Risk Modeling and Classification

EN5422/EV4238 | Fall 2023

w05_classification_1.pdf

(Week 5 – 1/2)

Contents

1 Risk Modeling Intro

1.1 Credit Card Default Data (Dafault)

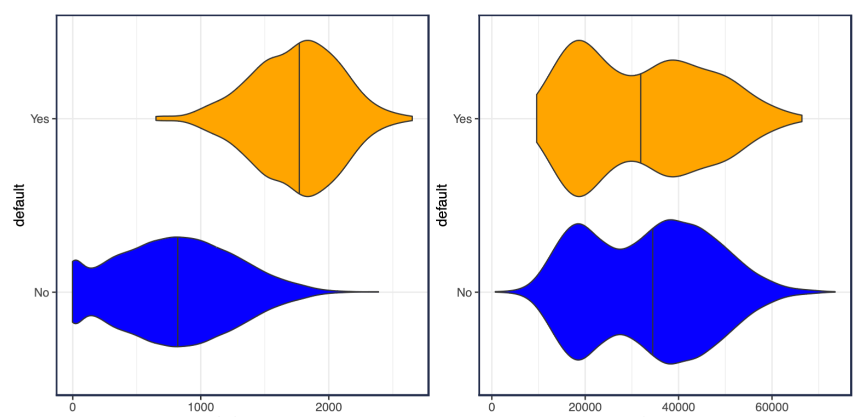
The textbook *An Introduction to Statistical Learning (ISL)* has a description of a simulated credit card default dataset. The interest is on predicting whether an individual will default on their credit card payment.

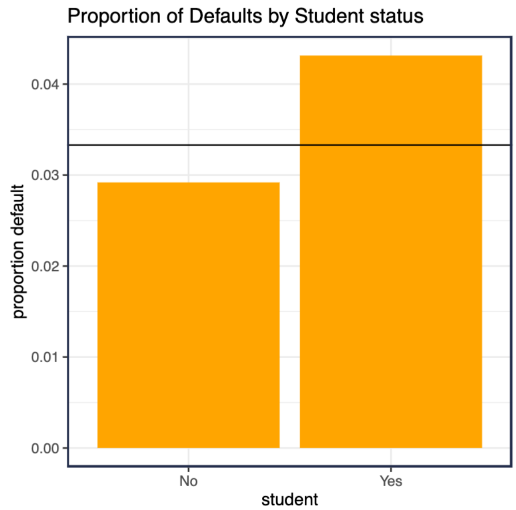
```
# The URL of the xlsx file
url = 'https://github.com/JWarmenhoven/ISLR-python/raw/4100e941914519eea18385daadc9b3dab99ca8e2/Notebooks/Data/Default.xlsx'
# Load the xlsx data
df = pd.read_excel(url, engine='openpyxl')
```

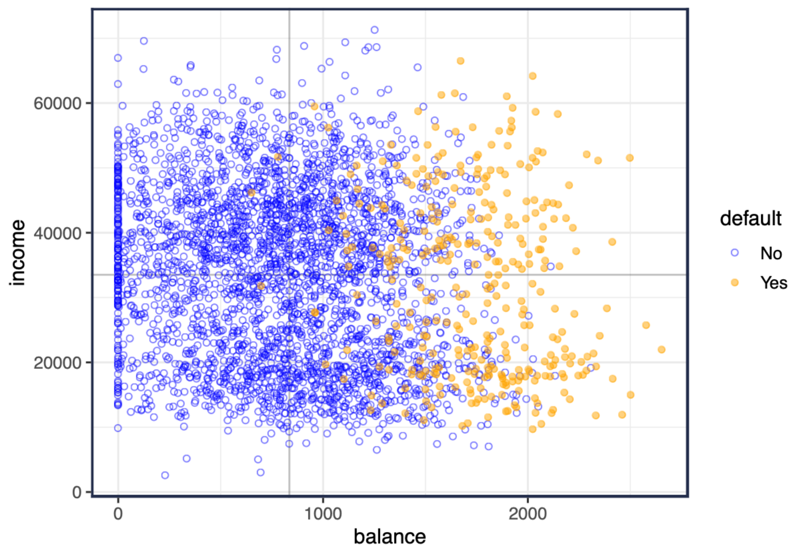
The variables are:

