Machine Learning – Classification

Michael Claudius, Associate Professor, Roskilde

15.09.2021

What is Machine Learning Classification

- **Classification:** Predict classes; e.g. digits, letters, faces
  - Correct prediction: Positive
  - Wrong prediction: Negative
- **Regression:** Predict values; e.g. slope 9.44, intersection 44..85
- **Both need a ML algorithms!**

Machine Learning Classification

Its about making data ready and to find the best classification model

1. Explore and prepare data
- Handle missing feature instances -> How to fix it

2. Metric
- Defining the ML Classification type

3. Train models on training set
- Training and Evaluating on the Training Set

4. Analyze the models by performance measures:
- Cross validation
- Confusion Matrix
- Precision – Recall
- ROC-curve

5. Choose the best model and launch

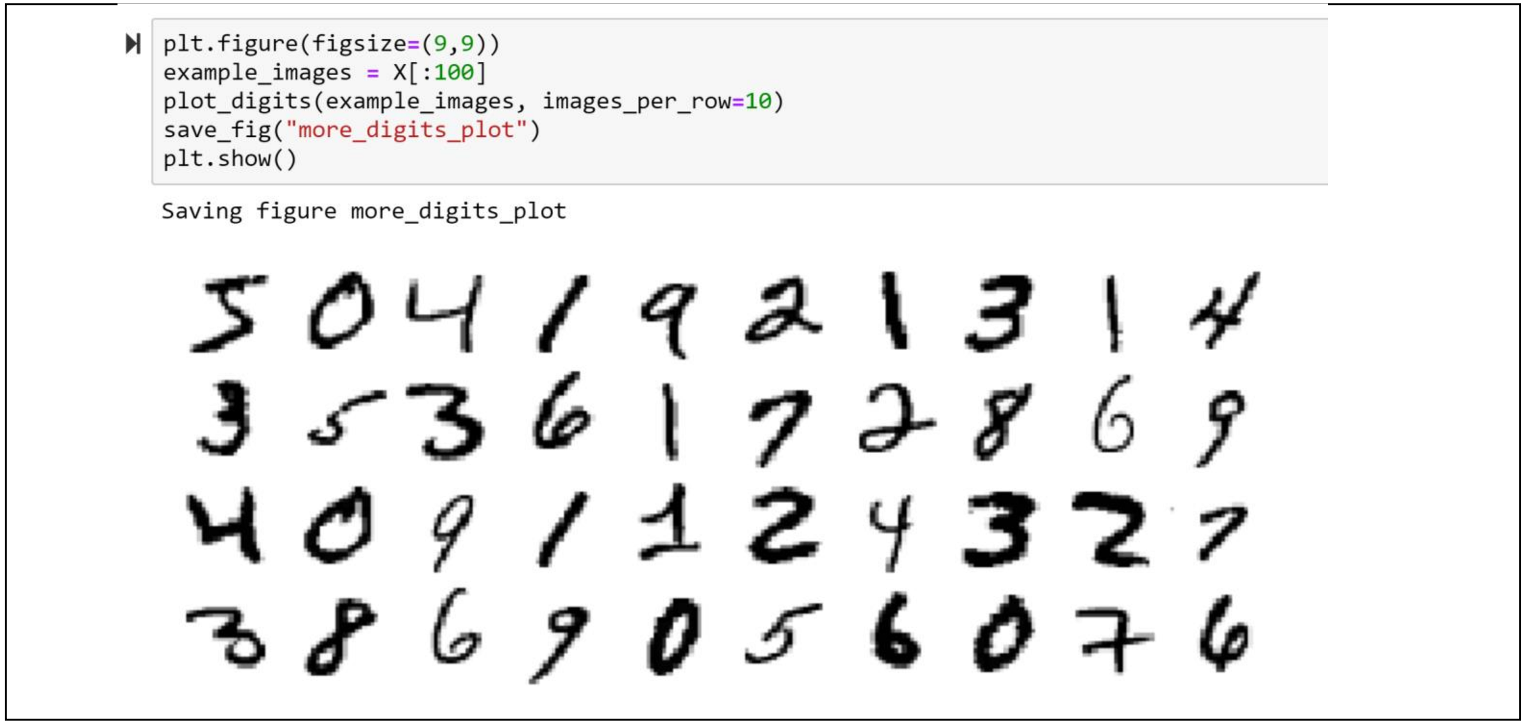
- **A detailed checklist is given on** [ML Management Checklist (PDF)]()
- **Remember always adapt the order and the checklist to your needs**

