Machine Learning

| Error Analysis

Prof. Dr. Mostafa Elhosseini

https://youtube.com/@ProfElhosseiniTechVerse

Agenda

- Error Analysis

Error Analysis

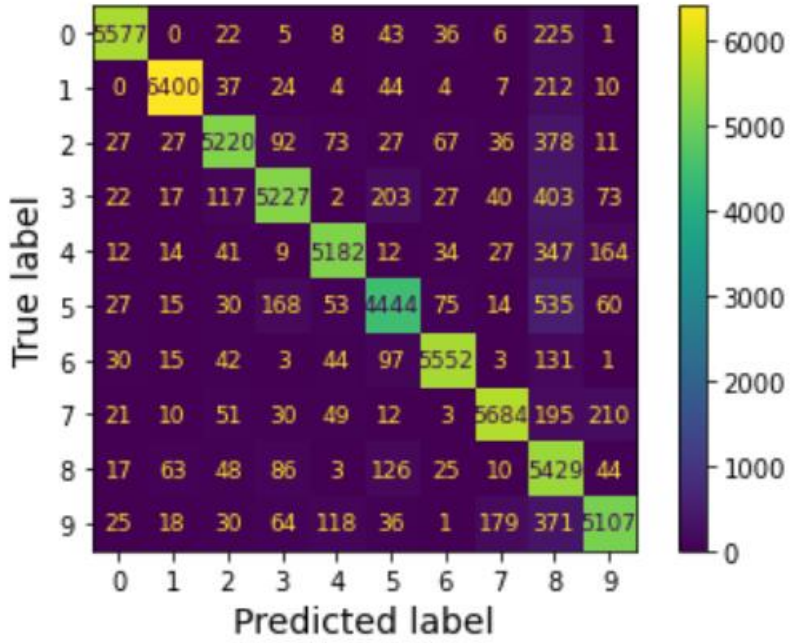
- Here, we will assume that you have found a promising model and you want to find ways to improve it. One way to do this is to **analyze the types of errors** it makes.
- First, you can look at the confusion matrix. You need to make predictions using the **cross_val_predict()** function, then call the **confusion_matrix()** function
- A colored diagram of the confusion matrix is much easier to analyze

Error Analysis

- A colored diagram of the confusion matrix is much easier to analyze

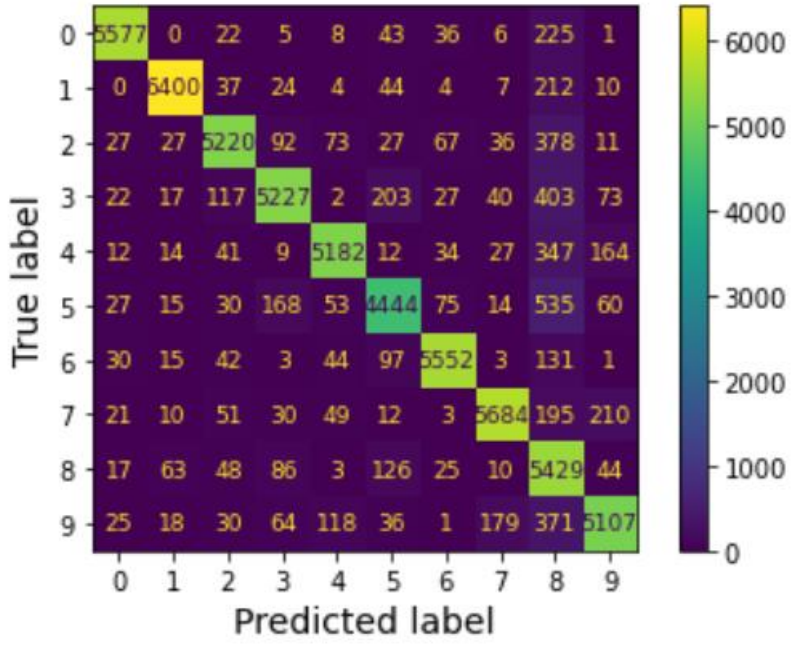
```
from sklearn.metrics import ConfusionMatrixDisplay

y_train_pred = cross_val_predict(sgd_clf, X_train_scaled, y_train, cv=3)
plt.rc('font', size=9)  # extra code – make the text smaller
ConfusionMatrixDisplay.from_predictions(y_train, y_train_pred)
plt.show()
```



