Chap5-Exercises_LuisCorreia-745724_v2

September 15, 2025

1 MAP5935 - Statistical Learning (Chapter 5 - Resampling Methods)

Prof. Christian Jäkel

https://www.statlearning.com/

```
[1]: import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     from matplotlib.pyplot import subplots
     import statsmodels.api as sm
```

1.1 *Conceptual Exercises*

1.1.1 (1) Using basic statistical properties of the variance, as well as single-variable calculus, derive (5.6). In other words, prove that α given by (5.6) does indeed minimize $Var(\alpha X + (1-\alpha)Y)$.

Solution: We want to show that the choice of α given in (5.6) minimizes

$$\mathrm{Var}\left(\alpha X + (1-\alpha)Y\right).$$

Using the properties of variance and covariance:

$$\begin{aligned}
f(\alpha) &= \mathrm{Var}(\alpha X + (1-\alpha)Y) \\
&= \alpha^2 \, \mathrm{Var}(X) + (1-\alpha)^2 \, \mathrm{Var}(X) + 2\alpha(1-\alpha)\, \mathrm{Cov}(X,Y) \\
&= \alpha^2 \sigma_X^2 + (1-\alpha)^2 \sigma_Y^2 + 2\alpha(1-\alpha)\sigma_{XY},
\end{aligned}$$

where

$$\sigma_X^2 = \mathrm{Var}(X), \quad \sigma_Y^2 = \mathrm{Var}(Y), \quad \sigma_{XY} = \mathrm{Cov}(X,Y).$$

Now minimizing $f(\alpha)$, corresponds to set to 0 the first derivative in relate to α:

$$f'(\alpha) = 2\alpha\sigma_X^2 - 2(1-\alpha)\sigma_Y^2 + 2(1-2\alpha)\sigma_{XY}.$$

Setting $f'(\alpha) = 0$:

$$\alpha(\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}) + (\sigma_{XY} - \sigma_Y^2) = 0.$$

$$\boxed{\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}}}$$

which is exactly equation (5.6).

1.2 *Applied Exercises*

1.2.1 (5) In Chapter 4, we used logistic regression to predict the probability of default using income and balance on the `Default` data set. We will now estimate the test error of this logistic regression model using the validation set approach. Do not forget to set a random seed before beginning your analysis.

1.2.2 (a) Fit a logistic regression model that uses income and balance to predict default.

```
[2]: Default = pd.read_csv("../Data/Default.csv")
```

```
[3]: Default.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  10000 non-null  int64
 1   default     10000 non-null  object
 2   student     10000 non-null  object
 3   balance     10000 non-null  float64
 4   income      10000 non-null  float64
dtypes: float64(2), int64(1), object(2)
memory usage: 390.8+ KB
```

```
[4]: Default.head()
```

```
[4]:    Unnamed: 0 default student      balance        income
     0           0      No      No   729.526495  44361.625074
     1           1      No     Yes   817.180407  12106.134700
     2           2      No      No  1073.549164  31767.138947
     3           3      No      No   529.250605  35704.493935
     4           4      No      No   785.655883  38463.495879
```

We'll fit a logistic regression using **income** and **balance** to predict **default**. The response needs to be binary, so we map `"Yes"→1`, `"No"→0`.

```
[5]: import statsmodels.formula.api as smf
```

```python
# --- setup ---
np.random.seed(42)

# df is your Default DataFrame
df = Default.copy()
df = df.drop(columns=["Unnamed: 0"])
df["default_bin"] = (df["default"].str.lower() == "yes").astype(int)

# --- logistic regression: default ~ income + balance ---
logit_mod = smf.logit("default_bin ~ income + balance", data=df).fit()
print(logit_mod.summary())

# Optional: coefficients as odds ratios with 95% CI
odds = np.exp(logit_mod.params).rename("odds_ratio")
ci = np.exp(logit_mod.conf_int())
ci.columns = ["ci_low", "ci_high"]
or_table = pd.concat([odds, ci], axis=1)
or_table
```

```
Optimization terminated successfully.
         Current function value: 0.078948
         Iterations 10
                          Logit Regression Results
==============================================================================
Dep. Variable:            default_bin   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9997
Method:                           MLE   Df Model:                            2
Date:                Mon, 15 Sep 2025   Pseudo R-squ.:                  0.4594
Time:                        02:12:35   Log-Likelihood:                -789.48
converged:                       True   LL-Null:                       -1460.3
Covariance Type:            nonrobust   LLR p-value:                4.541e-292
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    -11.5405      0.435    -26.544      0.000     -12.393     -10.688
income      2.081e-05   4.99e-06      4.174      0.000     1.1e-05    3.06e-05
balance        0.0056      0.000     24.835      0.000       0.005       0.006
==============================================================================

Possibly complete quasi-separation: A fraction 0.14 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

[5]:
	odds_ratio	ci_low	ci_high
Intercept	0.000010	0.000004	0.000023
income	1.000021	1.000011	1.000031
balance	1.005663	1.005215	1.006111

1.2.3 (b) Using the validation set approach, estimate the test error of this model. In order to do this, you must perform the following steps:

- Split the sample set into a training set and a validation set.
- Fit a multiple logistic regression model using only the training observations.
- Obtain a prediction of default status for each individual in the validation set by computing the posterior probability of default for that individual, and classifying the individual to the `default` category if the posterior probability is greater than 0.5.
- Compute the validation set error, which is the fraction of the observations in the validation set that are misclassified.

