17.7.3

Make Predictions and Evaluate Results

Now that you have preprocessed your dataset, you can now turn your attention to using the decision tree model to make predictions and evaluate the results.

Fit the Decision Tree Model

After scaling the features data, the decision tree model can be created and trained. First, we create the decision tree classifier instance and then we train or fit the "model" with the scaled training data.

Add and run the following code block to create the decision tree instance and fit the model:

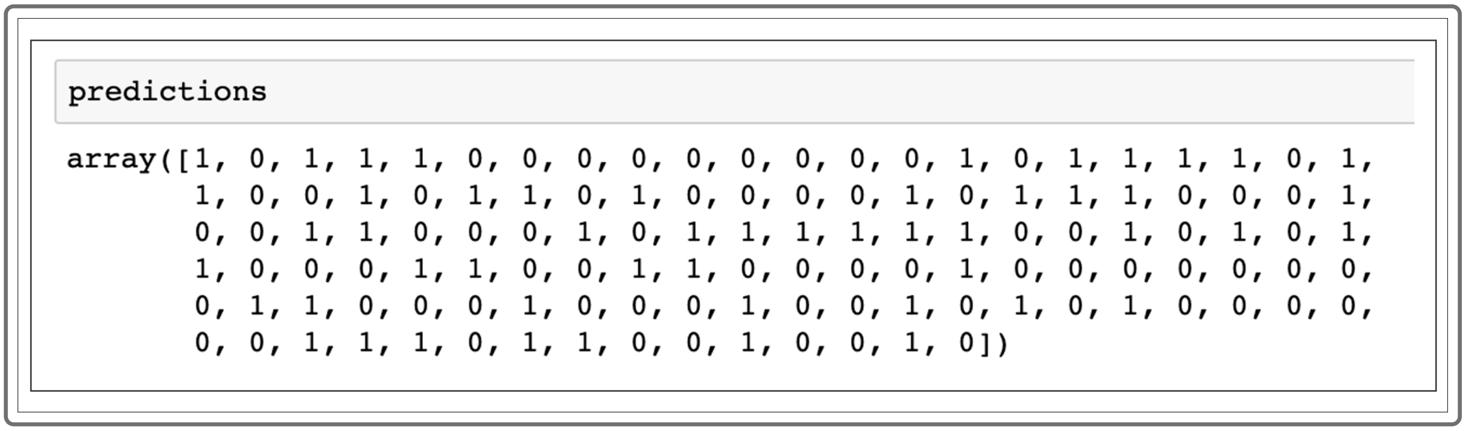
```
# Creating the decision tree classifier instance.
model = tree.DecisionTreeClassifier()
# Fitting the model.
model = model.fit(X_train_scaled, y_train)
```

Make Predictions Using the Testing Data

After fitting the model, we can run the following code to make predictions using the scaled testing data:

```
# Making predictions using the testing data.
predictions = model.predict(X_test_scaled)
```

The output from this code will be an array of 125 predictions with either a 1 for a bad loan application or a 0 for a good, or approved, loan application.



```
predictions
```

```
array([1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1,
       1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1,
       0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0])
```

NOTE

Your predictions array may look different if you don't use the same seeding in the `random_state` parameter.

Evaluate the Model

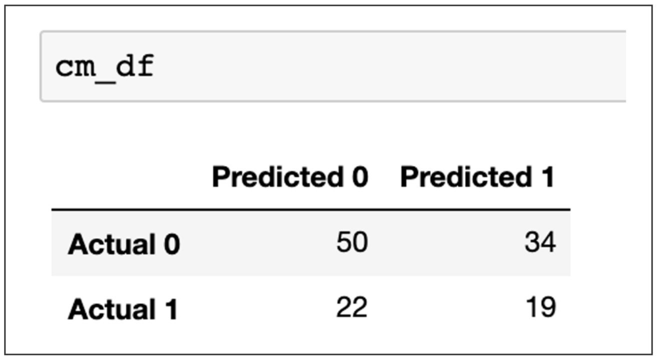
Finally, we'll determine how well our model classifies loan applications. First, we need to use a confusion matrix.