Classification Types

- **Binary:** One class (e.g. digit 5) and one Not-class (Not a digit 5)
- **Multiclass**: More classes (e.g. 10 digits: 0, 1, 2, ... 9)
- **Multilabel:** One class has several labels
  - e.g. larger than 7, odd? (e.g. is 5 larger than 7, is 5 odd?)
- **Multioutput:** Each label has 2 or more classes
  - e.g. 10 digits: 0, 1, 2, ... 9, color: blue, green red….
  - Picture class background: city, country-side, forest. Class foreground animal: dog, cat, bird

The Context: Mnist-784 Data Set

- **Pictures of 70.000 handwritten digits**
- **One picture 28x28 pixel**
- **Shuffled already: training set first 60.000 test set last 10.000 pictures**
- **Download form openlm.org/d/554**

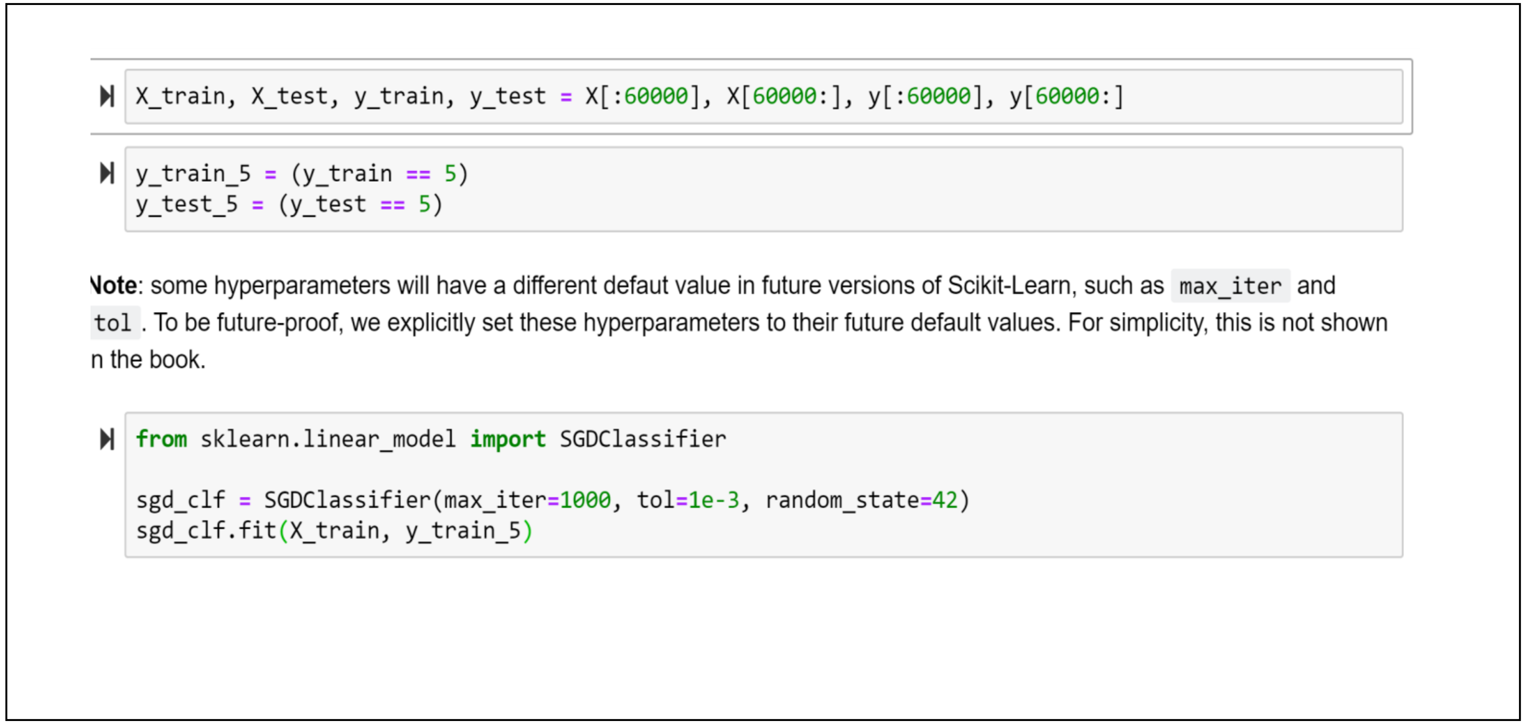


```
plt.figure(figsize=(9,9))
example_images = X[:100]
plot_digits(example_images, images_per_row=10)
save_fig("more_digits_plot")
plt.show()
```

```
Saving figure more_digits_plot
```

