Fine-Tuning Your Model

Contents

Having trained models, now you will learn how to evaluate them. In this chapter, you will be introduced to several metrics along with a visualization technique for analyzing classification model performance using scikit-learn. You will also learn how to optimize classification and regression models through the use of hyperparameter tuning.

How good is your model?

Deciding on a primary metric

As you have seen, several metrics can be useful to evaluate the performance of classification models, including accuracy, precision, recall, and F1-score.

In this exercise, you will be provided with three different classification problems, and your task is to select the problem where **precision** is best suited as the primary metric.

- ☐ A model predicting the presence of cancer as the positive class.
- ☐ A classifier predicting the positive class of a computer program containing malware.
- ☑ A model predicting if a customer is a high-value lead for a sales team with limited capacity.

Correct! With limited capacity, the sales team needs the model to return the highest proportion of true positives compared to all predicted positives, thus minimizing wasted effort.

Assessing a diabetes prediction classifier

In this chapter you'll work with the `diabetes_df` dataset introduced previously.

The goal is to predict whether or not each individual is likely to have diabetes based on the features body mass index (BMI) and age (in years). Therefore, it is a binary classification problem. A target value of `0` indicates that the individual does *not* have diabetes, while a value of `1` indicates that the individual *does* have diabetes.

`diabetes_df` has been preloaded for you as a pandas DataFrame and split into `X_train`, `X_test`, `y_train`, and `y_test`. In addition, a `KNeighborsClassifier()` has been instantiated and assigned to `knn`.

You will fit the model, make predictions on the test set, then produce a confusion matrix and classification report.

- Import `confusion_matrix` and `classification_report`.
- Fit the model to the training data.
- Predict the labels of the test set, storing the results as `y_pred`.
- Compute and print the confusion matrix and classification report for the test labels versus the predicted labels.

```
# edited/added
df = pd.read_csv('archive/Supervised-Learning-with-scikit-
learn/datasets/diabetes_clean.csv')
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,
random_state=42)

# Import confusion matrix
from sklearn.metrics import confusion_matrix, classification_report

knn = KNeighborsClassifier(n_neighbors=6)

# Fit the model to the training data
knn.fit(X_train, y_train)

# Predict the labels of the test data: y_pred
```

```
## KNeighborsClassifier(n_neighbors=6)
```

```
y_pred = knn.predict(X_test)

# Generate the confusion matrix and classification report
print(confusion_matrix(y_test, y_pred))
```

```
## [[176  30]
##  [ 56  46]]
```

```
print(classification_report(y_test, y_pred))
```

```
##               precision    recall  f1-score   support
##
##            0       0.76      0.85      0.80       206
##            1       0.61      0.45      0.52       102
##
##     accuracy                           0.72       308
##    macro avg       0.68      0.65      0.66       308
## weighted avg       0.71      0.72      0.71       308
```

Excellent! The model produced 116 true positives, 34 true negatives, 35 false negatives, and 46 false positives. The classification report shows a better F1-score for the zero class, which represents individuals who do not have diabetes.

Logistic regression and the ROC curve

Building a logistic regression model

In this exercise, you will build a logistic regression model using all features in the `diabetes_df` dataset. The model will be used to predict the probability of individuals in the test set having a diabetes diagnosis.

The `diabetes_df` dataset has been split into `X_train`, `X_test`, `y_train`, and `y_test`, and preloaded for you.

- Import `LogisticRegression`.
- Instantiate a logistic regression model, `logreg`.
- Fit the model to the training data.
- Predict the probabilities of each individual in the test set having a diabetes diagnosis, storing the array of positive probabilities as `y_pred_probs`.

```
# Import LogisticRegression
from sklearn.linear_model import LogisticRegression

# Instantiate the model
logreg = LogisticRegression()

# Fit the model
logreg.fit(X_train, y_train)

# Predict probabilities
```

```
## LogisticRegression()
##
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
```

```
y_pred_probs = logreg.predict_proba(X_test)[:, 1]

print(y_pred_probs[:10])
```

```
## [0.29470305 0.19804348 0.14437298 0.16915287 0.51158696 0.47996502
##  0.01555776 0.60402348 0.53766072 0.78702224]
```

Nicely done! Notice how the probability of a diabetes diagnosis for the first 10 individuals in the test set ranges from 0.01 to 0.79. Now let's plot the ROC curve to visualize performance using different thresholds.

The ROC curve

Now you have built a logistic regression model for predicting diabetes status, you can plot the ROC curve to visualize how the true positive rate and false positive rate vary as the decision threshold changes.

The test labels, `y_test`, and the predicted probabilities of the test features belonging to the positive class, `y_pred_probs`, have been preloaded for you, along with `matplotlib.pyplot` as `plt`.