```
[6]: from sklearn.metrics import confusion_matrix
     from sklearn.model_selection import train_test_split

     # (i) split into training/validation (50/50), stratified to preserve class␣
      ↪balance
     train_idx, val_idx = train_test_split(
         df.index, test_size=0.50, random_state=1, stratify=df["default_bin"]
     )
     train = df.loc[train_idx].copy()
     valid = df.loc[val_idx].copy()

     # (ii) fit logistic regression on training data
     logit_fit = smf.logit("default_bin ~ income + balance", data=train).fit()
     print(logit_fit.summary())

     # (iii) predict posterior probabilities on validation set and classify (p>0.5␣
      ↪-> default)
     p_valid = logit_fit.predict(valid)                    # P(default=1 | X)
     yhat_valid = (p_valid > 0.5).astype(int)

     # (iv) compute validation-set error (fraction misclassified)
     val_error = (yhat_valid != valid["default_bin"]).mean()
     print(f"Validation set error (misclassification rate): {val_error:.4f}")

     # Confusion matrix for additional insight
     cm = confusion_matrix(valid["default_bin"], yhat_valid, labels=[0,1])
     cm_df = pd.DataFrame(cm, index=["True 0","True 1"], columns=["Pred 0","Pred 1"])
     cm_df
```

```
Optimization terminated successfully.
         Current function value: 0.080810
         Iterations 10
                          Logit Regression Results
==============================================================================
Dep. Variable:            default_bin   No. Observations:                 5000
Model:                          Logit   Df Residuals:                     4997
Method:                           MLE   Df Model:                            2
```

```
Date:              Mon, 15 Sep 2025   Pseudo R-squ.:              0.4479
Time:                    02:12:35     Log-Likelihood:            -404.05
converged:                   True     LL-Null:                   -731.85
Covariance Type:        nonrobust     LLR p-value:             4.369e-143
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    -11.5096      0.613    -18.767      0.000     -12.712     -10.308
income      2.278e-05   7.04e-06      3.235      0.001    8.98e-06    3.66e-05
balance        0.0056      0.000     17.655      0.000       0.005       0.006
==============================================================================

Possibly complete quasi-separation: A fraction 0.14 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
Validation set error (misclassification rate): 0.0254
```

```
[6]:        Pred 0  Pred 1
    True 0    4815      19
    True 1     108      58
```

1.2.4 (c) Repeat the process in (b) three times, using three different splits of the observations into a training set and a validation set. Comment on the results obtained.

```python
[7]: def val_error_for_seed(seed: int):
         """Fit logit on a 50% training split and return metrics on the validation␣
     ↪split."""
         train_idx, val_idx = train_test_split(
             df.index, test_size=0.50, random_state=seed, stratify=df["default_bin"]
         )
         train, valid = df.loc[train_idx], df.loc[val_idx]

         fit = smf.logit("default_bin ~ income + balance", data=train).
     ↪fit(disp=False)
         p = fit.predict(valid)
         yhat = (p > 0.5).astype(int)
         err = (yhat != valid["default_bin"]).mean()

         cm = confusion_matrix(valid["default_bin"], yhat, labels=[0,1])
         tn, fp, fn, tp = cm.ravel()
         return {
             "seed": seed,
             "val_error": err,
             "TN": tn, "FP": fp, "FN": fn, "TP": tp,
             "fit": fit  # keep if you want to inspect coefficients
         }
```

```
seeds = [63, 72, 12]
results = pd.DataFrame([val_error_for_seed(s) for s in␣
 ↪seeds])[["seed","val_error","TN","FP","FN","TP"]]
results
```

[7]:   seed val_error   TN FP   FN TP
     0   63   0.0264 4820 14 118 48
     1   72   0.0252 4817 17 109 57
     2   12   0.0272 4812 22 114 52

[8]: ```
# Simple summary across splits
mean_err = results["val_error"].mean()
std_err = results["val_error"].std(ddof=1)

baseline_error = 1 - (df["default_bin"]==0).mean() # error if always predict␣
 ↪"No default"

print(f"Baseline (always predict 'No'): {baseline_error:.4f}")
print(f"Validation error mean over seeds: {mean_err:.4f} (sd = {std_err:.4f})")
```

```
Baseline (always predict 'No'): 0.0333
Validation error mean over seeds: 0.0263  (sd = 0.0010)
```

Comments

- The baseline error (predicting "No" for everyone) is approximately the prevalence of defaults
  (about 3–4% in Default). Our logistic model beat this baseline (typically around 2–3% mis-
  classification with a 0.5 threshold).