Classification Metrics

- **Binary 5 or Not-5**
- **Classifiers: Stochastic Gradient Descent (SGD) and Random Forest**



```
X_train, X_test, y_train, y_test = X[:60000], X[60000:], y[:60000], y[60000:]
```

```
y_train_5 = (y_train == 5)
y_test_5 = (y_test == 5)
```

Note: some hyperparameters will have a different defaut value in future versions of Scikit-Learn, such as `max_iter` and `tol`. To be future-proof, we explicitly set these hyperparameters to their future default values. For simplicity, this is not shown n the book.

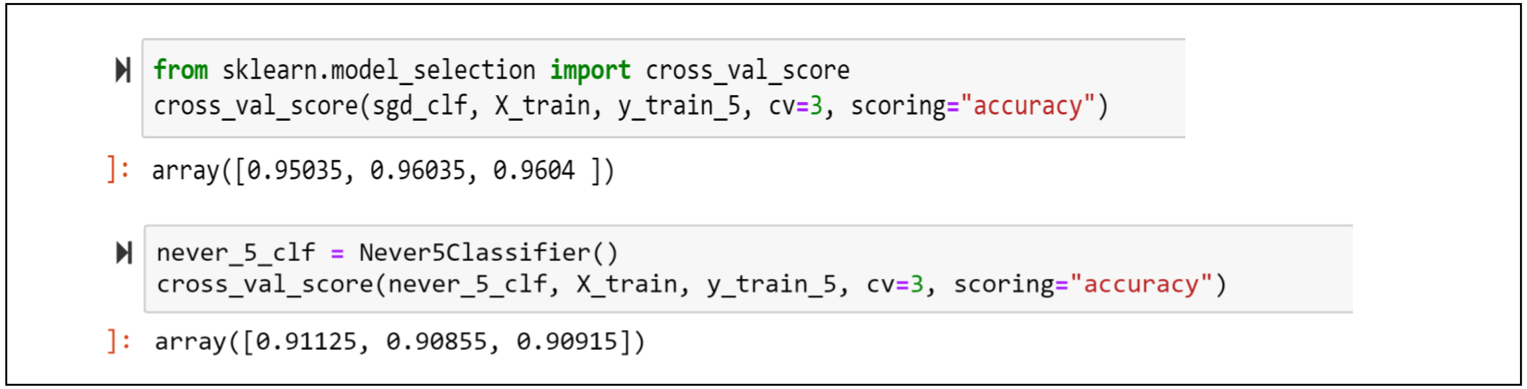
```
from sklearn.linear_model import SGDClassifier

sgd_clf = SGDClassifier(max_iter=1000, tol=1e-3, random_state=42)
sgd_clf.fit(X_train, y_train_5)
```

Performance Measures

- **Cross Validation**
- **Confusion Matrix**
- **Precision – Recall**
- **ROC-curve and AUC**

Performance Measure: Cross Validation

- **Cross validation with scoring accuracy and using 3 folds**



```
from sklearn.model_selection import cross_val_score
cross_val_score(sgd_clf, X_train, y_train_5, cv=3, scoring="accuracy")
```

```
]: array([0.95035, 0.96035, 0.9604 ])
```

```
never_5_clf = Never5Classifier()
cross_val_score(never_5_clf, X_train, y_train_5, cv=3, scoring="accuracy")
```

```
]: array([0.91125, 0.90855, 0.90915])
```

- **Predicting accuracy for 5: >95%**
- **Predicting accuracy for Not-5 > 90%**

- **Impressing or NOT?**
- **Actually NOT. Calm down. Remember 90% of numbers are Not-5!!**

- **Conclusion: Cross validation not the right performance measure for classification**

Performance Measure: Confusion Matrix

- **Divides data into:**
  - Actual Positive and Actual Negative
  - Predicted Positive and Predicted Negative