- *Outcome variable* is categorical (factor) Yes and No (default)
- The categorical (factor) variable (student) is either Yes or No
- The average balance a customer has after making their monthly payment (balance)
- The customer's income (income)

default	student	balance	income
No	No	729.526495	44361.625074
No	Yes	817.180407	12106.134700
No	No	1073.549164	31767.138947
No	No	529.250605	35704.493935
No	No	785.655883	38463.495879







Your Turn #1
How would you construct a model to predict the risk of default?

1.2 Set-up

- The outcome variable is categorical and denoted $G \in \mathcal{G}$.
  - Default Credit Card Example: $\mathcal{G} = $ Yes,No
  - Medical Diagnosis Example: $\mathcal{G} = $ stroke,heart attack,drug overdose
- The training data is $D = (X_1, G_1), (X_2, G_2), \ldots, (X_n, G_n)$
- The optimal decision/classification is often based on the posterior probability $\Pr(G = g|\mathbf{X} = \mathrm{x})$

1.3 Binary Risk Modeling

- Classification is simplified when there are only 2 classes.
  - Many multi-class problems can be addressed by solving a set of binary classification problems (e.g., on-vs-rest).
- It is often convenient to transform the outcome variable to a binary {0, 1} variable:

$$Y_i = \begin{cases} 1 G_i \in \mathcal{G}_1 (\text{outcomeofinterest}) \\ 0 \quad G_i \in \mathcal{G}_2 \end{cases}$$

- In the **Default** data, it would be natural to set **default=Yes** to 1 and **default=No** to 0.

1.3.1 Linear Regression

- In this set-up we can run linear regression:

$$\hat{y}(\mathbf{x}) = \hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j x_j$$

```
# Fit the model with sm.OLS lib
X = sm.add_constant(X)
model = sm.OLS(y, X).fit()
# Print the summary of the regression
print(model.summary())
```

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.0812	0.008	-9.685	0.000	-0.098	-0.065
student	-0.0103	0.006	-1.824	0.068	-0.021	0.001
balance	0.0001	3.55e-06	37.412	0.000	0.000	0.000
income	1.992e-07	1.92e-07	1.039	0.299	-1.77e-07	5.75e-07

```
# Fit Linear Regression Model using glmnet
from glmnet import ElasticNet
# Note: alpha=0 makes it Ridge regression (no lasso penalty). To make it
similar to a plain linear regression.
X = Default[['student', 'balance', 'income']].values
y = Default['default'].values
```

```
m = ElasticNet(alpha=0, fit_intercept=True)
m = m.fit(X, y)
# Print the coefficients and intercept
print("Intercept:", m.intercept_)
print("Coefficients:", m.coef_)
```

```
Intercept: -0.062397715192901146
Coefficients: [-4.75394908e-03  1.09145729e-04  1.76617254e-07]
```

Your Turn #2

1. For the binary Y, what is linear regression estimating?
2. What is the *loss function* that linear regression is using?
3. How could you create a *hard classification* from the linear model?
4. Does it make sense to use linear regression for binary risk modeling and classification?