You will create a ROC curve and then interpret the results.

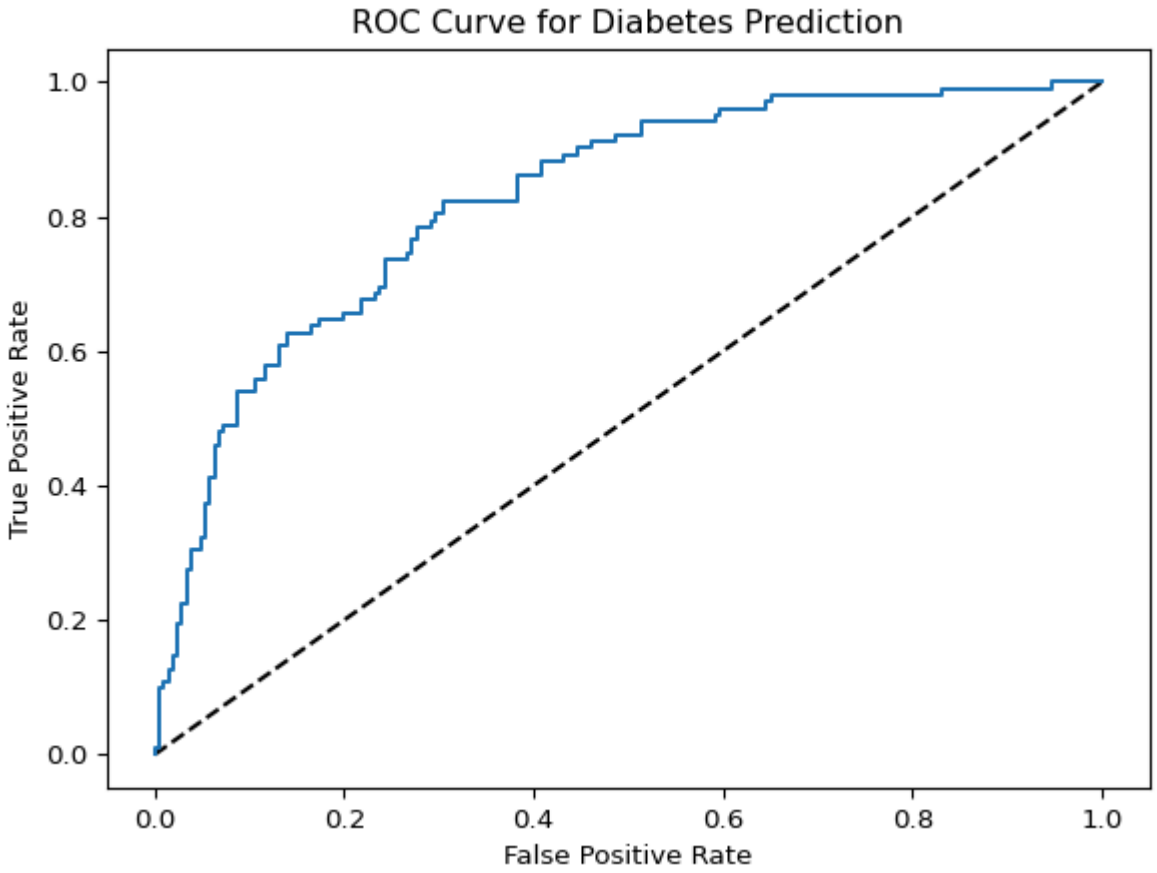
- Import `roc_curve`.
- Calculate the ROC curve values, using `y_test` and `y_pred_probs`, and unpacking the results into `fpr`, `tpr`, and `thresholds`.
- Plot true positive rate against false positive rate.

```
# Import roc_curve
from sklearn.metrics import roc_curve

# Generate ROC curve values: fpr, tpr, thresholds
fpr, tpr, thresholds = roc_curve(y_test, y_pred_probs)

plt.plot([0, 1], [0, 1], 'k--')

# Plot tpr against fpr
plt.plot(fpr, tpr)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve for Diabetes Prediction')
plt.show()
```



Well done on producing the ROC curve for the diabetes prediction model.

But, what does the plot tell you about the model's performance?

- ☐ The model is about as good as randomly guessing the class of each observation.
- ☐ The model is much worse than randomly guessing the class of each observation.
- ☑ The model is much better than randomly guessing the class of each observation.
- ☐ It is not possible to conclude whether the model performs better or worse than randomly guessing the class of each observation.

Well done! The ROC curve is above the dotted line, so the model performs better than randomly guessing the class of each observation.

ROC AUC

The ROC curve you plotted in the last exercise looked promising.

Now you will compute the area under the ROC curve, along with the other classification metrics you have used previously.

