Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Business Analytics using Data Mining & Forecasting

BU7143 & BU7144

Dr. Nicholas P. Danks
Business Analytics
Email address

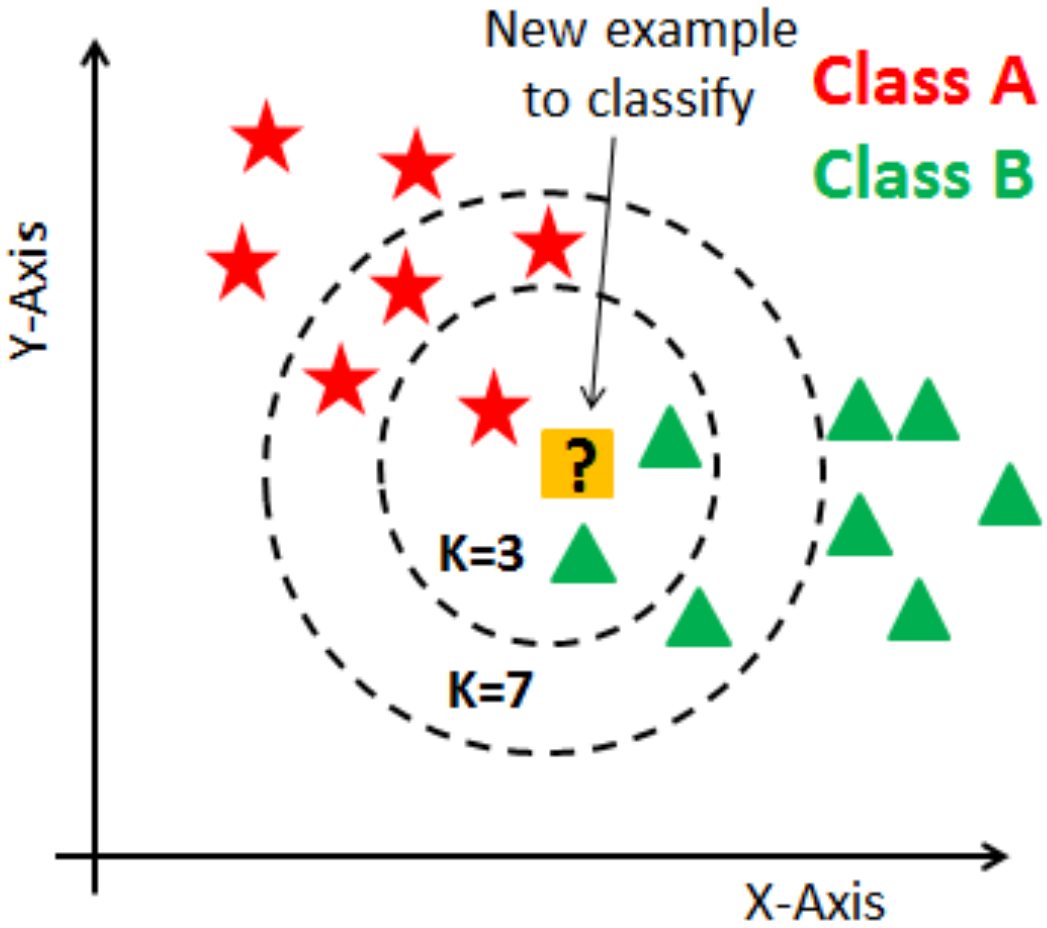
Overview of Today's Session

1. K-Nearest Neighbours
2. Logistic Regression

Part I
K-NN

Characteristics of K-NN

- Data-driven, not model-driven
- Makes no assumptions about the data



Basic Idea

- For a given record to be classified, identify nearby records
- "Near" means records with similar predictor values $X_1, X_2, \ldots X_p$
- Classify the record as whatever the predominant class is among the nearby records (the "neighbors")

How to measure “nearby”?