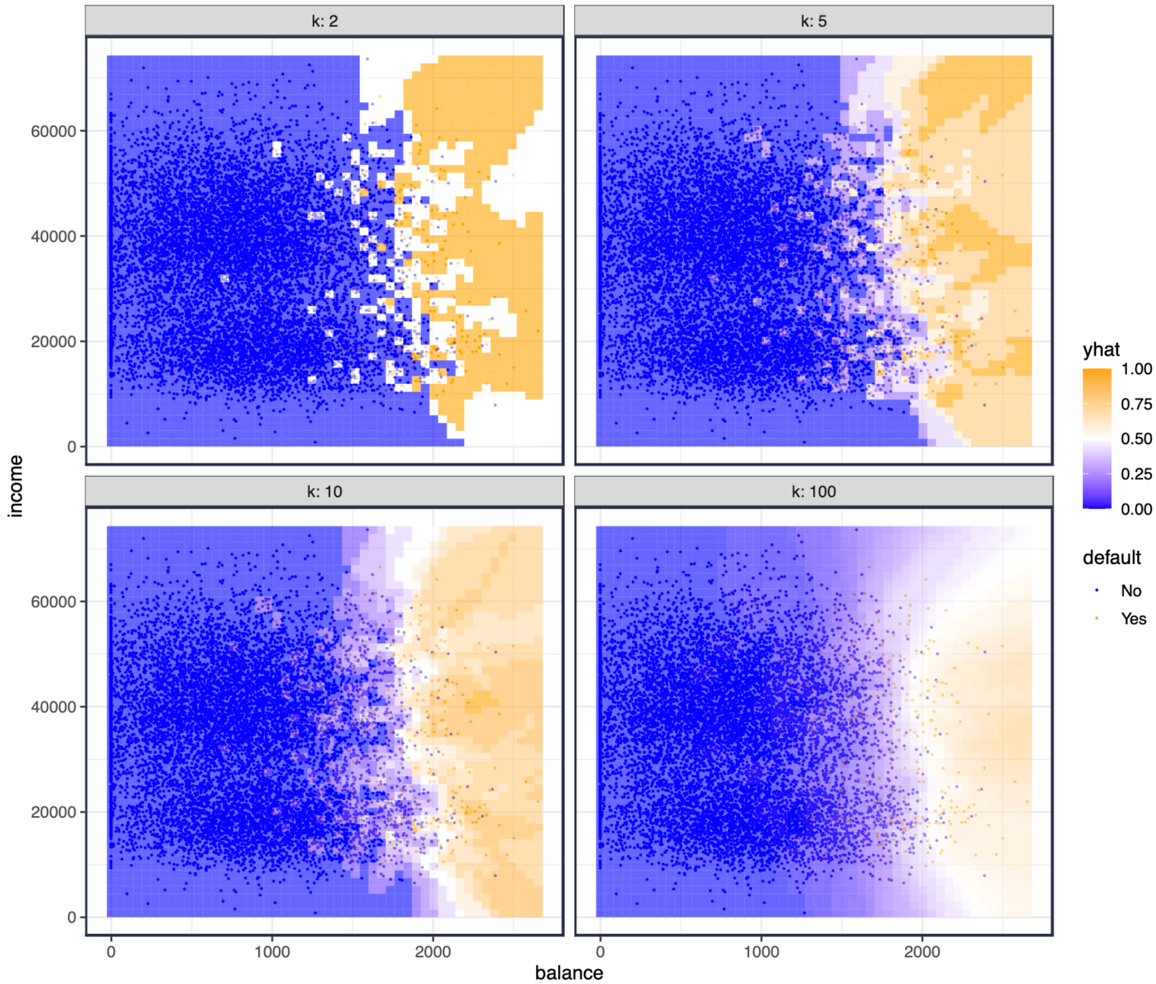
1.3.2 *k*-nearest neighbor (kNN)

- The k-NN method is a non-parametric *local* method, meaning that to make a prediction $\hat{y}|x$, it only uses the training data in the *vicinity* of x.
    - Contrast with OLS linear regression, which uses all $\boldsymbol{X}$'s to get prediction.
- The model (for regression and binary classification) is simple to describe:

$$f_{kNN}(x;k) = \frac{1}{k} \sum_{i:x_i \in N_k(x)} y_i = \text{Avg}(y_i \mid x_i \in N_k(x))$$

    - $N_k(x)$ are the set of k nearest neighbors.
    - Only the k closet y's are used to generate a prediction
- When y is binary (i.e., $y \in 0,1$), the kNN model estimates:

$$f_{kNN}(x;k) \approx p(x) = \Pr(Y = 1|\boldsymbol{X} = x)$$



Your Turn #3

The above plots show a kNN model using the *continuous* predictors of balance and income.