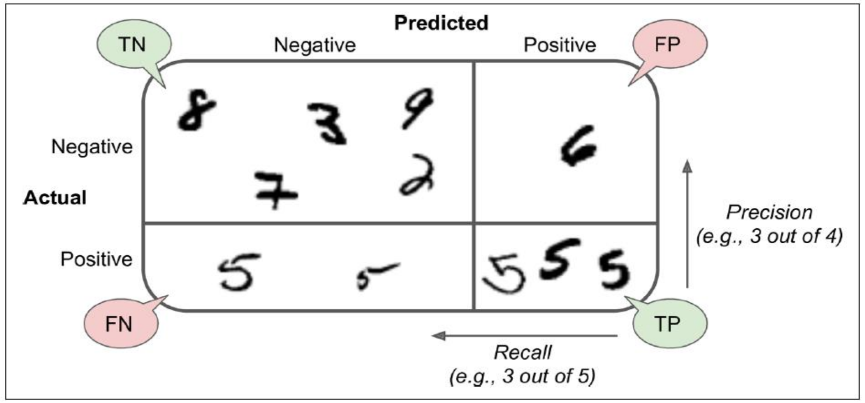
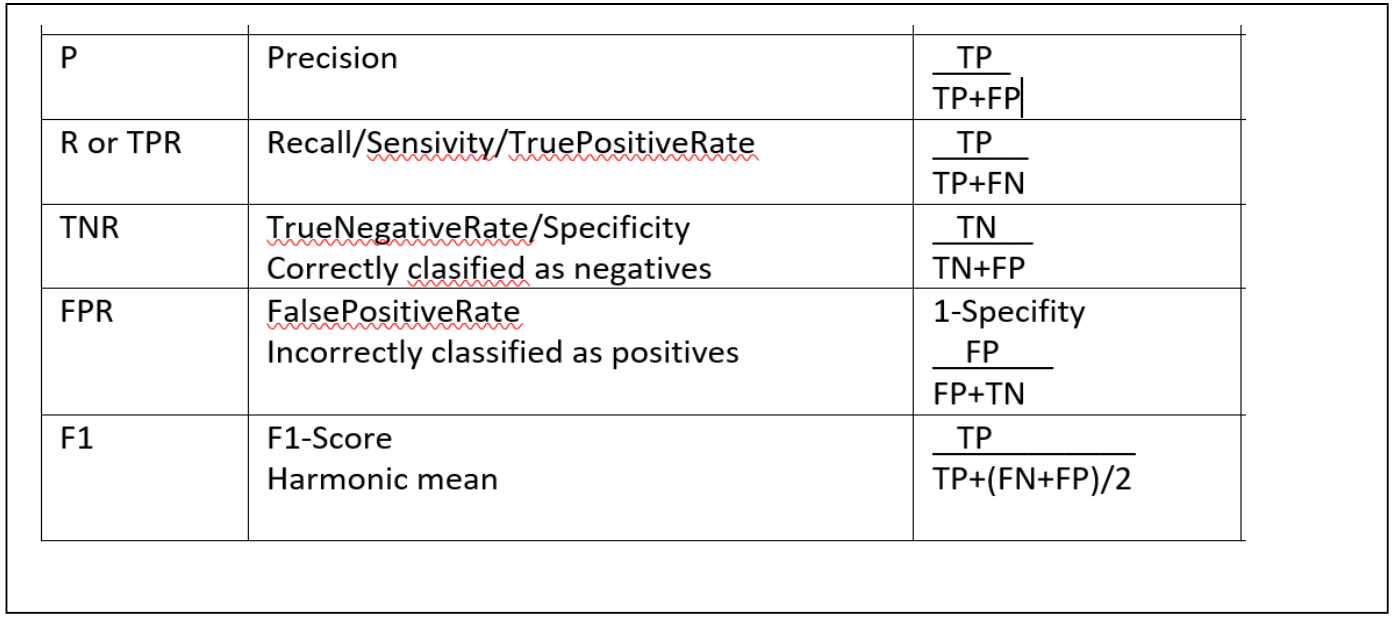


Figure 3-2. An illustrated confusion matrix shows examples of true negatives (top left), false positives (top right), false negatives (lower left), and true positives (lower right)

