Classification: Logistic Regression Part 2: Estimating and Evaluating the Performance of a Logistic Classifier

Aram Balagyozyan

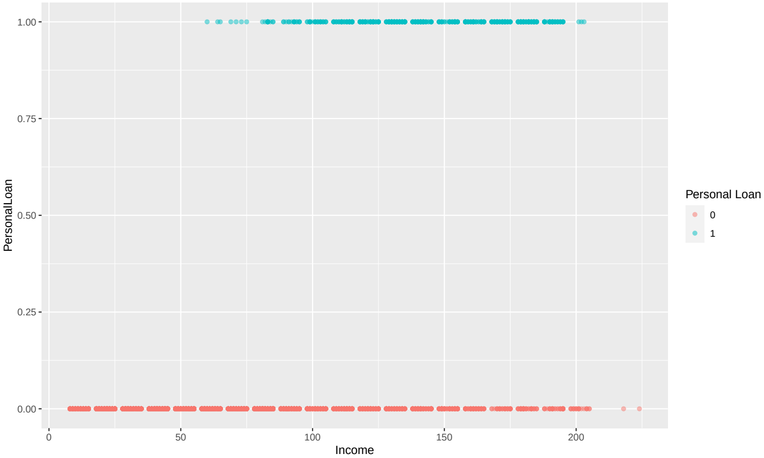
Department of Operations and Information Management
Kania School of Management
The
University of Scranton

August 9, 2020

1. Estimating a Logistic Regression Using R
2. Evaluating Classification Performance

Estimating a Simple Logistic Regression Model Using R

- As an example, let's construct a simple regression model for classification of customers using only one predictor, *Income*.



- There aren't any low-income customers who have accepted a loan offer but there are many high-income customers who have not.

Estimating a Simple Logistic Regression Model Using R

- The equation relating the outcome variable to the predictor in terms of probabilities is:

$$P(PersonalLoan = Yes) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 Income)}}$$

- Or equivalently, in terms of odds:

$$Odds(PersonalLoan = Yes) = e^{\beta_0 + \beta_1 Income}$$

- To estimate the logistic regression coefficients, use the *glm()* function of the base package:

```
simple.logitreg<-glm(PersonalLoan ~Income,
  data = bank.df, family = "binomial")
options(scipen=999)
summary(simple.logitreg)
```

Estimating a Simple Logistic Regression Model Using R

term	estimate	std.error	statistic	p.value
(Intercept)	-6.1273153	0.1860778	-32.92878	0
Income	0.0371252	0.0013786	26.92987	0

- The estimation results imply that:

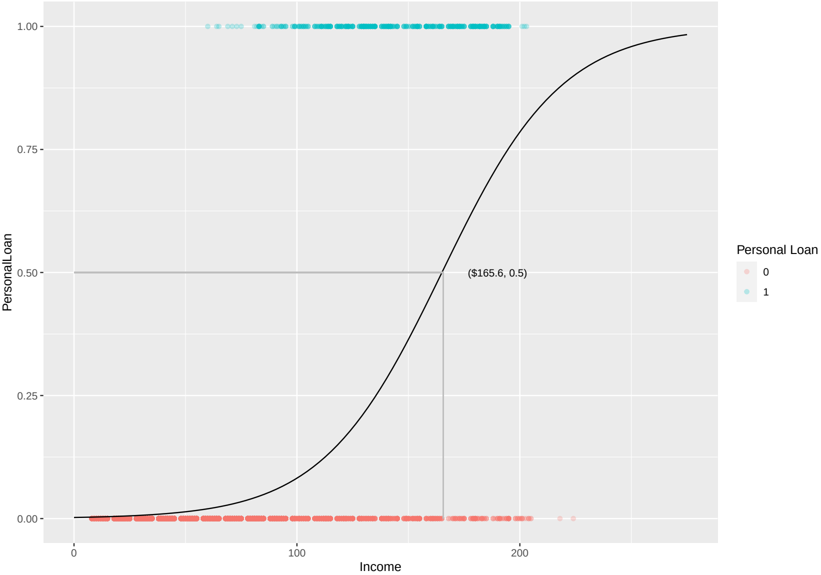
$$P(\mathit{PersonalLoan} = \mathit{Yes}) = \frac{1}{1 + e^{-(-6.127+0.037\mathit{Income})}}$$

- The odds that a customer with income zero will accept the loan is is estimated by $e^{-(-6.127+0.037\times 0)} \approx 0.0022$. These are the base case odds ($\approx 1/500$).
- The odds of accepting the loan with an income of 100 thousand dollars will increase by a multiplicative factor of $e^{0.037*100} = 40.45$ over the base, so the odds that such a customer will accept the offer are $e^{-6.127+0.037\times 100} \approx 0.08830$, approximately $1/11$.

Estimating a Simple Logistic Regression Model Using R

- The estimated logistic regression equation on the previous slide produces the loan-acceptance probability for a customer with a given income (see the graph on the next slide). To end up with classification of a customer into either 1 or 0 (e.g., a customer either accepts the loan offer or not), we need to decide on a threshold, or cutoff value for the probability.
- Thus, in order to classify a new customer as an acceptor/nonacceptor of the loan offer, we use the information on his/her income by plugging it into the fitted equation above. This yields an estimated probability (propensity) of accepting the loan offer (the s-shaped curve on the next slide is the graph of the propensities for various income levels). We then compare it to a assumed cutoff value. The customer is classified as an acceptor if the probability of his/her accepting the offer is above the cutoff.