- How could you use kNN with the categorical student predictor?

- The kNN model also has a more general description when the outcome variables is categorical $G \in \mathcal{G}$.

$$f_g^{knn}(x;k) = \frac{1}{k} \sum_{i:x_i \in N_k(x)} 1(g_i = g) = \widehat{Pr}(G_i = g \mid x_i \in N_k(x))$$

  - $N_k(x)$ are the set of k nearest neighbors.
  - Only the k closest y's are used to generate a prediction.
  - It is a *simple proportion* of the k nearest observations that are of class g.

2 Logistic Regression

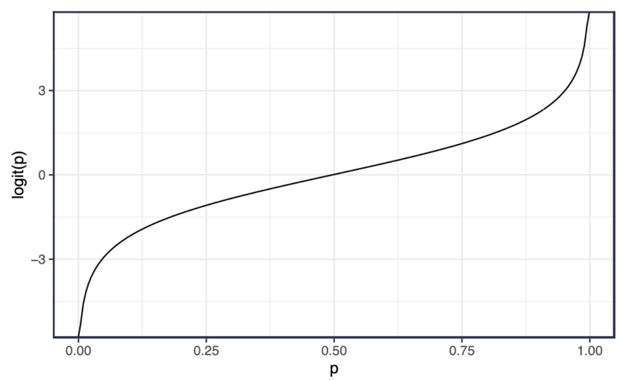
2.1 Basics

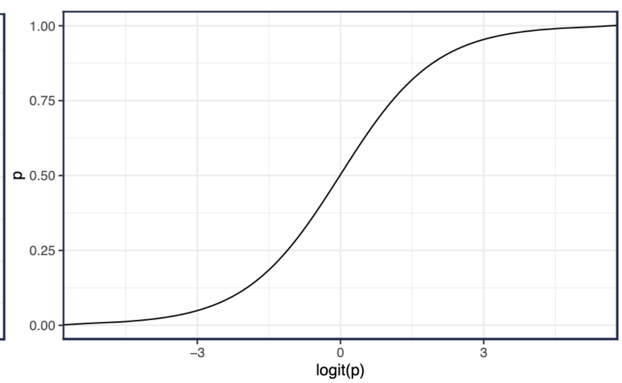
- Let $0 \leq p \leq 1$ be a probability.
- The log-odds of p is called the *logit*.

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right)$$

- The inverse logit is the *logistic function.* Let $f = \text{logit}(p)$, then

$$p = \frac{e^f}{1+e^f} = \frac{1}{1+e^{-f}}$$





- For binary outcome variables $Y \in 0{,}1$, Linear Regression models

$$\text{E}[Y \mid \boldsymbol{X} = x] = \Pr(Y = 1 \mid \boldsymbol{X} = x) = \beta^{\text{T}} x$$

- Alternatively, Logistic Regression models

$$\text{logit}\Pr(Y = 1 \mid X = x) = \log\left(\frac{Pr(Y = 1 \mid X = x)}{1 - Pr(Y = 1 \mid X = x)}\right) = \beta^{\text{T}} x$$

And thus,

$$\Pr(Y = 1 \mid \boldsymbol{X} = x) = \hat{p}(x) = \frac{e^{\beta^{\text{T}} x}}{1 + e^{\beta^{\text{T}} x}} = \left(1 + e^{-\beta^{\text{T}} x}\right)^{-1}$$