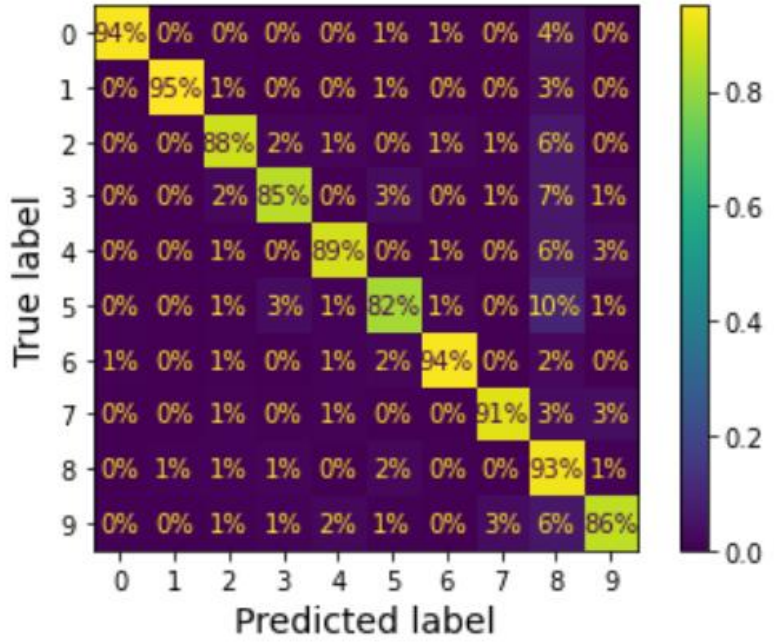
Error Analysis

- Most images are on the main diagonal, which means that they were classified correctly.
- Notice that the cell on the diagonal in row #5 and column #5 looks **slightly darker** than the other digits. This could be because the model made more errors on 5s, or because there are fewer 5s in the dataset than the other digits



Error Analysis

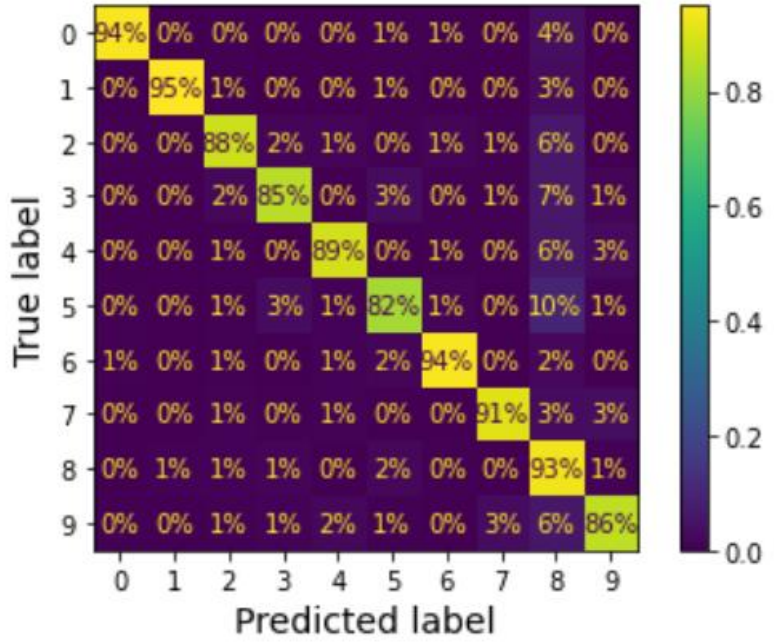
- That's why it's important to normalize the confusion matrix by dividing each value by the total number of images in the corresponding (true) class (i.e., divide by the row's sum).




```
plt.rc('font', size=10)
ConfusionMatrixDisplay.from_predictions(y_train, y_train_pred,
                                        normalize="true", values_format=".0%")
plt.show()
```