Estimating a Simple Logistic Regression Model Using R



Estimating a Simple Logistic Regression Model Using R

- As an example, suppose that you decide that the cutoff probability of $P(Y = 1) = 0.5$ is a good threshold. Thus for a customer with \$175,000 annual income, the loan acceptance propensity is equal to:

$$P(PersonalLoan = Yes) = \frac{1}{1 + e^{-(-6.127+0.037\times 175)}} = 0.5863$$

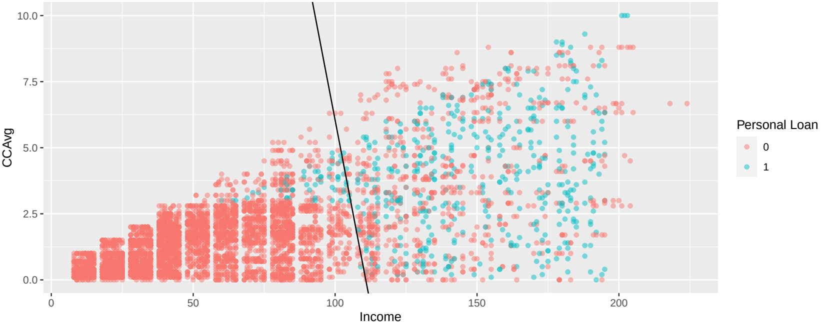
- Since the predicted propensity for anyone with income \$175,000 is $0.5863 > 0.5$, you will classify the customer as a loan acceptor.
- With cutoff probability $= 0.5$, anyone with Income > \$165,595 will be classified as acceptor.
- It should be obvious from the example and graph that the current simple classifier is quite poor at making correct classification calls; there are quite a few non-acceptors with income higher than \$165,595 and there are many acceptors with income lower income than \$165,595

Separation Line Between Two Classes

- Suppose instead of having only one predictor (`Income`), you had two predictors in the logistic equation:

$$P(PersonalLoan = Yes) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 Income + \beta_2 CCavg)}}$$

- Given the estimated values of the intercept and slope coefficients in the logistic regression equation above ($\widehat{\beta}_0$, $\widehat{\beta}_1$, and $\widehat{\beta}_2$) and your decision on the cutoff probability (p_{cut}), you can easily derive the formula of the separation line on the graph below.



Formula of the Separation Line Between Two Classes

- The formula of the separation line can be derived as follows:
- Plug in the known values of $\widehat{\beta}_0$, $\widehat{\beta}_1$, $\widehat{\beta}_2$, and p_{cut} into the logistic regression equation:

$$p_{cut} = \frac{1}{1 + e^{-(\widehat{\beta}_0 + \widehat{\beta}_1 Income + \widehat{\beta}_2 CCavg)}}$$

- Solve the above for `CCAvg`

$$CCAvg = \frac{1}{\widehat{\beta}_2}(-\widehat{\beta}_0 - \ln(\frac{1}{p_{cut}} - 1)) - \frac{\widehat{\beta}_1}{\widehat{\beta}_2} Inc$$

- The formula of the separation line is

$$CCavg = b + m \times Inc$$

where $b = \frac{1}{\widehat{\beta}_2}(-\widehat{\beta}_0 - \ln(\frac{1}{p_{cut}} - 1))$ and $m = -\frac{\widehat{\beta}_1}{\widehat{\beta}_2}$

Two Motivating Questions for What's Coming

- By now you should have some questions. Some obvious ones are:
  1. What is the correct cutoff probability.
  2. How does one assess the performance of a classifier?
- In what follows, we will try to address some of these questions.

Estimating a (Bigger) Logistic Regression Model Using R

- Normally, one wouldn't hope to build a powerful classifier using a single predictor. It is more appropriate to include other predictors in the model as well.
- Also, as we've done it before, it would only be prudent to divide the dataset into a training set (to fit a logistic regression) and validation set (to assess the model's performance).
- Finally, note that the predictor variable `Education` takes on integer values 1, 2, or 3. We need to convert it into a factor variable. When a predictor is a factor with 3 levels, R's `glm()` function automatically converts it into two dummy variables (disregarding one additional dummy to avoid the multicolinearity issue). On the other hand, when the variable is an integer, `glm()` will not recognize the need for the conversion and may produce unreliable estimates.