  - **TN:** True Negative, predicted *negative* and it is actual *negative* (Not-5)
  - **FN:** False Negative, predicted *negative* **but** the digit is actual *positive* (5)
  - **TP:** True Positive, predicted *positive* and the digit is actual *positive* (5)
  - **FP:** False Positive, predicted *positive* **but** the digit is actual *negative* (Not-5)

Confusion Matrix: Concise Metrics

- **Precision: Accuracy of positive predictions**
- **Recall: Accuracy of positive actuals, (ratio of positives correctly detected)**
- **F1: Harmonic mean of Precision and Recall**



P	Precision	$\frac{TP}{TP+FP}$
R or TPR	Recall/Sensivity/TruePositiveRate	$\frac{TP}{TP+FN}$
TNR	TrueNegativeRate/Specificity Correctly clasified as negatives	$\frac{TN}{TN+FP}$
FPR	FalsePositiveRate Incorrectly classified as positives	1-Specifity $\frac{FP}{FP+TN}$
F1	F1-Score Harmonic mean	$\frac{TP}{TP+(FN+FP)/2}$

- ***Confused About Confusion Matrix and Metrics . . . . LOL . . .***
- ***Don't worry lets go straight to an assignment:*** Classification Chapter 3 Questions **and solve it in 20 minutes !**

Precision/Recall Tradeoff: Example

- **Wanted: High Precision and High Recall !**
- **Not possible !!**
- **SGDClassifier computes a score based on a decision function based on a “threshold” value**

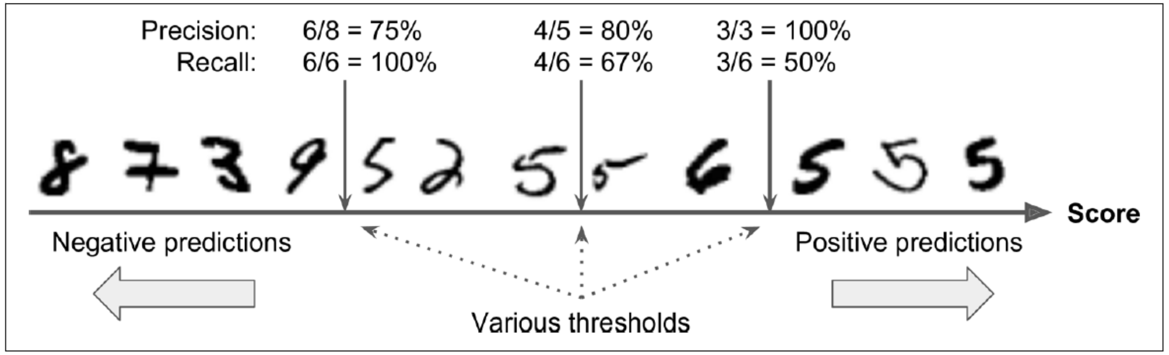


Figure 3-3. In this precision/recall trade-off, images are ranked by their classifier score, and those above the chosen decision threshold are considered positive; the higher the threshold, the lower the recall, but (in general) the higher the precision

- ***Depending on the “threshold “ high precision and low recall or opposite***

Precision/Recall Tradeoff: Curve

- **Want 0.90 (90%) precision. Select Threshold = 8.000. Gets Recall = 0.43 (43%)**