2.2 Estimation

- The input data for logistic regression are: $(\mathbf{x}_i, y_i)_{i=1}^n$ where $y_i \in 0{,}1$, $\mathbf{x}_i = \left(x_{i0}, x_{i1}, \ldots, x_{ip}\right)^{\text{T}}$.
- $y_i \mid \mathbf{x}_i \sim \text{Bern}\big(p_i(\beta)\big)$

- $p_i(\beta) = \Pr(Y = 1 \mid \mathbf{X} = \mathbf{x}_i;\ \beta) = \left(1 + e^{-\beta^{\mathrm{T}} x}\right)^{-1}$
- Where $\beta^{\mathrm{T}}\mathbf{x}_i = \mathbf{x}_i^{\mathrm{T}}\beta = \beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j$

- Bernoulli Likelihood Function is:

$$L(\beta) = \prod_{i=1}^{n} p_i(\beta)^{y_i}\left(1 - p_i(\beta)\right)^{1-y_i}$$

$$\begin{aligned}\log L(\beta) &= \sum_{i=1}^{n}\left\{y_i \ln p_i(\beta) + (1 - y_i)\ln\left(1 - p_i(\beta)\right)\right\} \\ &= \sum_{i=1}^{n}\begin{cases}\ln p_i(\beta) & y_i = 1 \\ \ln\left(1 - p_i(\beta)\right) & y_i = 0\end{cases} \\ &= \sum_{i:y_i=1} \ln p_i(\beta) + \sum_{i:y_i=0} \ln\left(1 - p_i(\beta)\right)\end{aligned}$$

- The usual approach to estimating the Logistic Regression coefficients is *maximum likelihood*:

$$\hat{\beta} = \arg\max_{\beta} L(\beta) = \arg\max_{\beta} \log L(\beta)$$

- We can also view this as the coefficients that minimize the *loss function* $\ell(\beta)$, where the loss function is the negative log-likelihood:

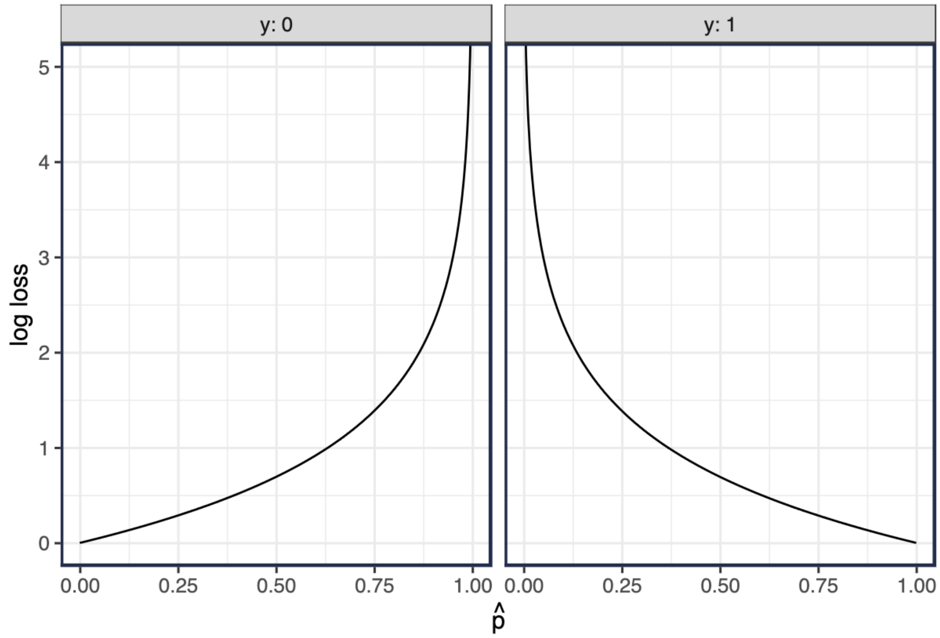
$$\hat{\beta} = \arg\min_{\beta} \ell(\beta)$$

Using loss $\ell(\beta) = -C \log L(\beta)$ where $C > 0$ is some positive constant, e.g., $C = 1/n$