Estimating a (Bigger) Logistic Regression Model Using R

- The code below converts the Education variable into a factor, divides the dataset into a training set and validation set, and fits a logistic regression to the training set:

```r
#convert the Education variable into a factor
bank.df$Education <- factor(
  bank.df$Education, levels = c(1, 2, 3),
  labels = c("Undergrad", "Graduate",
         "Advanced/Professional"))
# partition data
set.seed(2)
train.index <- sample(c(1:dim(bank.df)[1]),
                   dim(bank.df)[1]*0.6)
train.df <- bank.df[train.index, ]
valid.df <- bank.df[-train.index, ]
# run logistic regression using all predictors
logit.reg <- glm(PersonalLoan ~ ., data=train.df,
                 family = "binomial")
```

Estimating a (Bigger) Logistic Regression Model Using R

- Below is the summary output

```r
options(scipen = 999) #avoid scientific notation
summary(logit.reg)
```

term	estimate	std.error	statistic	p.value
(Intercept)	-15.5820754	2.2351307	-6.9714382	0.0000000
Age	0.0743198	0.0804261	0.9240753	0.3554471
Experience	-0.0593021	0.0801234	-0.7401348	0.4592182
Income	0.0627328	0.0040045	15.6656869	0.0000000
Family	0.5476224	0.0978787	5.5949101	0.0000000
CCAvg	0.1651545	0.0588724	2.8052951	0.0050271
EducationGraduate	4.2286088	0.3614010	11.7006009	0.0000000
EducationAdvanced/Professional	4.2208138	0.3622092	11.6529713	0.0000000
Mortgage	0.0011339	0.0007789	1.4558598	0.1454314
SecuritiesAccount	-0.7064409	0.3820337	-1.8491583	0.0644350
CDAccount	3.5878387	0.4345389	8.2566562	0.0000000
Online	-0.5602502	0.2161742	-2.5916608	0.0095514
CreditCard	-1.2225669	0.2842447	-4.3011078	0.0000170

Interpreting Logistic Regression Coefficients

- The output above implies that the fitted logistic regression model looks as follows:

$$
\begin{aligned}
&P(PersonalLoan = Yes) = Logit(\\
&\quad -15.58 + 0.07 Age \\
&\quad -0.06 Experience + 0.06 Income + 0.55 Family \\
&\quad +0.17 CCAvg + 4.23 EducationGraduate \\
&\quad +4.22 EducationAdvanced/Professional \\
&\quad +0.001 Mortgage - 0.71 SecuritiesAccount \\
&\quad +3.59 CDAccount - 0.56 Online \\
&\quad -1.22 CreditCard)
\end{aligned}
$$

Interpreting Logistic Regression Coefficients

- Logistic models, can give useful information about the roles played by different predictor variables. For example, we may be interested in how increasing family income by one unit will affect the probability of acceptance.
- Note that the change in the probability, p, for a unit increase in a particular predictor variable, while holding all other predictors constant, is not a constant, it depends on the specific values of the predictor variables. E.g. if we increase `Age` from 22 to 23, the effect on `p` will be different that if we increase `Age` from 23 to 24.
- Thus, In terms of predicted probabilities (propensities), the coefficients warrant only qualitative interpretations E.g., using the sign to assess the direction of the effect.
- As we alluded above, when instead of propensities odds are considered, the coefficients can be interpreted in a straight-forward way .

Interpreting Coefficients in Terms of Propensities

- The positive coefficients for the dummy variables `EducationGraduate`, `EducationAdvanced/Professional`, and `CDAccount` mean that holding a CD account and having graduate or professional education (all marked by 1 in the dummy variables) are associated with higher probabilities of accepting the loan offer.
- In contrast, having a securities account, using online banking, and owning a Universal Bank credit card are associated with lower acceptance rates.
- For the continuous predictors, positive coefficients indicate that a higher value on that predictor is associated with a higher probability of accepting the loan offer (e.g., `Income`: higher-income customers tend more to accept the offer).
- Similarly, negative coefficients indicate that a higher value on that predictor is associated with a lower probability of accepting the loan offer (e.g., `Experience`: customers with more professional experience are less likely to accept the offer).

Interpreting Coefficients in Terms of Odds

- Earlier, when we considered a single-predictor logit model, we elaborated on the interpretation of the coefficients in terms of odds. Those interpretations change only slightly when more than one predictors is involved.
- Recall that odds are given by

$$Odds = e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_q x_q}$$

- e^{β_i} is the multiplicative factor by which the odds (of belonging to class 1) increase when the value of i-th predictor x_i increases by one unit, *holding all other predictors constant*.

Interpreting Coefficients in Terms of Odds

- E.g., the intercept in the estimated equation above is -15.58. This means that the base case odds are $e^{-15.58} \approx 0$. This means that $Odds = P(PersonalLoan = Yes) \approx 0$ when all $x_i = 0$.
- The coefficient of `Income` is 0.06. This means that a $\$100K$ increase in income and keeping all other predictors constant and equal to zero, will increase the odds of accepting a loan by a factor of $e^{0.06} \times 100 = 106.18$.

Interpreting Coefficients in Terms of Odds

- When a predictor is a dummy variable, the interpretation is technically the same but has a slightly different practical meaning.
- For instance, the coefficient for `CDAccount` was estimated from the data to be 3.610057.
- We interpret this coefficient as follows: $e^{3.610057} = 36.97$ are the odds that a customer who has a CD account will accept the offer relative to a customer who does not have a CD account, holding all other variables constant. This means that customers who hold CD accounts at Universal Bank are more likely to accept the offer than customers without a CD account (holding all other variables constant).