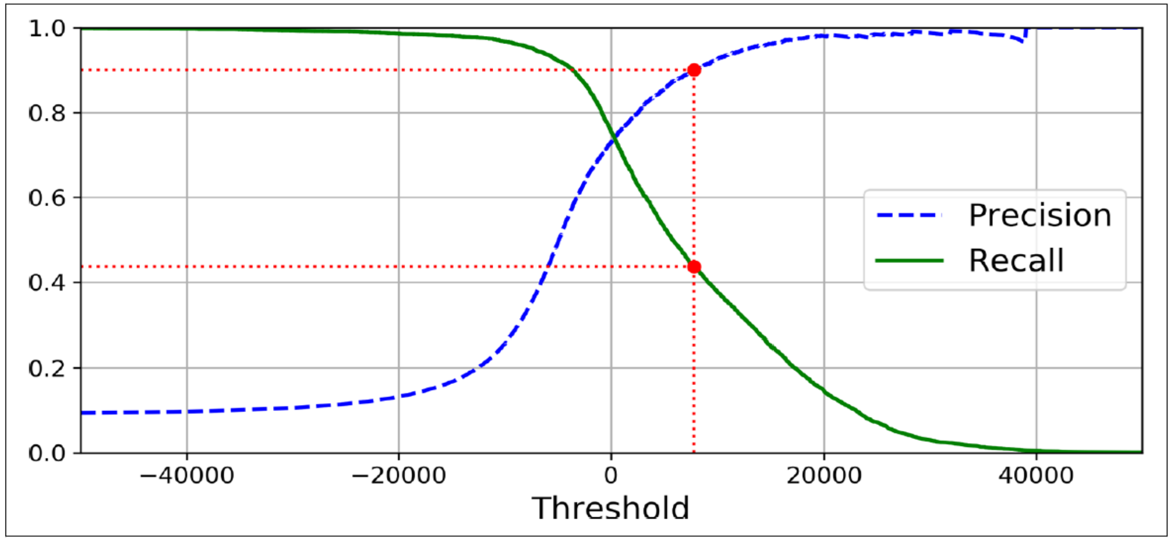


Figure 3-4. Precision and recall versus the decision threshold

- ***Depending on the “Threshold “ high precision and low recall or opposite***

ROC: Receiver Operating Classifier

- **The ROC curve plots True Positive Rate against False Positive Rate**
- **for many possible threshold values**
- **AUC: Area Under Curve**

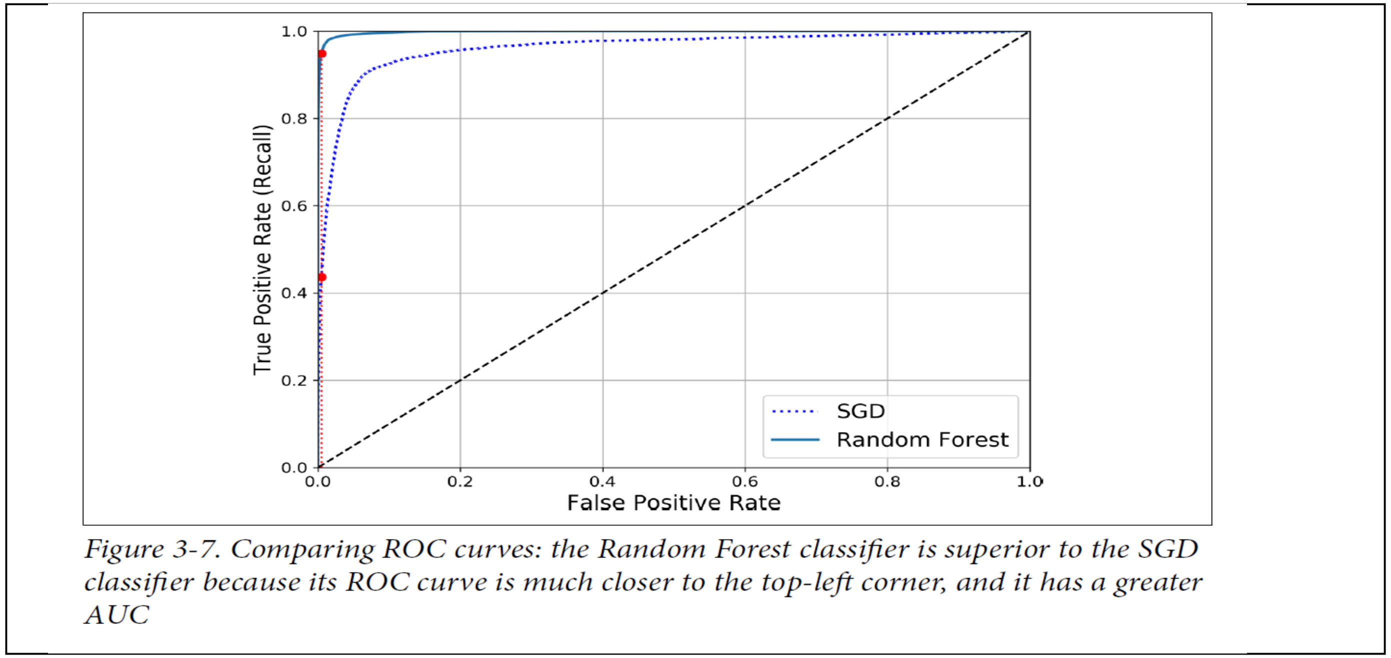


Figure 3-7. Comparing ROC curves: the Random Forest classifier is superior to the SGD classifier because its ROC curve is much closer to the top-left corner, and it has a greater AUC

