.

50. Find and interpret the value of *s*.

51. Find and interpret R^2_{adj}.

52. Find $MAE_{Baseline}$ and $MAE_{Regression}$, and determine whether the regression model outperformed its baseline model.

For the following exercises, work with the *white_wine_training* and the *white_wine_test* data sets. Use either Python or R to solve each problem.

53. Use the training set to run a regression predicting *Quality*, based on *Alcohol* and *Sugar*. Obtain a summary of the model. Do both predictors belong in the model?

54. Validate the model from the previous exercise.

55. Use the regression equation to complete this sentence: "The estimated Quality equals…."

56. Interpret the coefficient for *Alcohol*.

57. Interpret the coefficient for *Sugar*.

58. Find and interpret the value of *s*.

59. Find and interpret R^2_{adj}.

60. Find $MAE_{Baseline}$ and $MAE_{Regression}$, and determine whether the regression model outperformed its baseline model.