Judging Classifier Performance (Schmueli et al.,2018, Section 5.3)

- A natural criterion for judging the performance of any classifier is the probability of making a misclassification error on the validation set.
- Misclassification means that the record belongs to one class but the model classifies it as a member of a different class.
- You must have notice from the credit acceptance exercise above that there are two ways in which the model can get things wrong. It may classify some records as likely acceptances that turn out to be non-acceptances (false positives) and it may label some records as likely non-acceptances turn out to be actual acceptances (false negatives).
- In practice, most accuracy measures are derived from the *confusion matrix*, also called *classification matrix*. This matrix summarizes the number of correct and incorrect classifications that a classifier produced for a certain dataset.

Judging Classifier Performance: Confusion Matrix

- Consider the validation set of the loan acceptance data. To assess the performance of the model we trained on the training set, let's produce the predicted probability that the records in the validation set will accept the loan offer, $\widehat{P}(LoanOffer = Yes)$.

```r
# use predict() with type = "response" to compute
# predicted probabilities.
logit.reg.pred <- predict(logit.reg,
                          newdata=valid.df,
                          type = "response")
ActPred.df<-
  data.frame(actualClass = valid.df$PersonalLoan,
         predictedProb = logit.reg.pred)
```

Judging Classifier Performance: Confusion Matrix

```r
options(scipen = 999)
head(ActPred.df)
```

```
##    actualClass  predictedProb
## 1            0 0.000128278630
## 3            0 0.000005182088
## 4            0 0.117204971114
## 6            0 0.003390944766
## 7            0 0.023748906445
## 16           0 0.000251333626
```

- The first column of `ActPred.df` reflects whether each customer in the validation set has actually accepted the loan offer. The second column is our logistic model's predicted probability that each customer will accept the offer.
- If we decide to set the cutoff probability at 0.5, then all of the 6 records above would be (correctly) classified as non-acceptors (since all of the predicted probabilities are less than 0.5).

Judging Classifier Performance: Confusion Matrix

- The `confusionMatrix()` function in the **caret** package produces the confusion matrix.

```
# createa a colum with predicted class
ActPred.df$predictedClass<-
    ifelse(ActPred.df$predictedProb > 0.5,1,0)
#run confusionMarix()
# need to convert numeric to factor
library(caret)
confMat<-confusionMatrix(
            factor(ActPred.df$predictedClass),
            factor(ActPred.df$actualClass))
confMat$table
```

```
##           Reference
## Prediction    0    1
##          0 1794   65
##          1   18  123
```

Interpreting Confusion Matrix

- Rows and columns of the confusion matrix correspond to the counts of predicted and true (actual) classes, respectively.
- The two diagonal cells (upper left, lower right) give the number of correct classifications, where the predicted class coincides with the actual class of the record.
- The off-diagonal cells give counts of misclassification.
- The top-right cell gives the number of class 1 members (acceptors) that were misclassified as 0's (nonacceptors, in this example, there were 65 such misclassifications).
- Similarly, the lower-left cell gives the number of class 0 members (non-acceptors) that were misclassified as 1's (acceptors, 125 such records).

Accuracy Rates Derived from the Confusion Matrix

- Several accuracy measures can be derived from the confusion matrix above. Two of the most obvious ones are the **overall accuracy** and **overall error** rates:
  1. **Overall Accuracy Rate** (or simply accuracy): the fraction of correctly classified records:

$$\mathit{accuracy} = \frac{1794 + 123}{1794 + 123 + 18 + 65} = 0.9585$$

  2. **Overall Error (or missclassification) Rate**: the fraction of incorrectly classified records:

$$\mathit{error} = 1 - \mathit{accuracy} = \frac{18 + 65}{1794 + 123 + 18 + 65} = 0.0415$$

Accuracy Rates Derived from the Confusion Matrix

- In general, for any estimated confusion matrix:

		Reference (Actual) Class	
		C_1	C_2
Predicted Class	C_1	$n_{1,1}$	$n_{2,1}$
	C_2	$n_{1,2}$	$n_{2,2}$

$$accuracy = \frac{n_{1,1} + n_{2,2}}{n_{1,1} + n_{1,2} + n_{2,1} + n_{2,2}}$$

$$error = \frac{n_{1,2} + n_{2,1}}{n_{1,1} + n_{1,2} + n_{2,1} + n_{2,2}}$$

Cutoff Probabilities and Accuracy Rates

- Note that that numbers in the confusion matrix and, therefore, the accuracy and error rates depend on our choice of the cutoff probability.
- The question is then, what is the appropriate cutoff probability?
- To answer this question, let's let's create an accuracy table for different cutoff probabilities and visualize the results on a plot.