- Across different splits, the error will vary slightly (usually a few tenths of a percent), which
  reflects sampling variability from using a single validation split.

- The confusion matrix shows more false negatives (FN) than false positives (FP) because the
  positive class (default) is rare; with a 0.5 threshold, the classifier is *conservative*.

- Performance is stable across splits, but a resampling scheme with more averaging (e.g., K-fold
  CV or repeated CV) provides a more reliable estimate than a single split.

1.2.5 (d) Now consider a logistic regression model that predicts the probability of default using `income`, `balance`, and a dummy variable for `student`. Estimate the test error for this model using the validation set approach. Comment on whether or not including a dummy variable for `student` leads to a reduction in the test error rate.

[9]: ```
--- single 50/50 split, stratified by the rare class for comparability ---
train_idx, val_idx = train_test_split(
    df.index, test_size=0.50, random_state=1, stratify=df["default_bin"]
)
train, valid = df.loc[train_idx], df.loc[val_idx]
```

```python
# --- fit base vs. extended models on the *same* training set ---
fit_base = smf.logit("default_bin ~ income + balance", data=train).
 ↪fit(disp=False)
fit_ext  = smf.logit("default_bin ~ income + balance + C(student)", data=train).
 ↪fit(disp=False)

# --- predict probabilities on validation set and classify with 0.5 threshold␣
 ↪---
p_base = fit_base.predict(valid)
p_ext  = fit_ext.predict(valid)
yhat_base = (p_base > 0.5).astype(int)
yhat_ext  = (p_ext  > 0.5).astype(int)

# --- misclassification rates (validation-set error) ---
err_base = (yhat_base != valid["default_bin"]).mean()
err_ext  = (yhat_ext  != valid["default_bin"]).mean()
print(f"Validation error (base: income + balance): {err_base:.4f}")
print(f"Validation error (+ student dummy)        : {err_ext:.4f}")
print(f"Δ error (extended − base)                 : {err_ext - err_base:+.4f}")

# Optional: confusion matrices to see the trade-offs
cm_base = confusion_matrix(valid["default_bin"], yhat_base, labels=[0,1])
cm_ext  = confusion_matrix(valid["default_bin"], yhat_ext,  labels=[0,1])
pd.DataFrame(cm_base, index=["True 0","True 1"], columns=["Pred 0","Pred 1"])
pd.DataFrame(cm_ext,  index=["True 0","True 1"], columns=["Pred 0","Pred 1"])
```

```
Validation error (base: income + balance): 0.0254
Validation error (+ student dummy)        : 0.0260
Δ error (extended − base)                 : +0.0006
```

```
[9]:         Pred 0  Pred 1
    True 0     4812      22
    True 1      108      58
```

```python
[10]: def compare_err(seed: int):
          tr, va = train_test_split(df.index, test_size=0.50, random_state=seed,
                                    stratify=df["default_bin"])
          tr, va = df.loc[tr], df.loc[va]
          m0 = smf.logit("default_bin ~ income + balance", data=tr).fit(disp=False)
          m1 = smf.logit("default_bin ~ income + balance + C(student)", data=tr).
       ↪fit(disp=False)
          e0 = ((m0.predict(va) > 0.5).astype(int) != va["default_bin"]).mean()
          e1 = ((m1.predict(va) > 0.5).astype(int) != va["default_bin"]).mean()
          return {"seed": seed, "err_base": e0, "err_plus_student": e1, "delta": e1 -␣
       ↪e0}
```

```
res = pd.DataFrame([compare_err(s) for s in seeds])
res, res[["err_base","err_plus_student","delta"]].mean().to_frame("mean").T
```

[10]: (seed err_base err_plus_student   delta
       0   63   0.0264   0.0268 0.0004
       1   72   0.0252   0.0262 0.0010
       2   12   0.0272   0.0272 0.0000,
             err_base err_plus_student   delta
       mean 0.026267   0.026733 0.000467)

Comments

- **No gain from `student`:** Across splits, the mean Δ error is $+0.00047$ (extended $-$ base), i.e., adding the `student` dummy slightly **worsens** validation error. The model with only `income + balance` is preferable.

- **Imbalanced data behavior:** From the confusion matrix (TN=4812, FP=22, FN=108, TP=58), specificity is very high (**99.5%**) while sensitivity is modest (~**35%**) under the 0.5 threshold—typical with rare positives.