A lot more “distances” than you think:
- Manhattan
- Euclidean
…
https://en.wikipedia.org/wiki/Distance

The most popular distance measure is **Euclidean distance**

$$\sqrt{(x_1 - u_1)^2 + (x_2 - u_2)^2 + \cdots + (x_p - u_p)^2}$$

- Typically, predictor variables are first normalized (= standardized) to put them on comparable scales
- Use preProcess() from caret package to normalize
- Otherwise, metrics with large scales dominate

Choosing k

- *K* is the number of nearby neighbors to be used to classify the new record
  - *K*=1 means use the single nearest record
  - *K*=5 means use the 5 nearest records

- Typically choose that value of *k* which has lowest error rate in validation data

Low *k* vs. High *k*

- Low values of *k* (1, 3, …) capture local structure in data (but also noise)
- High values of *k* provide more smoothing, less noise, but may miss local structure

  Note: the extreme case of k = n (i.e., the entire data set) is the same as the "naïve rule" (classify all records according to majority class)

Example: Riding Mowers

Data: 24 households classified as owning or not owning riding mowers

Predictors: Income, Lot Size

Income	Lot_Size	Ownership
60.0	18.4	owner
85.5	16.8	owner
64.8	21.6	owner
61.5	20.8	owner
87.0	23.6	owner
110.1	19.2	owner
108.0	17.6	owner
82.8	22.4	owner
69.0	20.0	owner
93.0	20.8	owner
51.0	22.0	owner
81.0	20.0	owner
75.0	19.6	non-owner
52.8	20.8	non-owner
64.8	17.2	non-owner
43.2	20.4	non-owner
84.0	17.6	non-owner
49.2	17.6	non-owner
59.4	16.0	non-owner
66.0	18.4	non-owner
47.4	16.4	non-owner
33.0	18.8	non-owner
51.0	14.0	non-owner
63.0	14.8	non-owner

Finding nearest neighbors in R

- Library FNN provides a list of neighbors
- Library `class` allows numerical output
- See Table 7.2 for code using knn from FNN library
- compares each record from validation* (or test) set to k nearest records in training
- Use library `caret` to get accuracy of different values of k, applied to validation data (see next slide for code)

Output

```
> accuracy.df
    k accuracy
1   1      0.7
2   2      0.7
3   3      0.8
4   4      0.9
5   5      0.8
6   6      0.9
7   7      0.9
8   8      1.0
9   9      0.9
10 10      0.9
11 11      0.9
12 12      0.8
13 13      0.4
14 14      0.4
```

- Even is possible for ties to occur
- R breaks ties randomly

KNN Code For Riding Mower Example

```
library(caret)

# initialize a data frame with two columns: k, and accuracy.
accuracy.df <- data.frame(k = seq(1, 14, 1),
                          accuracy = rep(0, 14))

# compute knn for different k on validation.
for(i in 1:14) {
  knn.pred <- knn(train.norm.df[, 1:2],
                  valid.norm.df[, 1:2],
                  cl = train.norm.df[, 3],
                  k = i)
  accuracy.df[i, 2] <- confusionMatrix(knn.pred,
                                       valid.norm.df[,3])$overall[1]
}
```

Using K-NN for Prediction (for Numerical Outcome)

- Instead of “majority vote determines class” use average of response values
- May be a weighted average, weight decreasing with distance

Advantages

- Simple
- No assumptions required about Normal distribution, etc.
- Effective at capturing complex interactions among variables without having to define a statistical model

Shortcomings

- Required size of training set increases exponentially with # of predictors, *p*
  - This is because expected distance to nearest neighbor increases with *p* (with large vector of predictors, all records end up "far away" from each other)
- In a large training set, it takes a long time to find distances to all the neighbors and then identify the nearest one(s)
- These constitute "curse of dimensionality"

Dealing with the Curse

- Reduce dimension of predictors (e.g., with PCA)
- Computational shortcuts that settle for "almost nearest neighbors"

Summary

- Find distance between record-to-be-classified and all other records
- Select k-nearest records
  - Classify it according to majority vote of nearest neighbors
  - Or, for prediction, take the as average of the nearest neighbors
- “Curse of dimensionality” – need to limit # of predictors

Part II
Logistic Regression

Logistic Regression

- Extends idea of linear regression to situation where outcome variable is categorical
- Widely used, particularly where a structured model is useful to explain (=*profiling*) or to predict
- We focus on binary classification
  - i.e. Y=0 or Y=1

