Supervised Learning: Classification

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Machine Learning for Economics and Finance
Bachelor in Economics

Marcel Weschke

27.05.2025

Learning Goals

At the end of this lecture, you should be able to:

- Revisit the basics of classification problems
- Understand how to measure in-sample and out-of-sample errors for classification problems
- Apply the concepts above in Python

Book Chapter: 4

Recall: Supervised Learning

Suppose you have a quantitative response Y and p different predictors $X = (X_1, X_2, \ldots, X_p)$.

In supervised learning, we try to establish a relation

$$Y = f(X) + \epsilon$$

f: unknown function that represents the systematic information that X provides about Y.

ϵ: random, independent error term with mean zero

Key task: find $\hat{f} \approx f$ that 'fits the data well'

Recall: Supervised Learning

We differentiate between two types of problems:

- **Regression**: Y is quantitative (predict the stock return tomorrow)
- **Classification**: Y is a category (predict whether returns tomorrow are positive or negative)

Today: Introduction to classification problems.

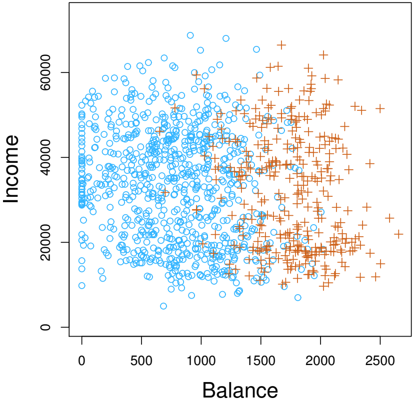
Classification

- In classification problems the outcomes Y are qualitative (and unordered) instead of quantitative
- Often qualitative variables are referred to categorical so we try to predict whether Y belongs to a certain category or class
- Examples:
  - Predict whether a company will default or not
  - Predict whether there will be a recession tomorrow or not
  - Predict whether party A, party B or party C wins the next election
- How to predict categories:
  - Use a machine learning model to assign a probability to each class given the data
  - Use the class with the highest predicted probability as the prediction

Example: *Default* data

- The *Default* dataset contains data on 10,000 customers.
- The aim is to predict which customers will default on their credit card debt.
- Features X: data on income, balance on credit card, data on whether costumer is a student or not
- Outcomes Y: Data on whether costumer has defaulted ('Yes' = 1) or not ('No' = 0)
- **Goal**: Predict probability of default, given data X: $Pr(Y = 1|X)$
- Simple notation: $Pr(Y = 1|X) \doteq p(X)$

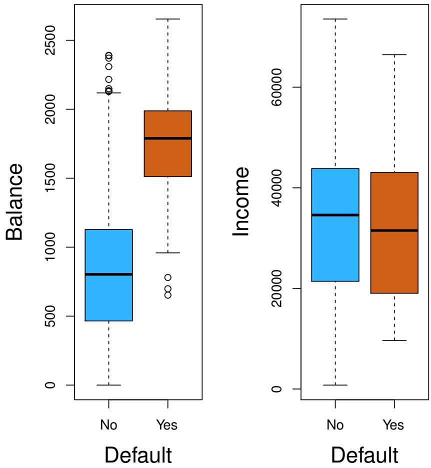
Who defaults?



brown: Default

blue: No default

Box Plots


Balance
0
500
1000
1500
2000
2500
No
Yes
Default
Income
0
20000
40000
60000
No
Yes
Default

Could we use a linear regression for classification?

- Default $= Y \in \{1, 0\}$
- Regress Y on X (linear probability model)
- Linear regression with binary outcomes

$$p(X) = \beta_0 + \beta_1 X$$

- "Likely" to default if $p(X) = \hat{Y} > 0.5$

What are potential issues with this approach?

Logistic Regression

- Logistic regression uses the form

$$p(X) = \frac{\exp(\beta_0 + \beta_1 X)}{1 + \exp(\beta_0 + \beta_1 X)}$$

- We see that $p(X)$ will have values between 0 and 1.

- The log odds ratio is linear in X

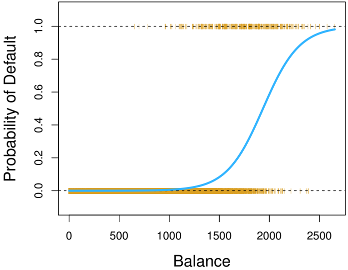
$$\ln\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X$$

Logistic Regression

```python
import statsmodels.api as sm
import numpy as np
from ISLP import load_data

# Load data
default_data = load_data('Default')

# Ensure the target variable is numeric
default_data['default'] = default_data['default'].map({'No': 0, 'Yes': 1})

X = default_data[['balance']]
X = sm.add_constant(X)  # Adds an intercept term to the model
y = default_data['default']

# Fit the logistic regression model
logit_fit = sm.Logit(y, X).fit()
```