Cutoff Probabilities and Accuracy Rates

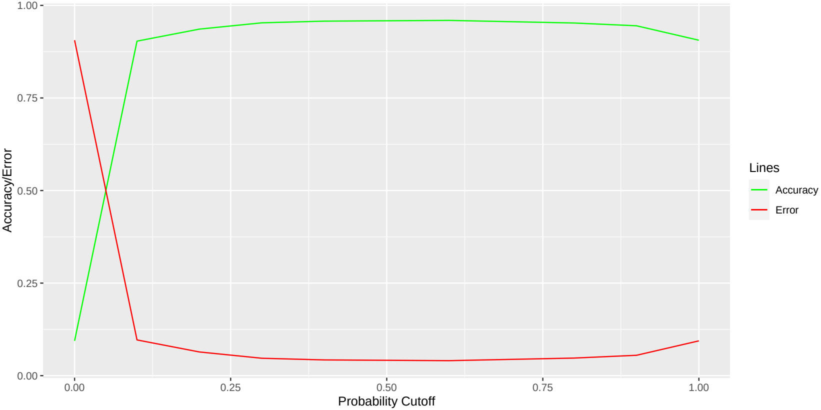
```r
#Create an empty accuracy data frame
accTab<-data.frame(Cut=numeric(),Accuracy=numeric())
# compute accuracy for different cutoffs
for (cut in seq(0,1,0.1)){ #for each cut
#get the predicted class
  ActPred.df$predictedClass<-
          ifelse(ActPred.df$predictedProb>cut,1,0)
#Get the confusion matrix
  RollConfMat<-confusionMatrix(
                factor(ActPred.df$predictedClass),
                factor(ActPred.df$actualClass))
#Get the accuracy from confusionMatrix()
  RollAcc<-data.frame(Cut=cut,
                 Accuracy=RollConfMat$overall[1])
  row.names(RollAcc)<-c() #eliminate the rowname
  #Add accuracy to the data frame
  accTab<-rbind(accTab,RollAcc)
}
```

Cutoff Probabilities and Accuracy Rates

- Plot the accuracy and error rates against different cutoff probabilities.

```r
ggplot(data = accTab, aes(x=Cut, y=Accuracy))+
  geom_line(aes(color="Accuracy"))+
  geom_line(aes(x=Cut,y=1-Accuracy,color="Error"))+
  scale_color_manual(values =
                c( 'Accuracy' = 'green',
                    'Error' = 'red')) +
  labs(x="Probability Cutoff", y="Accuracy/Error",
   color="Lines")
```

Cutoff Probabilities and Accuracy Rates



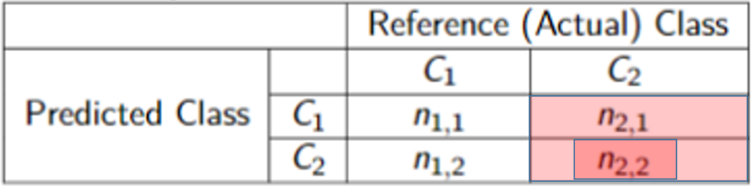
- The accuracy/error plot implies that raising the cutoff probability above 0.3 leaves the classification accuracy (error) rate relatively unchanged, peaking at around 0.6.
- Thus, it is optimal to choose cutoff probability = 0.6 since it maximizes classification accuracy and minimizes error.

Classifier Performance in Case of Unequal Importance of Classes

- The costs of missclassification of a certain class may be significantly higher than the cost of missclassification of the other class(es). For example, in predicting the financial status (bankrupt/solvent) of firms, it may be more important to predict correctly a firm that is going bankrupt than to predict correctly a firm that is going to remain solvent.
- In our case, it is probably more important to predict the acceptance of a loan offer than non-acceptance. If a potential acceptance is misclassifies as a non-acceptance, the Bank forgoes a potentially lucrative profit opportunity. If a potential non-acceptance is classified as an acceptance then the cost of missclassification may be minuscule (e.g. the cost of promotional items and mailing).
- In such cases, the overall accuracy (error) rate is not a good measure for evaluating the classifier. **Sensitivity** and **Specificity** are more appropriate measures of classifier performance.

Sensitivity (or recall)

- **The Sensitivity** (also termed recall) of a classifier is its ability to detect the important class members correctly. It is the percentage of all the actual members of the important class that were also predicted by the classifier to belong to that class. If C_2 is the important class then



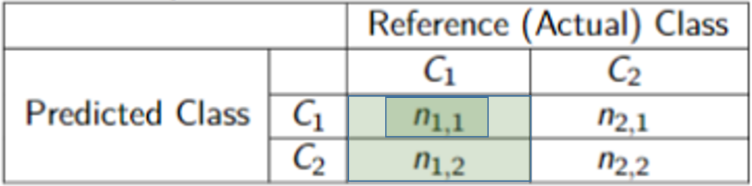
		Reference (Actual) Class	
		C_1	C_2
Predicted Class	C_1	$n_{1,1}$	$n_{2,1}$
	C_2	$n_{1,2}$	$n_{2,2}$

$$Sensitivity = \frac{n_{2,2}}{n_{1,2} + n_{2,2}}$$

- In our case, when the cutoff probability is set at 0.5, sensitivity $= 123/(65 + 123) = 65.43$. Only 65.43% of all customers who actually accepted the loan offer were also predicted by the model as acceptors. This is pretty low sensitivity considering the costs.

Specificity

- The specificity of a classifier is its ability to rule out the members of the unimportant class correctly. It is the percentage of all the actual members of the unimportant class that were also predicted by the classifier to belong to that class. If C_1 is the unimportant class then



		Reference (Actual) Class	
		C_1	C_2
Predicted Class	C_1	$n_{1,1}$	$n_{2,1}$
	C_2	$n_{1,2}$	$n_{2,2}$

$$Specificity = \frac{n_{1,1}}{n_{1,1} + n_{1,2}}$$

- In our case, when the cutoff probability is set at 0.5, specificity $= 1794/(1794 + 18) = 99.01\%$. 99.01% of all the customers who actually did not accept the loan offer were also predicted by the model as non-acceptors. This is pretty high specificity.