The Logit

- **Goal:** Find a function of the predictor variables that relates them to a 0/1 outcome
- Instead of *Y* as outcome variable (like in linear regression), we use a function of Y called the ***logit***
- Logit can be modeled as a linear function of the predictors
- The logit can be mapped back to a probability, which, in turn, can be mapped to a class

Step 1: Logistic Response Function

p = probability of belonging to class 1

Need to relate p to predictors with a function that guarantees $0 \leq p \leq 1$

Standard linear function (as shown below) does not:

$$p = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots \beta_q x_q$$

The Fix: use ***logistic response function***

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots \beta_q x_q)}}$$

Step 2: The Odds

The odds of an event are defined as:

$$Odds = \frac{p}{1-p}$$ ← p = probability of event

Or, given the odds of an event, the probability of the event can be computed by:

$$p = \frac{Odds}{1+Odds}$$

We can also relate the Odds to the predictors:

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

Step 3: Take log on both sides

This gives us the logit:

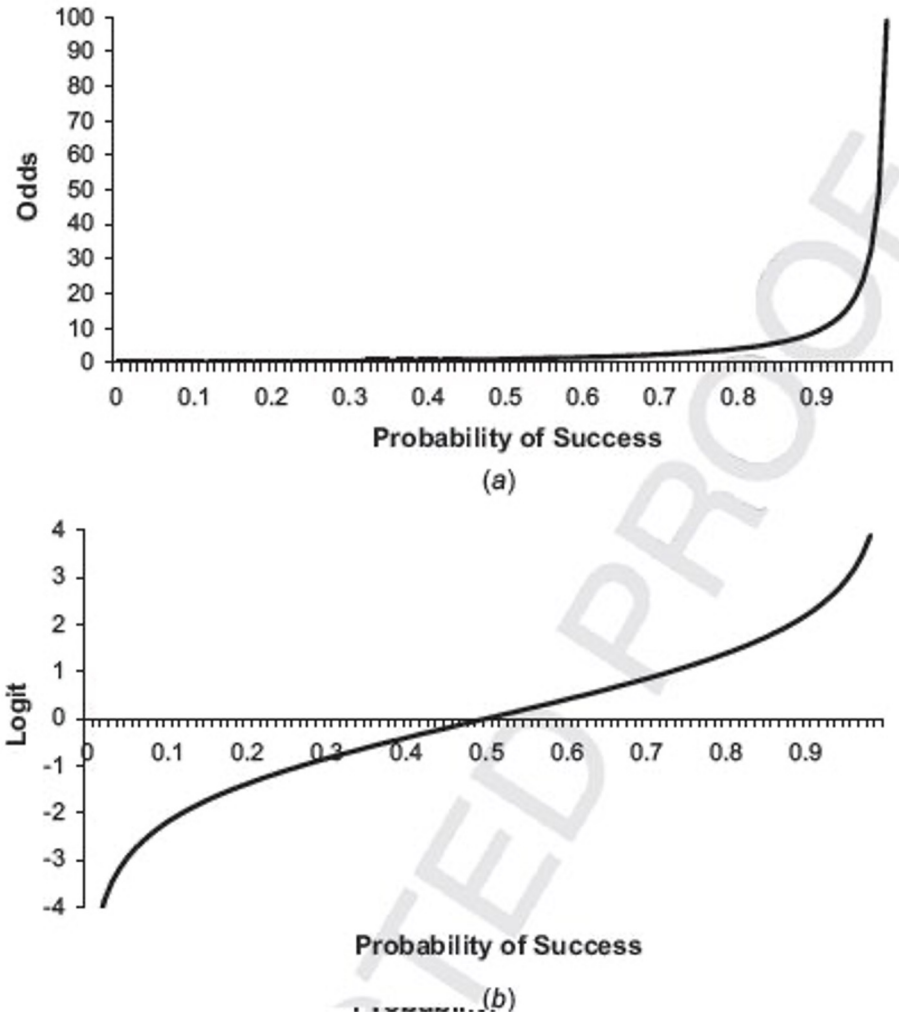
$$\log(Odds) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \square + \beta_q x_q$$

log(Odds) = *logit* (eq. 10.6)