The `confusion_matrix` and `classification_report` functions have been preloaded for you, along with the `logreg` model you previously built, plus `X_train`, `X_test`, `y_train`, `y_test`. Also, the model's predicted test set labels are stored as `y_pred`, and probabilities of test set observations belonging to the positive class stored as `y_pred_probs`.

A `knn` model has also been created and the performance metrics printed in the console, so you can compare the `roc_auc_score`, `confusion_matrix`, and `classification_report` between the two models.

- Import `roc_auc_score`.
- Calculate and print the ROC AUC score, passing the test labels and the predicted positive class probabilities.
- Calculate and print the confusion matrix.
- Call `classification_report()`.

```
# Import roc_auc_score
from sklearn.metrics import roc_auc_score

# Calculate roc_auc_score
print(roc_auc_score(y_test, y_pred_probs))

# Calculate the confusion matrix
```

```
## 0.8260517799352751
```

```
print(confusion_matrix(y_test, y_pred))

# Calculate the classification report
```

```
## [[176  30]
##  [ 56  46]]
```

```
print(classification_report(y_test, y_pred))
```

```
##               precision    recall  f1-score   support
##
##            0       0.76      0.85      0.80       206
##            1       0.61      0.45      0.52       102
##
##     accuracy                           0.72       308
##    macro avg       0.68      0.65      0.66       308
## weighted avg       0.71      0.72      0.71       308
```

Did you notice that logistic regression performs better than the KNN model across all the metrics you calculated? A ROC AUC score of 0.8002 means this model is 60% better than a chance model at correctly predicting labels! scikit-learn makes it easy to produce several classification metrics with only a few lines of code.

Hyperparameter tuning

Hyperparameter tuning with GridSearchCV

Now you have seen how to perform grid search hyperparameter tuning, you are going to build a lasso regression model with optimal hyperparameters to predict blood glucose levels using the features in the `diabetes_df` dataset.

`X_train`, `X_test`, `y_train`, and `y_test` have been preloaded for you. A `KFold()` object has been created and stored for you as `kf`, along with a lasso regression model as `lasso`.

- Import `GridSearchCV`.
- Set up a parameter grid for `"alpha"`, using `np.linspace()` to create 20 evenly spaced values ranging from `0.00001` to `1`.
- Call `GridSearchCV()`, passing `lasso`, the parameter grid, and setting `cv` equal to `kf`.
- Fit the grid search object to the training data to perform a cross-validated grid search.

```
# Import GridSearchCV
from sklearn.model_selection import GridSearchCV

# Set up the parameter grid
param_grid = {"alpha": np.linspace(0.00001, 1, 20)}

# Instantiate lasso_cv
lasso_cv = GridSearchCV(lasso, param_grid, cv=kf)

# Fit to the training data
lasso_cv.fit(X_train, y_train)
```

```
## GridSearchCV(cv=KFold(n_splits=6, random_state=5, shuffle=True),
##              estimator=Lasso(alpha=0.3),
##              param_grid={'alpha': array([1.00000000e-05, 5.26410526e-02,
1.05272105e-01, 1.57903158e-01,
##        2.10534211e-01, 2.63165263e-01, 3.15796316e-01, 3.68427368e-01,
##        4.21058421e-01, 4.73689474e-01, 5.26320526e-01, 5.78951579e-01,
##        6.31582632e-01, 6.84213684e-01, 7.36844737e-01, 7.89475789e-01,
##        8.42106842e-01, 8.94737895e-01, 9.47368947e-01, 1.00000000e+00])})
```

```
print("Tuned lasso paramaters: {}".format(lasso_cv.best_params_))
```

```
## Tuned lasso paramaters: {'alpha': 1e-05}
```

```
print("Tuned lasso score: {}".format(lasso_cv.best_score_))
```

```
## Tuned lasso score: 0.2715979243400802
```

Well done! Unfortunately, the best model only has an R-squared score of `0.33`, highlighting that using the optimal hyperparameters does not guarantee a high performing model!

Hyperparameter tuning with RandomizedSearchCV

As you saw, `GridSearchCV` can be computationally expensive, especially if you are searching over a large hyperparameter space. In this case, you can use `RandomizedSearchCV`, which tests a fixed number of hyperparameter settings from specified probability distributions.

Training and test sets from `diabetes_df` have been pre-loaded for you as `X_train`. `X_test`, `y_train`, and `y_test`, where the target is `"diabetes"`. A logistic regression model has been created and stored as `logreg`, as well as a `KFold` variable stored as `kf`.

You will define a range of hyperparameters and use `RandomizedSearchCV`, which has been imported from `sklearn.model_selection`, to look for optimal hyperparameters from these options.