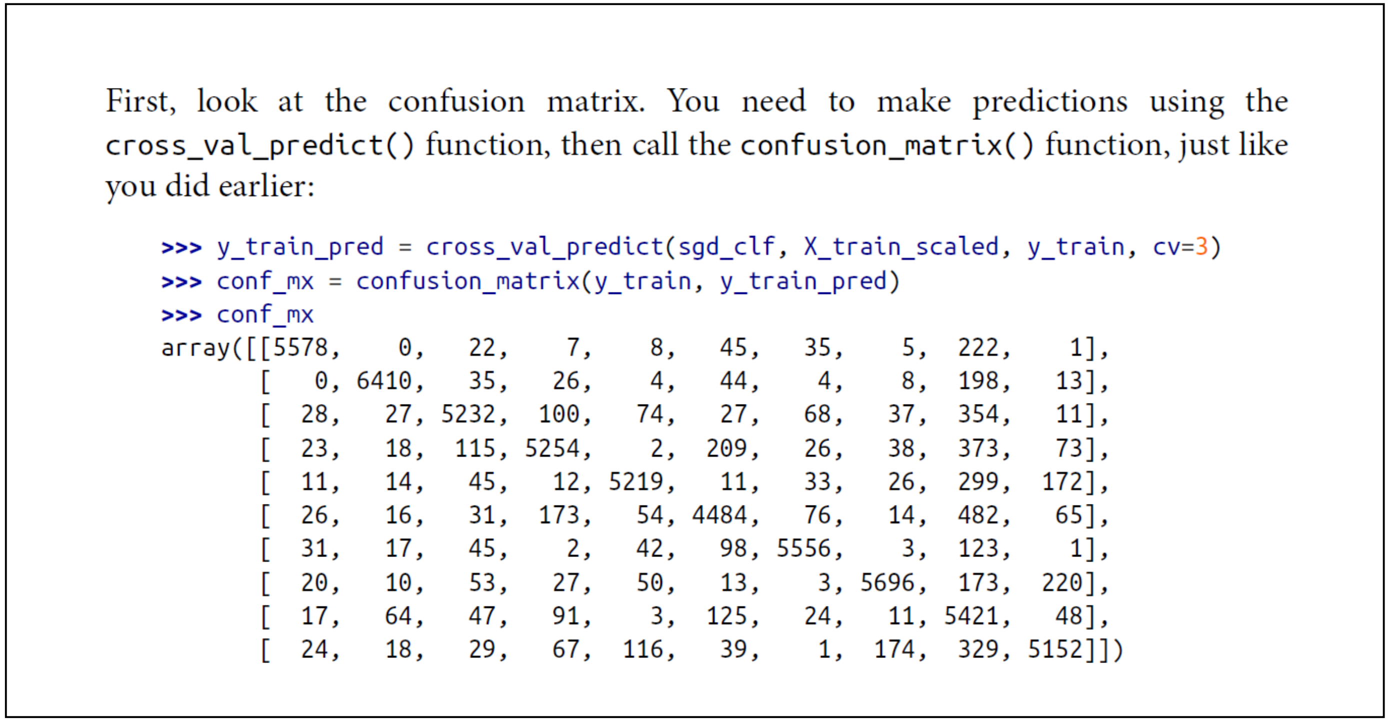
- **Objectives: Close to top left corner and AUC close to 1.0**
- **And the Winner is Random Forest! Precision 0.99, Recall 86.6%, AUC 0.998**

Precision/Recall (PR) vs. ROC

- **Normally ROC**
- **Precision/recall when positive class is seldom**
- **Precision/recall when False Positives (FP) are important**

Multi Class: ConfusionMatrix

- **2 or more classes, e.g. digits 0, 1, 2 . . . 9**
- **Use StandardScaler on pictures**