So, the logit is a linear function of predictors x_1, x_2, …

- Takes values from -infinity to +infinity

Odds (a) and Logit (b) as function of *P*



Example

Data Prep

Outcome variable: accept bank loan (0/1)

Predictors: Demographic info, and info about their bank relationship

```
bank.df <- read.csv("UniversalBank.csv")
bank.df <- bank.df[ , -c(1, 5)]  # Drop ID and zip code columns.
# treat Education as categorical (R will create dummy variables)
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, ]
```

training partition of 60%

Single Predictor Model

Modeling loan acceptance on income (x)

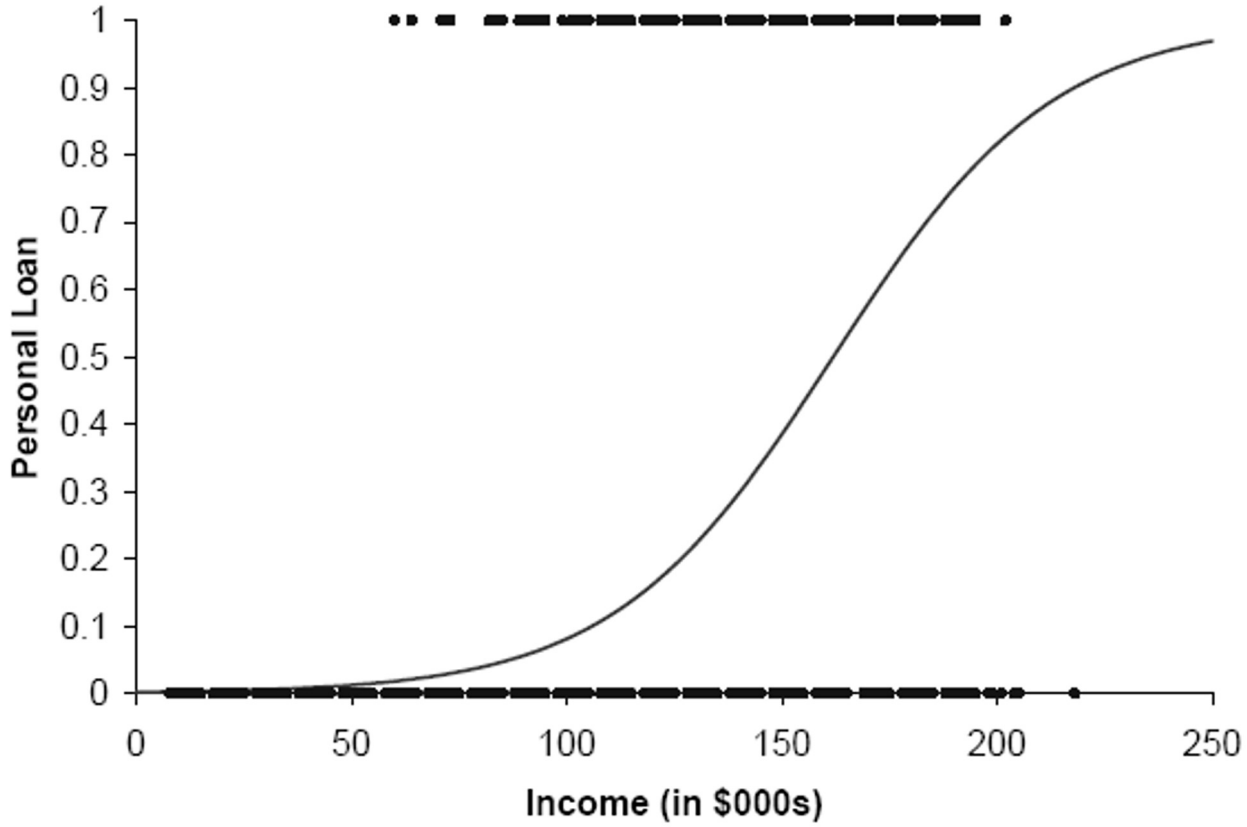
$$\text{Prob}(Personal\ Loan = Yes \mid Income = x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

Fitted coefficients (more later): $b_0 = -6.3525$, $b_1 = -0.0392$

$$P(Personal\ Loan = Yes \mid Income = x) = \frac{1}{1 + e^{6.3525 - 0.0392x}}$$

Seeing the Relationship

$$P(Personal\ Loan = Yes \mid Income = x) = \frac{1}{1 + e^{6.3525 - 0.0392x}}$$



Last step - classify

Model produces an estimated probability of being a “1”