- Create `params`, adding `"l1"` and `"l2"` as `penalty` values, setting `C` to a range of `50` float values between `0.1` and `1.0`, and `class_weight` to either `"balanced"` or a dictionary containing `0:0.8, 1:0.2`.
- Create the Randomized Search CV object, passing the model and the parameters, and setting `cv` equal to `kf`.
- Fit `logreg_cv` to the training data.
- Print the model's best parameters and accuracy score.

```
# edited/added
from sklearn.model_selection import RandomizedSearchCV

# Create the parameter space
params = {"penalty": ["l1", "l2"],
         "tol": np.linspace(0.0001, 1.0, 50),
         "C": np.linspace(0.1, 1.0, 50),
         "class_weight": ["balanced", {0:0.8, 1:0.2}]}

# Instantiate the RandomizedSearchCV object
logreg_cv = RandomizedSearchCV(logreg, params, cv=kf)

# Fit the data to the model
logreg_cv.fit(X_train, y_train)

# Print the tuned parameters and score
```

```
## RandomizedSearchCV(cv=KFold(n_splits=6, random_state=5, shuffle=True),
##                    estimator=LogisticRegression(),
##                    param_distributions={'C': array([0.1       , 0.11836735,
0.13673469, 0.15510204, 0.17346939,
##        0.19183673, 0.21020408, 0.22857143, 0.24693878, 0.26530612,
##        0.28367347, 0.30204082, 0.32040816, 0.33877551, 0.35714286,
##        0.3755102 , 0.39387755, 0.4122449 , 0.43061224, 0.44897959,
##        0.467346...
##        4.89846939e-01, 5.10253061e-01, 5.30659184e-01, 5.51065306e-01,
##        5.71471429e-01, 5.91877551e-01, 6.12283673e-01, 6.32689796e-01,
##        6.53095918e-01, 6.73502041e-01, 6.93908163e-01, 7.14314286e-01,
##        7.34720408e-01, 7.55126531e-01, 7.75532653e-01, 7.95938776e-01,
##        8.16344898e-01, 8.36751020e-01, 8.57157143e-01, 8.77563265e-01,
##        8.97969388e-01, 9.18375510e-01, 9.38781633e-01, 9.59187755e-01,
##        9.79593878e-01, 1.00000000e+00])})
##
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/model_selection/_validation.py:548: FitFailedWarning: Estimator fit
failed. The score on this train-test partition for these parameters will be set to
nan. Details:
## Traceback (most recent call last):
##   File "/Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/model_selection/_validation.py", line 531, in _fit_and_score
##     estimator.fit(X_train, y_train, **fit_params)
##   File "/Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
```

```
packages/sklearn/linear_model/_logistic.py", line 1304, in fit
##     solver = _check_solver(self.solver, self.penalty, self.dual)
##   File "/Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py", line 442, in _check_solver
##     raise ValueError("Solver %s supports only 'l2' or 'none' penalties, "
## ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.
##
##   warnings.warn("Estimator fit failed. The score on this train-test"
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
```

```
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/model_selection/_validation.py:548: FitFailedWarning: Estimator fit
failed. The score on this train-test partition for these parameters will be set to
nan. Details:
## Traceback (most recent call last):
##   File "/Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/model_selection/_validation.py", line 531, in _fit_and_score
##     estimator.fit(X_train, y_train, **fit_params)
##   File "/Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py", line 1304, in fit
##     solver = _check_solver(self.solver, self.penalty, self.dual)
##   File "/Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py", line 442, in _check_solver
##     raise ValueError("Solver %s supports only 'l2' or 'none' penalties, "
## ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.
##
##   warnings.warn("Estimator fit failed. The score on this train-test"
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
```

```
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
## /Users/macos/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to converge (status=1):
## STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
##
## Increase the number of iterations (max_iter) or scale the data as shown in:
##     https://scikit-learn.org/stable/modules/preprocessing.html
## Please also refer to the documentation for alternative solver options:
##     https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
##   n_iter_i = _check_optimize_result(
```

```
print("Tuned Logistic Regression Parameters: {}".format(logreg_cv.best_params_))
```

```
## Tuned Logistic Regression Parameters: {'tol': 0.7347204081632653, 'penalty': 'l2', 'class_weight': 'balanced', 'C': 0.13673469387755102}
```

```
print("Tuned Logistic Regression Best Accuracy Score:
{}".format(logreg_cv.best_score_))
```

```
## Tuned Logistic Regression Best Accuracy Score: 0.737012987012987
```

Great searching! Even without exhaustively trying every combination of hyperparameters, the model has an accuracy of over 70% on the test set! So far we have worked with clean datasets; however, in the next chapter, we will discuss the steps required to transform messy data before building supervised learning models.