Lift Charts

- Sensitivity and Specificity are very useful metrics for comparing different classification models. Another very useful way of evaluating a classifier is comparing its performance with a completely random classifier.
- Lift charts (a.k.a.lift curves, gains curves, or gains charts) are a very useful graphical tool for comparing the results of the classifier with a) completely ignorant and b) perfect classifiers. They are also very useful for visualizing the performance of two or more classifiers.
- The inputs required to construct a lift curve are is the response variable in the validation curve and the score (predicted probability that each observation belongs to the important class).
-

Lift Charts

- Going back to our case, we would like our classification model to sift through the records and sort them according to which ones are most likely to be responders.
- The logistic model will give us an estimate of the extent to which we will encounter more and more non-responders as we proceed through the sorted data starting with the records most likely to be responders. We can use the data sorted propensity to decide to which potential customers a limited-budget mailing should be targeted.
- In other words, we are describing the case when our goal is to obtain a rank ordering among the records according to their class membership predicted probabilities (propensities).

Lift Charts

- Suppose a hypothetical validation set (below, sorted by propensities) contains only 16 records.

Record	propensity	PersonalLoan
1	0.995976726	1
2	0.948110638	1
3	0.889297203	1
4	0.847631864	1
5	0.762806287	0
6	0.706991915	1
7	0.680754087	1
8	0.505506928	1
9	0.47134045	0
10	0.337117362	0
11	0.21796781	1
12	0.199240432	0
13	0.047962588	1
14	0.024850999	0
15	0.016129906	0
16	0.003559986	0

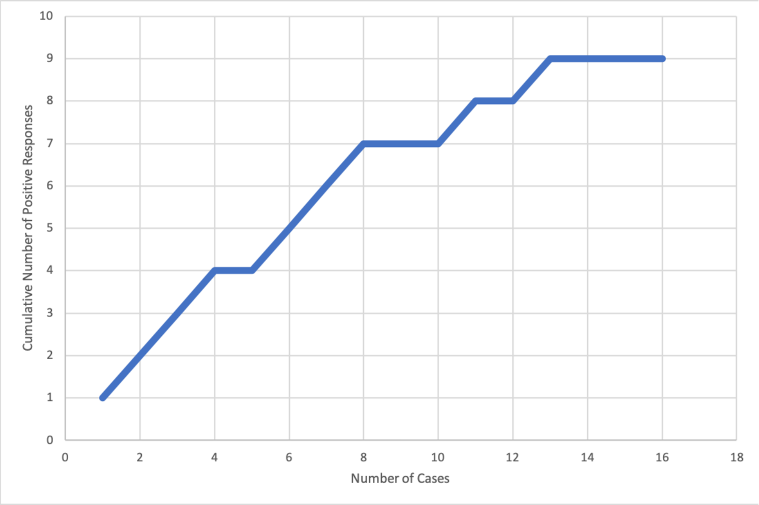
Lift Charts

- Add to the table below another column with the cumulative number of positive responses:

Record	propensity	PersonalLoan	Cum. Positive Response
1	0.995976726	1	1
2	0.948110638	1	2
3	0.889297203	1	3
4	0.847631864	1	4
5	0.762806287	0	4
6	0.706991915	1	5
7	0.680754087	1	6
8	0.505506928	1	7
9	0.47134045	0	7
10	0.337117362	0	7
11	0.21796781	1	8
12	0.199240432	0	8
13	0.047962588	1	9
14	0.024850999	0	9
15	0.016129906	0	9
16	0.003559986	0	9

Interpreting a Lift Chart

- A lift chart is a plot of the first and last columns of the table on the previous slide, with the first column being the variable on the horizontal axis and the fourth column being the variable on the vertical axis.



Interpreting a Lift Chart

- Obviously, the classifier above is not perfect. Although high propensities predict positive responses, at least one record with a high propensity has a negative response (record 5). On the flip side, a couple of records with a low propensity have a positive response (records 11 and 13).
- If the classifier was perfect, it would assign high propensities to **all** the records with positive responses. All the records with negative responses would be assigned low propensities.
- On the other hand, an ignorant classifier's assignment of propensities would be random. Positive (as well as negative) responses could be found among the records with both high and low propensities.

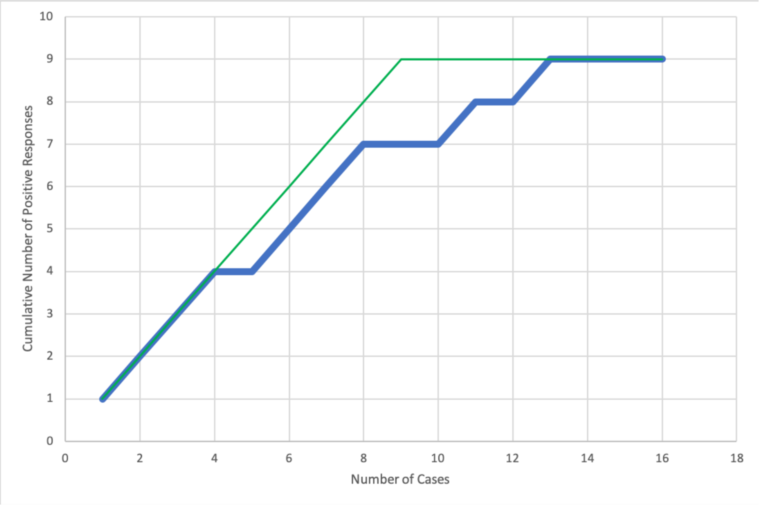
Lift Chart of a Perfect Classifier

- A perfect classifier would produce propensities that should look like the following:

Record	propensity	PersonalLoan	Cum. Positive Response
1	0.995976726	1	1
2	0.948110638	1	2
3	0.889297203	1	3
4	0.847631864	1	4
5	0.762806287	1	5
6	0.706991915	1	6
7	0.680754087	1	7
8	0.505506928	1	8
9	0.47134045	1	9
10	0.337117362	0	9
11	0.21796781	0	9
12	0.199240432	0	9
13	0.047962588	0	9
14	0.024850999	0	9
15	0.016129906	0	9
16	0.003559986	0	9

Lift Chart of a Perfect Classifier

- Thus, the lift chart of a perfect classifier looks like the green line in the chart below. It increases at a 45 degree line then becomes perfectly flat.



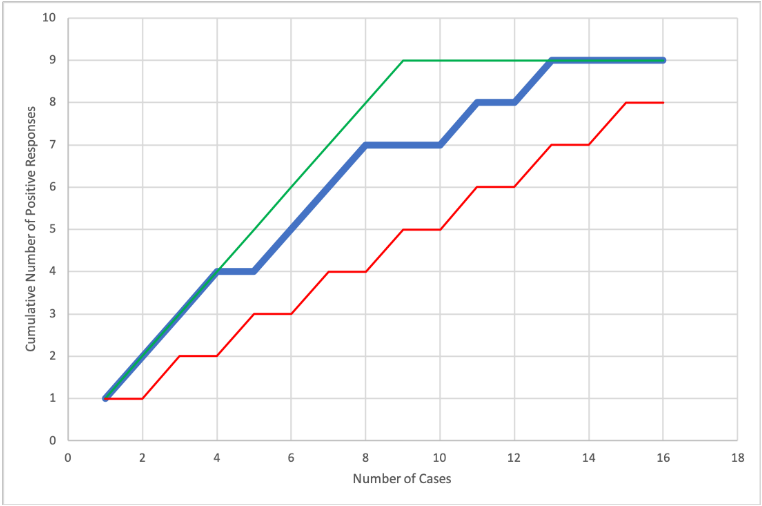
Lift Chart of an Ignorant Classifier

- An ignorant classifier assigns random propensities. Thus, it will produce propensities that should look like the following:

Record	propensity	PersonalLoan	Cum. Positive Response
1	0.995976726	1	1
2	0.948110638	0	1
3	0.889297203	1	2
4	0.847631864	0	2
5	0.762806287	1	3
6	0.706991915	0	3
7	0.680754087	1	4
8	0.505506928	0	4
9	0.47134045	1	5
10	0.337117362	0	5
11	0.21796781	1	6
12	0.199240432	0	6
13	0.047962588	1	7
14	0.024850999	0	7
15	0.016129906	1	8
16	0.003559986	0	8

Lift Chart of an Ignorant Classifier

- Thus, the lift chart of an ignorant classifier looks like the red line in the chart below. When the validation set is large, it is a straight line that starts from the initial point of the lift chart and ends at the last point of the lift chart.



Interpretation of a Lift Chart: Summary

- The “lift” over the the base curve indicates for a given number of cases (read on the *x*-axis), the additional responders (or percentage of responders) that you can identify by using the model rather than selecting records at random.
- The flatter the lift chart, the worse the classifier is.
- The closer the lift chart to the outer (perfect) lift chart, the better the classifier is.
- Between two classifiers, the one that results in a more concave lift chart the better it is.

Plotting a Lift Chart in R

- There are different ways of plotting a lift chart in R. The textbook relies on the **gains** library. Since we already have almost all the needed components, let's build a lift chart from scratch.
- The following code builds a lift chart from scratch. It plots the *percentage* rather than *number* of positive responses on the vertical axis.

Plotting a Lift Chart in R

```r
library(dplyr)
#sort ActPred.df by predicted probabilities
# in descending order
sortedActPred.df <- arrange(ActPred.df,-predictedProb)
# the total number of responses in the valid. set
TotNumOfResp <- dim(sortedActPred.df)[1]
#add a new column for the total num of positives
sortedActPred.df$CumActualPos<-numeric(TotNumOfResp)
#populate the column with total num of positives
for (i in 1:TotNumOfResp){
  sortedActPred.df$CumActualPos[i]<-
    sum(sortedActPred.df$actualClass[1:i])
}
```