- This view facilitates *penalized logistic regression*:

$$\hat{\beta} = \arg\min_{\beta} \ell(\beta) + \lambda P(\beta)$$

Ridge Penalty	$P(\beta) = \lVert\beta\rVert_2^2 = \sum_{j=1}^{p} \lvert\beta_j\rvert^2 = \beta^{\mathrm{T}}\beta$
Lasso Penalty	$P(\beta) = \lVert\beta\rVert_1 = \sum_{j=1}^{p} \lvert\beta_j\rvert$
Best Subsets	$P(\beta) = \lVert\beta\rVert_0 = \sum_{j=1}^{p} \lvert\beta_j\rvert^0 = \sum_{j=1}^{p} 1_{\beta_j \neq 0}$
Elastic Net	$P(\beta, \alpha) = \frac{(1-\alpha)\lVert\beta\rVert_2^2}{2} + \alpha\lVert\beta\rVert_1 = \sum_{j=1}^{p}\left(\frac{(1-\alpha)\lvert\beta_j\rvert^2}{2} + \alpha\lVert\beta\rVert_1\right)$



2.3 Logistic Regression in Action

- In **Python**, logistic regression can be implemented with smf.logit() function or glmnet LogitNet function.

```
import statsmodels.api as sm
import statsmodels.formula.api as smf

# Fit logistic regression model
model_formula = 'y ~ student + balance + income'
fit_lr = smf.logit(formula=model_formula, data=Default).fit()

# Print the summary
print(fit_lr.summary())
```

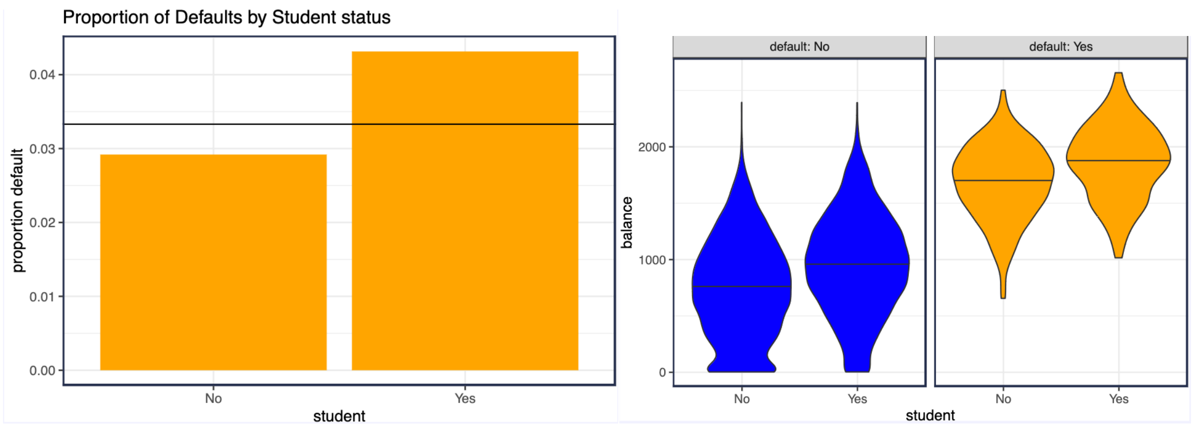
	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-10.8690	0.492	-22.079	0.000	-11.834	-9.904
student	-0.6468	0.236	-2.738	0.006	-1.110	-0.184
balance	0.0057	0.000	24.737	0.000	0.005	0.006
income	3.033e-06	8.2e-06	0.370	0.712	-1.3e-05	1.91e-05