- Convert to a classification by establishing cutoff level
- If estimated prob. > cutoff, classify as “1”

Ways to Determine Cutoff

- 0.50 is popular initial choice
- Additional considerations (see Chapter 5)
  - Maximize classification accuracy
  - Maximize sensitivity (subject to min. level of specificity)
  - Minimize false positives (subject to max. false negative rate)
  - Minimize expected cost of misclassification (need to specify costs)

Example, cont.

- Estimates of β's are derived through an iterative process called *maximum likelihood estimation*
- Let's include all 12 predictors in the model now
- In R use function glm (for general linear model) and family = "binomial"

Fitting the Model

```
# run logistic regression
# use glm() (general linear model) with family = "binomial" to fit a logistic
# regression.
logit.reg <- glm(Personal.Loan ~ ., data = train.df, family = "binomial")
options(scipen=999)
summary(logit.reg)
```

Output

```
                                Estimate  Std. Error z value             Pr(>|z|)
(Intercept)                -12.6805628   2.2903370  -5.537           0.0000000308 ***
Age                         -0.0369346   0.0848937  -0.435                0.66351
Experience                   0.0490645   0.0844410   0.581                0.56121
Income                       0.0612953   0.0039762  15.416 < 0.0000000000000002 ***
Family                       0.5434657   0.0994936   5.462           0.0000000470 ***
CCAvg                        0.2165942   0.0601900   3.599                0.00032 ***
EducationGraduate            4.2681068   0.3703378  11.525 < 0.0000000000000002 ***
EducationAdvanced/Professional  4.4408154   0.3723360  11.927 < 0.0000000000000002 ***
Mortgage                     0.0015499   0.0007926   1.955                0.05052 .
Securities.Account          -1.1457476   0.3955796  -2.896                0.00377 **
CD.Account                   4.5855656   0.4777696   9.598 < 0.0000000000000002 ***
Online                      -0.8588074   0.2191217  -3.919           0.0000888005 ***
CreditCard                  -1.2514213   0.2944767  -4.250           0.0000214111 ***
---
```

coefficients for logit

Converting to Probability

$$p = \frac{Odds}{1 + Odds}$$

Function `predict` does the conversion from logit to probabilities

```
# use predict() with type = "response" to compute
# predicted probabilities.

logit.reg.pred <- predict(logit.reg, valid.df[, -
     8], type = "response")
# first 5 actual and predicted records

data.frame(actual = valid.df$Personal.Loan[1:5],
     predicted = logit.reg.pred[1:5])
```

```
Output
> data.frame(actual = valid.df$Personal.Loan[1:5],
   + predicted = logit.reg.pred[1:5])

  actual predicted
2   0   0.00002707663
6   0   0.00326343313
9   0   0.03966293189
10  1   0.98846040544
11  0   0.59933974797
```