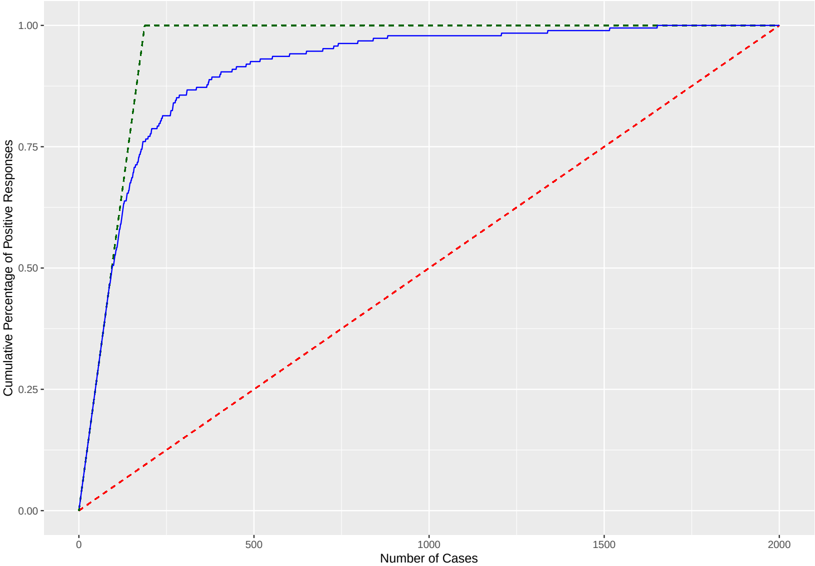
Plotting a Lift Chart in R

```
#what to plot
#Total number of actual positives in the valid. set
NumOfActualPos<-sum(sortedActPred.df$actualClass)
#create a dataframe with the needed data to plot
liftchart.df<-data.frame(
                  NumOfResp=1:TotNumOfResp,
CumPosResp = sortedActPred.df$CumActualPos/
                                    NumOfActualPos)
```

Plotting a Lift Chart in R

```r
#plot using ggplot
ggplot(data = liftchart.df, aes(x=NumOfResp,
                                y=CumPosResp))+
  geom_segment(aes(x = 0, y = 0,
                   xend = TotNumOfResp, yend = 1),
            color="red", size=0.5, lty="dashed")+
  geom_segment(aes(x = 0, y = 0,
                xend = NumOfActualPos , yend = 1),
      color="darkgreen", size=0.5, lty="dashed")+
  geom_segment(aes(x = NumOfActualPos, y = 1,
                   xend = TotNumOfResp , yend = 1),
      color="darkgreen", size=0.5, lty="dashed")+
  geom_line(color="blue")+
  labs(x="Number of Cases",
    y="Cumulative Percentage of Positive Responses")
```

Plotting a Lift Chart in R



Decile Chart

- The information from the lift chart can be portrayed as a *decile chart* which is widely used in direct marketing predictive modeling.
- The decile chart is a bar chart that aggregates all the lift information into 10 buckets.
- The height of the bars show the factor by which our model outperforms a random assignment of 0's and 1's, taking one decile at a time.
- The (somewhat lengthy) code below produces a decile chart for our logistic classifier.

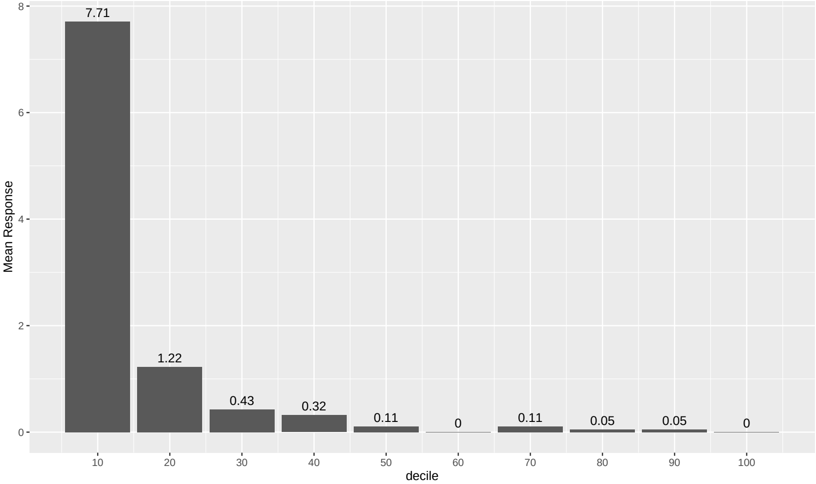
Decile Chart

```r
library(ggplot2)
#for each observation obtain weighted response
# by dividing actual response by average response
heights.df <-
  data.frame(height=sortedActPred.df[,1]/
               mean(valid.df$PersonalLoan))
#create a coumn with deciles 1 to 10 for all obs.
decile<-
  data.frame(decile=cut(1:TotNumOfResp,10,
                        labels=FALSE))
#combine the weighted response and decile
#columns into one data frame
heights.df<-cbind(heights.df,decile)
```

Decile Chart

```r
# obtain mean weighted response for each decile
data.for.plot <- aggregate(heights.df$height,
by = list(heights.df$decile),FUN = mean)
# change the column names
names(data.for.plot)<-c("decile","height")
#plot
ggplot(data.for.plot, aes(x=decile, y = height)) +
geom_col()+ geom_text(aes(label=round(height,2)),
                                  vjust = -0.5)+
ylab("Mean Response")+
  scale_x_continuous(breaks=1:10,
                     labels = 1:10*10)
```

Decile Charts



- Taking the 10% of the records that are ranked by the model as “most probable 1’s” yields 7.7 times as many 1’s as would simply selecting 10% of the records at random.