Your Turn #4: Interpreting Logistic Regression

1. What is the estimated probability of default for a Student with a balance of $1000?

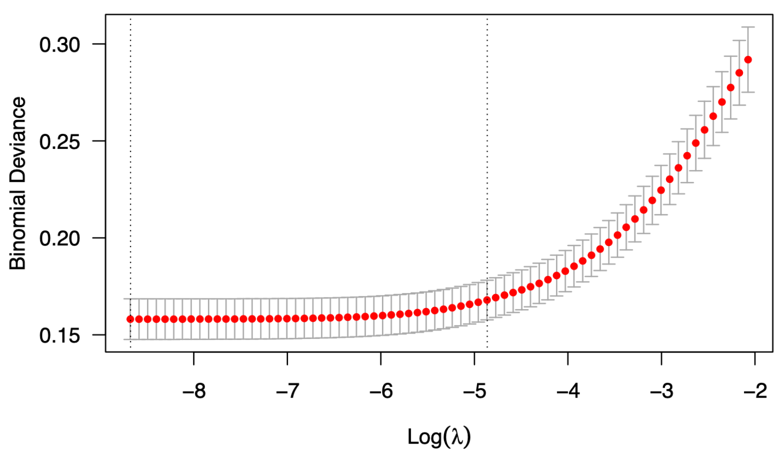
2. What is the estimated probability of default for a Non-Student with a balance of $1000?

3. Why does student=Yes appear to lower risk of default, when the plot of student status vs. default appears to increase risk?



2.3.1 Penalized Logistic Regression

- The smf lib can estimate logistic regression using an elastic net penalty (e.g., ridge, lasso).



term	unpenalized	lambda.min	lambda.1se
(Intercept)	-10.869	-11.056	-7.937
studentYes	-0.647	-0.299	-0.041
balance	0.006	0.006	0.004
income	0.000	0.000	0.000
studentNo	NA	0.325	0.044

Note:

In the context of logistic regression, binomial deviance (also known as the log loss or logistic loss) is a measure of model fit. Specifically, it quantifies how well predicted probabilities align with the true outcomes.

For binary logistic regression, the binomial deviance is given by:

$$D = -\sum_{i=1}^{N}[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)]$$

Where:

- N is the number of observations.
- y_i is the actual outcome for observation i, which takes a value of 0 or 1.
- p_i is the predicted probability for observation i of belonging to class 1.

A smaller value of the binomial deviance indicates a better model fit, while a larger value indicates a poorer fit.

When you introduce λ (lambda), you are moving into the realm of regularization in logistic regression. Regularization is a technique used to prevent overfitting by adding a penalty to the likelihood. The two most common types of regularization for logistic regression are L1 regularization (lasso) and L2 regularization (ridge). The regularization term is controlled by λ, with larger values of λ leading to stronger regularization.

The objective function for logistic regression with L2 regularization (ridge) is:

$$J(\theta) = D + \lambda\sum_{j=1}^{p}\theta_j^2$$

Where:

- D is the binomial deviance.
- p is the number of features (not including the intercept).
- θ_j represents the coefficients of the model.

For L1 regularization (lasso), the regularization term is the absolute value of the coefficients, leading to:

$$J(\theta) = D + \lambda\sum_{j=1}^{p}|\theta_j|$$

In both cases, the λ term penalizes the complexity of the model. As λ increases, the model becomes simpler, potentially leading to increased bias but reduced variance. Selecting an appropriate value of λ is essential for balancing bias and variance and achieving a model that generalizes well to new data.

2.3.2 Logistic Regression Summary

- Logistic Regression (both penalized and unpenalized) estimates a *posterior probability*, $\hat{p}(x) = \widehat{\Pr}(Y = 1 \mid X = x)$.
- This estimate is a function of estimated coefficients.

$$\hat{p}(x) = \frac{e^{\beta^{\mathrm{T}}x}}{1 + e^{\beta^{\mathrm{T}}x}} = \left(1 + e^{-\beta^{\mathrm{T}}x}\right)^{-1}$$

Your Turn #5

1. Given a person's student status, balance, and income, how could you use Logistic Regression to decide if they will default? (i.e., make a hard classification