Interpretation

- Interpreting what β_1 means is therefore not straightforward:
  - If $\beta_1 = 0$, there is no relationship between Y and X
  - If $\beta_1 > 0$, then larger X increases the likelihood of $Y = 1$
  - If $\beta_1 < 0$, then larger X decreases the likelihood of $Y = 1$
- The sensitivity of Y wrt X depends on the level of X



Estimating the Regression Coefficients

- We use maximum likelihood to estimate the parameters in a logistic regression
- This likelihood gives the probability of the observed zeros and ones in the data as a function of the parameters
- For example choosing a parameter of zero on the balance coefficient would give a lower likelihood than a positive parameter would
- We pick the parameters that maximize the likelihood of the data that we observed
- In Python we simply use the sm.Logit() function which maximizes the likelihood for us

Making Predictions

- Once we have estimated the coefficients β_i, we can use the model to make predictions:
- For a given feature x_i compute the predicted probability:

$$\hat{p}(x_i) = \frac{\exp(\hat{\beta}_0 + \hat{\beta}_1 x_i)}{1 + \exp(\hat{\beta}_0 + \hat{\beta}_1 x_i)}$$

```
logit_probs = logit_fit.predict(sm.add_constant(default_data))
```

- If $\hat{p}(x_i)$ is larger than a certain threshold, for example 50%, assign class 1 (Default = Yes), otherwise, assign class 0 (Default = No)

```
logit_pred = np.repeat("0", len(default_data))
logit_pred[glm_probs > 0.5] = "1"
```

How to Measure Accuracy of a Classification Method?

In the regression setup, we used the mean squared prediction error (MSE) to assess the in-sample and out-of-sample accuracy

We can apply a similar concept in the classification setting where y_i are qualitative.
For this we can use for example the error rate:

$$\text{Error Rate} = \frac{\text{Number of wrong predictions}}{\text{Total number of predictions}}$$

```
error_rate = np.mean(logit_pred != default_data['default'])
```

A good classifier is one with a low **test error rate**

Alternative: Accuracy

As an alternative, we could use the accuracy:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} = 1 - \text{Error Rate}$$

```
accuracy = np.mean(logit_pred == default_data['default'])
```

A good classifier is one with a high **test accuracy**

Useful extensions?

- Several variables?
- More than two outcomes?

Logistic regression with several variables

- We now have p predictors, $X_1, X_2, ..., X_p$ (but Y is still binary)
- Straightforward extension:

$$p(X) = \frac{\exp(\beta_0 + \beta_1 X_1 + ... + \beta_p X_p)}{1 + \exp(\beta_0 + \beta_1 X_1 + ... + \beta_p X_p)}$$

```python
# Ensure the target variables are numeric
default_data['default'] = default_data['default'].map({'No': 0, 'Yes':
↪  1})
default_data['student'] = default_data['student'].map({'No': 0, 'Yes':
↪  1})

X = default_data[['balance', 'income', 'student']]
X = sm.add_constant(X)  # Adds an intercept term to the model
y = default_data['default']

# Fit the logistic regression model
logit_fit = sm.Logit(y, X).fit()
```

Multiclass Logistic Regression

- Logistic regression with two classes generalizes to k classes

$$Pr(Y = k|X) = \frac{\exp(\beta_{0k} + \beta_{1k}X_1 + ... + \beta_{pk}X_p)}{\sum_l^K \exp(\beta_{0l} + \beta_{1l}X_1 + ... + \beta_{pl}X_p)}$$

- Fit model with the suitable Python package (e.g., `statsmodels`, `scikit-learn`, etc.)
- Predictions: choose class with the highest probability among the k classes

Example: Logistic Regressions

Let's have a look at the *Default* dataset (`02_Default_data.ipynb`)

Task 1: Logistic Regressions

Take `02_Default_data.ipynb` as your starting point

1. Randomly split the data into 7000 observations for training and 3000 observations for testing and set the seed to 1 before sampling the data. Call these two datasets *train_data* and *test_data* respectively. (Hint: use the code to split the data from `01_Auto_data_2.ipynb`)
2. Fit a logistic regression of *default* on *income* using the training data. Analyze the significance of the estimated coefficients.
3. Compute the out-of-sample accuracy and error rate and compare to the in-sample statistics. Do you think this is a good model to predict *default*?
4. Add *balance* as a predictor and compute the out-of-sample error rate and accuracy. Do you think this is a good model to predict *default*?
5. Compare the results for Task 1.4 to a model with only *balance* as a predictor. Which model would you choose?
6. Take the model from Task 1.4 but now re-estimate the model using different seeds to draw your training and test data. Does your test error rate change with the seed? What's going on here?

Further Practice Exercises

- Chapter 4, Exercise 13 (a)-(d) and (j)
- Chapter 4, Exercise 14 (a)-(c) and (f)