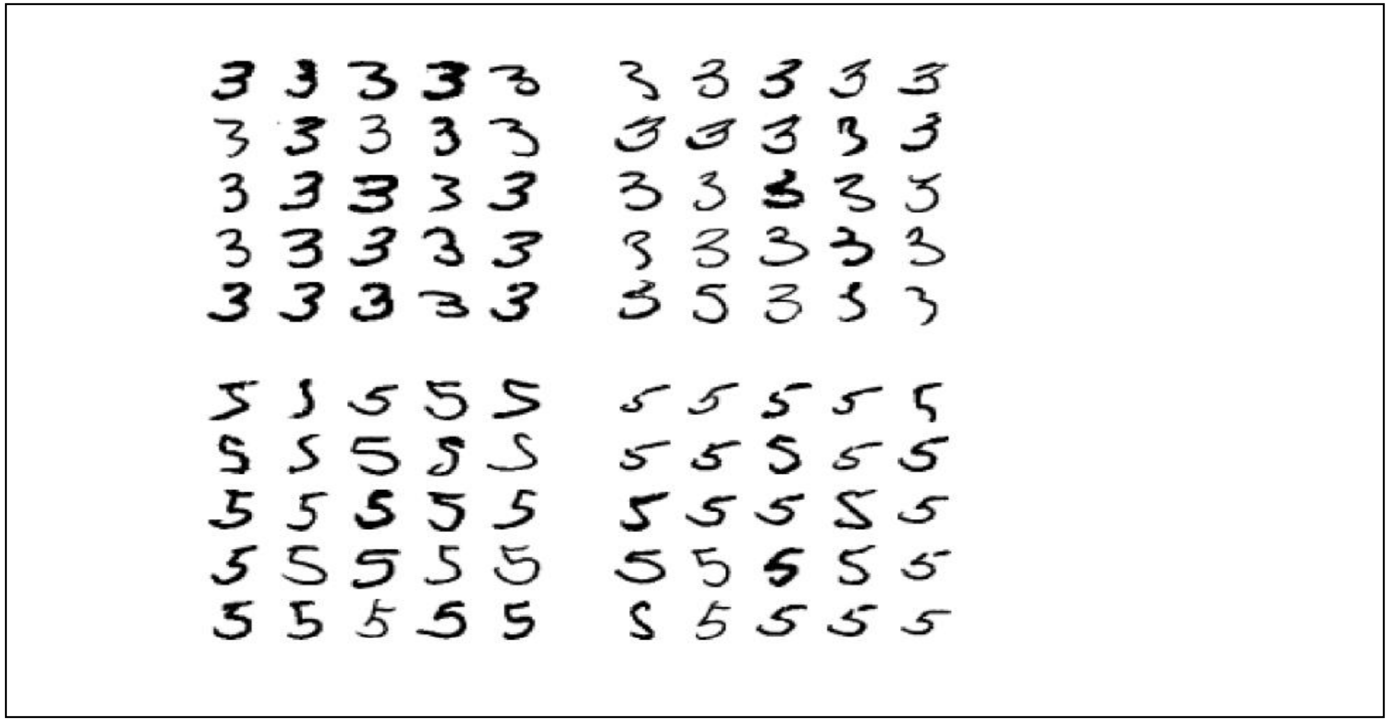
First, look at the confusion matrix. You need to make predictions using the `cross_val_predict()` function, then call the `confusion_matrix()` function, just like you did earlier:

```
>>> y_train_pred = cross_val_predict(sgd_clf, X_train_scaled, y_train, cv=3)
>>> conf_mx = confusion_matrix(y_train, y_train_pred)
>>> conf_mx
array([[5578,    0,   22,    7,    8,   45,   35,    5,  222,    1],
       [   0, 6410,   35,   26,    4,   44,    4,    8,  198,   13],
       [  28,   27, 5232,  100,   74,   27,   68,   37,  354,   11],
       [  23,   18,  115, 5254,    2,  209,   26,   38,  373,   73],
       [  11,   14,   45,   12, 5219,   11,   33,   26,  299,  172],
       [  26,   16,   31,  173,   54, 4484,   76,   14,  482,   65],
       [  31,   17,   45,    2,   42,   98, 5556,    3,  123,    1],
       [  20,   10,   53,   27,   50,   13,    3, 5696,  173,  220],
       [  17,   64,   47,   91,    3,  125,   24,   11, 5421,   48],
       [  24,   18,   29,   67,  116,   39,    1,  174,  329, 5152]])
```

- ***Observe: Many False-8***
- ***Observe: False-5 as 3, False 3 as 5***

Multi Class: Errors

- **Left column classified as 3**
- **Right column classified as 5**



- ***Conclusion Classifier sensitive to image rotation and shifting***

Exercise

- **It is time for discussing classification and exploring the MNIST-data set**

- Classfication MNIST Exercise