Interpreting Odds, Probability

For predictive classification, we typically use probability with a cutoff value

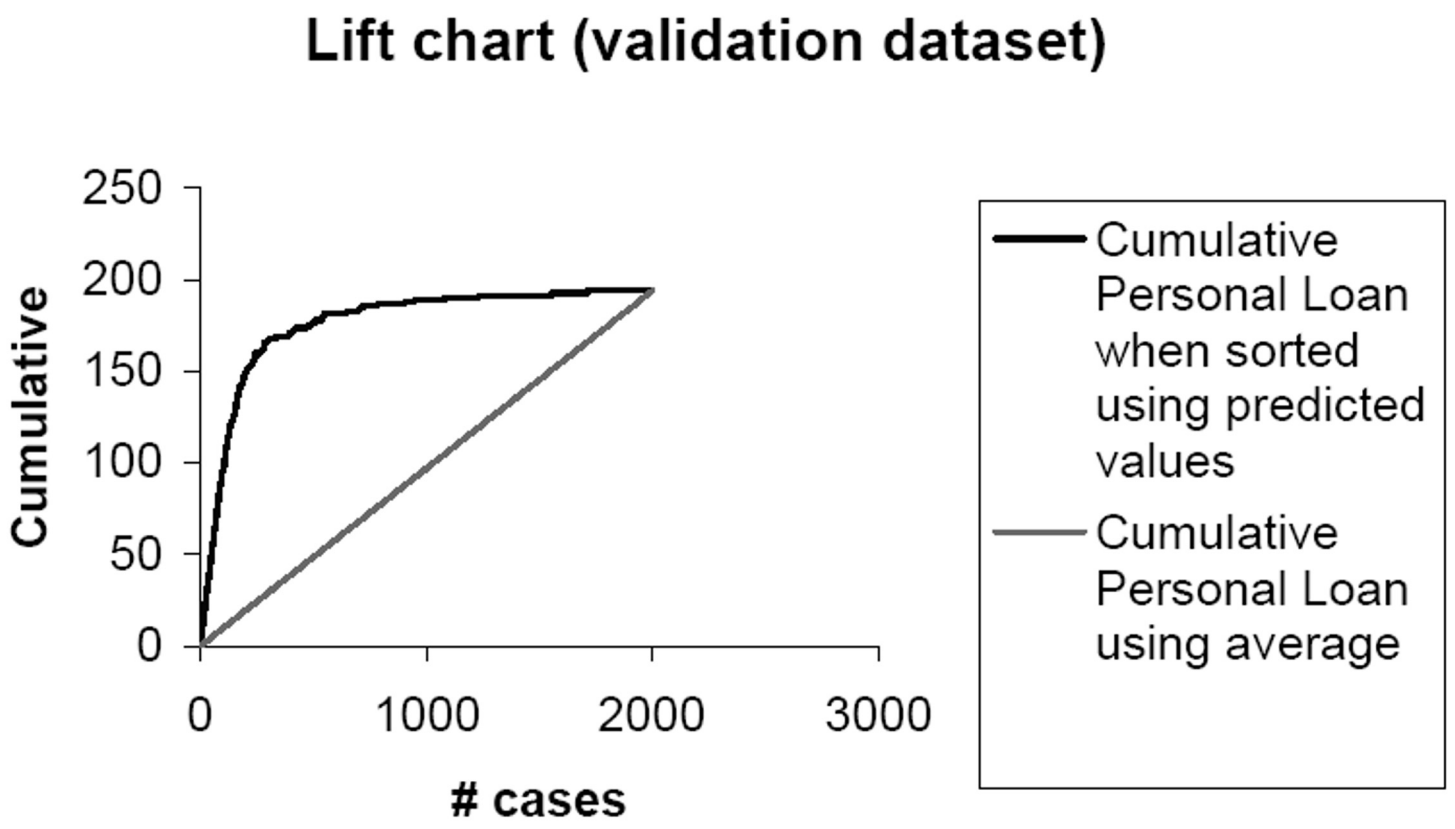
For explanatory purposes, odds have a useful interpretation:

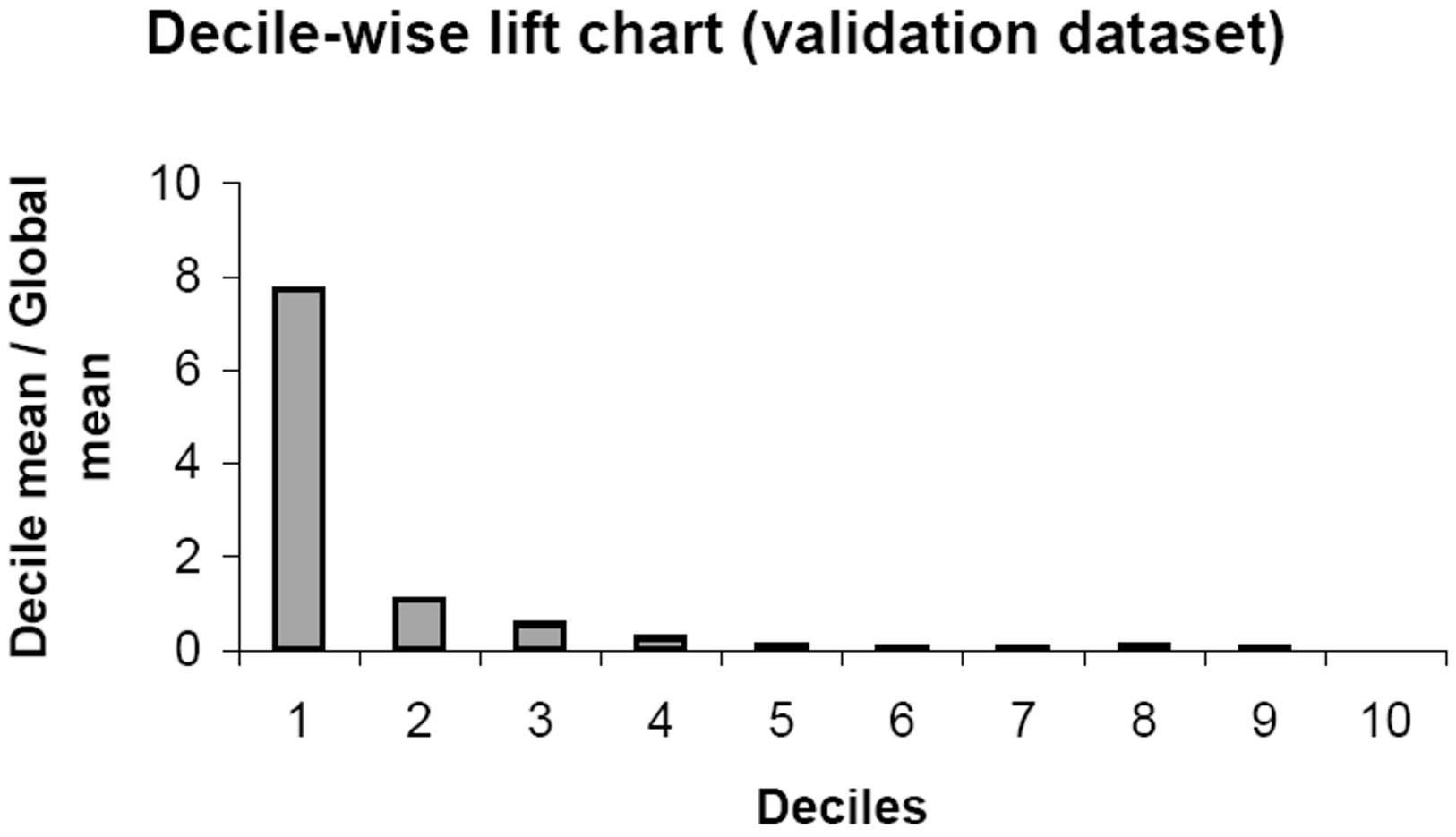
- If we increase x_1 by one unit, holding x_2, x_3 … x_q constant, then
- b_1 is the factor by which the odds of belonging to class 1 increase

Loan Example:
Evaluating Classification Performance

Performance measures: Confusion matrix and % of misclassifications

More useful in this example: **lift**




Decile-wise lift chart (validation dataset)
Decile mean / Global mean
10
8
6
4
2
0
1
2
3
4
5
6
7
8
9
10
Deciles

Multicollinearity

Problem: As in linear regression, if one predictor is a linear combination of other predictor(s), model estimation will fail

- Note that in such a case, we have at least one redundant predictor

Solution: Remove extreme redundancies (by dropping predictors via variable selection, or by data reduction methods such as PCA)

Variable Selection

This is the same issue as in linear regression

The number of correlated predictors can grow when we create derived variables such as **interaction terms** (e.g. *Income x Family)*, to capture more complex relationships

Problem: Overly complex models have the danger of overfitting

Solution: Reduce variables via automated selection of variable subsets (as with linear regression)

See Chapter 6

P-values for Predictors

- Test null hypothesis that coefficient = 0
- Useful for review to determine whether to include variable in model
- Important in profiling tasks, but less important in predictive classification

Summary

- Logistic regression is similar to linear regression, except that it is used with a categorical response
- It can be used for explanatory tasks (=profiling) or predictive tasks (=classification)
- The predictors are related to the response Y via a nonlinear function called the *logit*
- As in linear regression, reducing predictors can be done via variable selection
- Logistic regression can be generalized to more than two classes