The following code block creates the `confusion_matrix` using the `y_test` and the `predictions` that we just calculated and adds the `confusion_matrix` array to a DataFrame:

```
# Calculating the confusion matrix
cm = confusion_matrix(y_test, predictions)

# Create a DataFrame from the confusion matrix.
cm_df = pd.DataFrame(
    cm, index=["Actual 0", "Actual 1"], columns=["Predicted 0", "Predicted 1"])

cm_df
```



```
cm_df
```

	Predicted 0	Predicted 1
Actual 0	50	34
Actual 1	22	19

The results show that:

- Out of 84 good loan applications (Actual 0), 50 were predicted to be good (Predicted 0), which we call true positives.
- Out of 84 good loan applications (Actual 0), 34 were predicted to be bad (Predicted 1), which are considered false negatives.
- Out of 41 bad loan applications (Actual 1), 22 were predicted to be good (Predicted 0) and are considered false positives.
- Out of 41 bad loan applications (Actual 1), 19 were predicted to be bad (Predicted 1) and are considered true negatives.

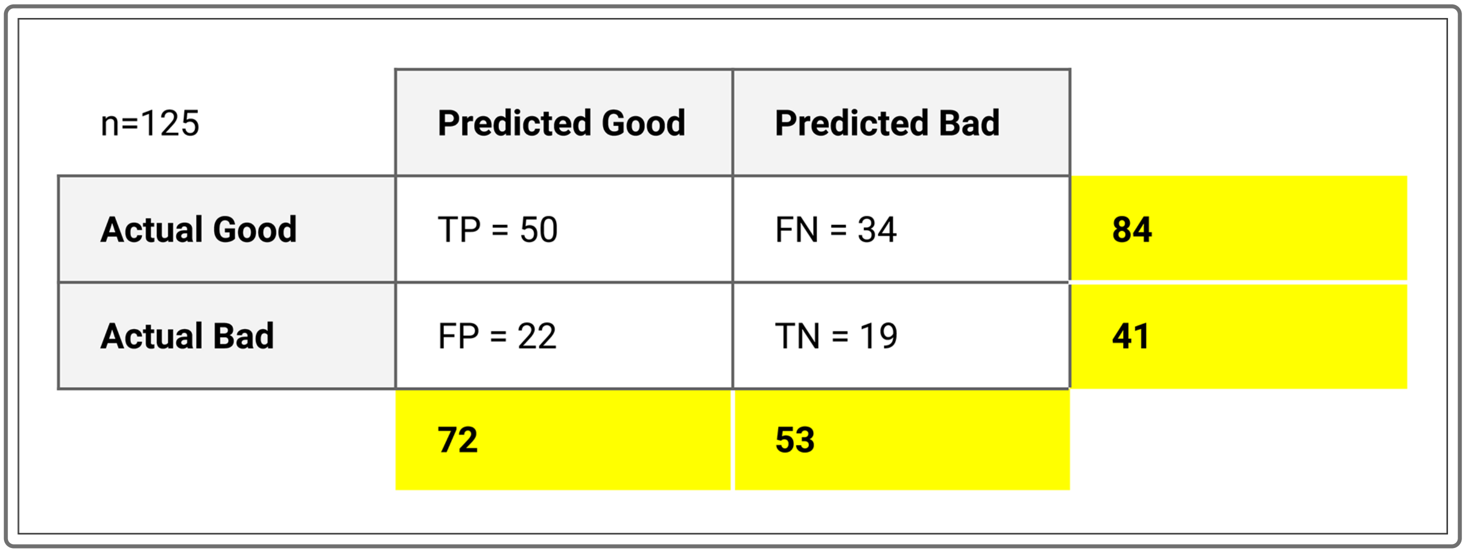
What is the recall (sensitivity) for bad loans (Actual 1)?

- ○ 0.13
- ○ 0.25
- ○ 0.46
- ○ 0.78

Check Answer

Finish ▸

We can add these terms to the confusion matrix and add the row and column totals to get the following table:



n=125	**Predicted Good**	**Predicted Bad**	
Actual Good	TP = 50	FN = 34	**84**
Actual Bad	FP = 22	TN = 19	**41**
	72	**53**	

Next, we can determine the accuracy, or how often the classifier is correct with the model, by running the following code:

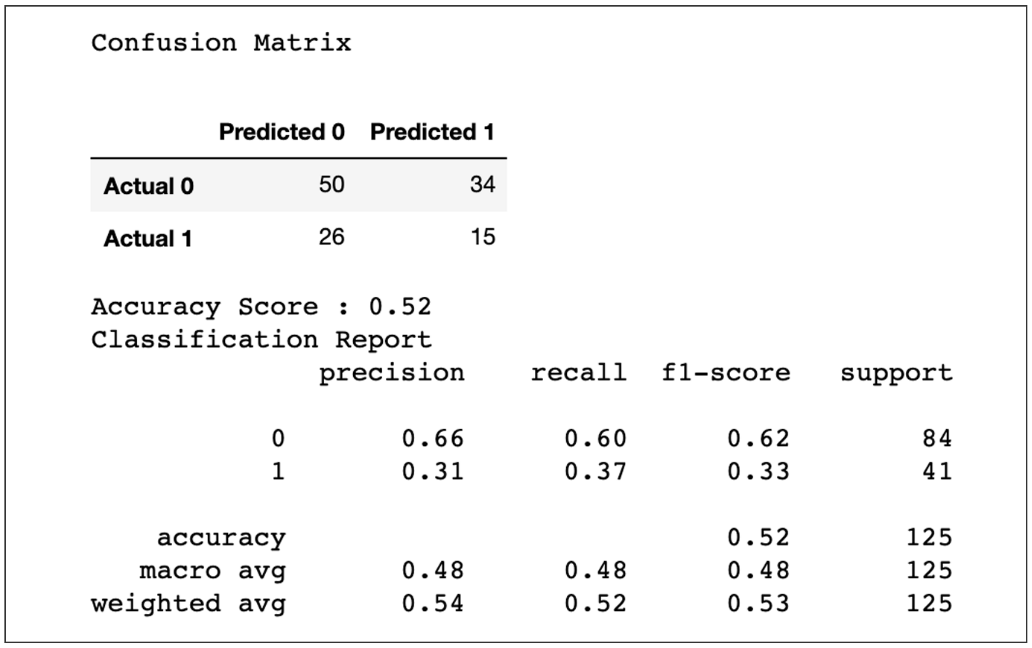
```
# Calculating the accuracy score.
acc_score = accuracy_score(y_test, predictions)
```

The accuracy of our model is 0.552, which can also be calculated as follows:

```
(True Positives (TP) + True Negatives (TN)) / Total = (50 + 19)/125 = 0.552
```

Lastly, we can print out the above results along with the classification report, which will give us the precision, recall, F1 score, and support for the two classes.

```
# Displaying results
print("Confusion Matrix")
display(cm_df)
print(f"Accuracy Score : {acc_score}")
print("Classification Report")
print(classification_report(y_test, predictions))
```



```
Confusion Matrix
```

	Predicted 0	Predicted 1
Actual 0	50	34
Actual 1	26	15

```
Accuracy Score : 0.52
Classification Report
              precision    recall  f1-score   support

           0       0.66      0.60      0.62        84
           1       0.31      0.37      0.33        41

    accuracy                           0.52       125
   macro avg       0.48      0.48      0.48       125
weighted avg       0.54      0.52      0.53       125
```

Let's go over the results in the classification report:

- **Precision:** Precision is the measure of how reliable a positive classification is. From our results, the precision for the good loan applications can be determined by the ratio TP/(TP + FP), which is 50/(50 + 22) = 0.69. The precision for the bad loan applications can be determined as follows: 19/(19 + 34) = 0.358. A low precision is indicative of a large number of false positives—of the 53 loan applications we predicted to be bad applications, 34 were actually good loan applications.
- **Recall:** Recall is the ability of the classifier to find all the positive samples. It can be determined by the ratio: TP/(TP + FN), or 50/(50 + 34) = 0.595 for the good loans and 19/(19 + 22) = 0.463 for the bad loans. A low recall is indicative of a large number of false negatives.
- **F1 score:** F1 score is a weighted average of the true positive rate (recall) and precision, where the best score is 1.0 and the worst is 0.0.
- **Support:** Support is the number of actual occurrences of the class in the specified dataset. For our results, there are 84 actual occurrences for the good loans and 41 actual occurrences for bad loans.

In summary, this model may not be the best one for preventing fraudulent loan applications because the model's accuracy, 0.552, is low, and the precision and recall are not good enough to state that the model will be good at classifying fraudulent loan applications. Modeling is an iterative process: you may need more data, more cleaning, another model parameter, or a different model. It's also important to have a goal that's been agreed upon, so that you know when the model is good enough.

NOTE

Consult the **sklearn.metrics.precision_recall_fscore_support documentation (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html)** for additional information about the classification scores.

© 2020 - 2022 Trilogy Education Services, a 2U, Inc. brand. All Rights Reserved.