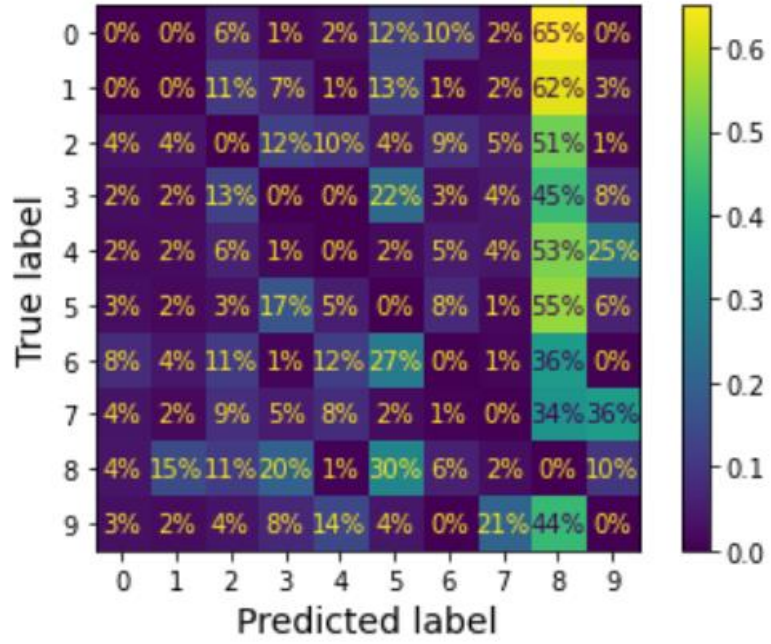
Error Analysis

- only 82% of the images of 5s were classified correctly.
- The most common error the model made with images of 5s was to misclassify them as 8s: this happened for 10% of all 5s.
- But only 2% of 8s got misclassified as 5s
- Confusion matrices are generally not symmetrical



Error Analysis

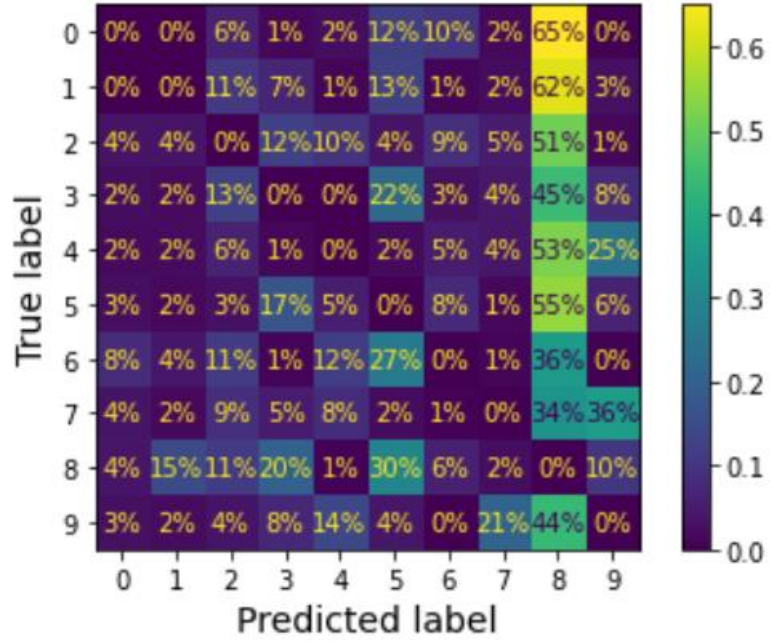
- If you want to make the errors stand out more, you can try putting zero weight on the correct predictions




```
sample_weight = (y_train_pred != y_train)
plt.rc('font', size=10)  # extra code
ConfusionMatrixDisplay.from_predictions(y_train, y_train_pred,
                                        sample_weight=sample_weight,
                                        normalize="true", values_format=".0%")
plt.show()
```

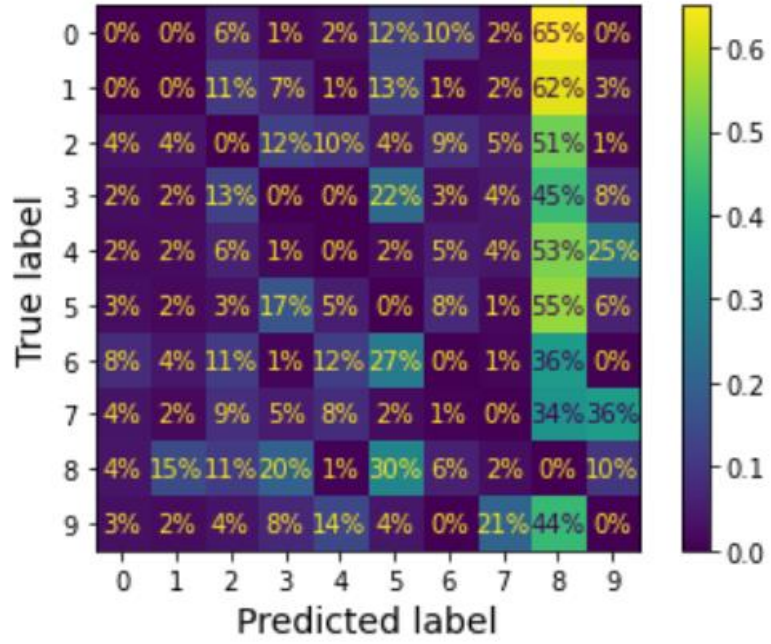
Error Analysis

- Remember that rows represent actual classes, while columns represent predicted classes.
- The column for class 8 is now really bright, which confirms that many images got misclassified as 8s
- In fact this is the most common misclassification for almost all classes. But be careful how you interpret the percentages in this diagram: remember that we've excluded the correct predictions



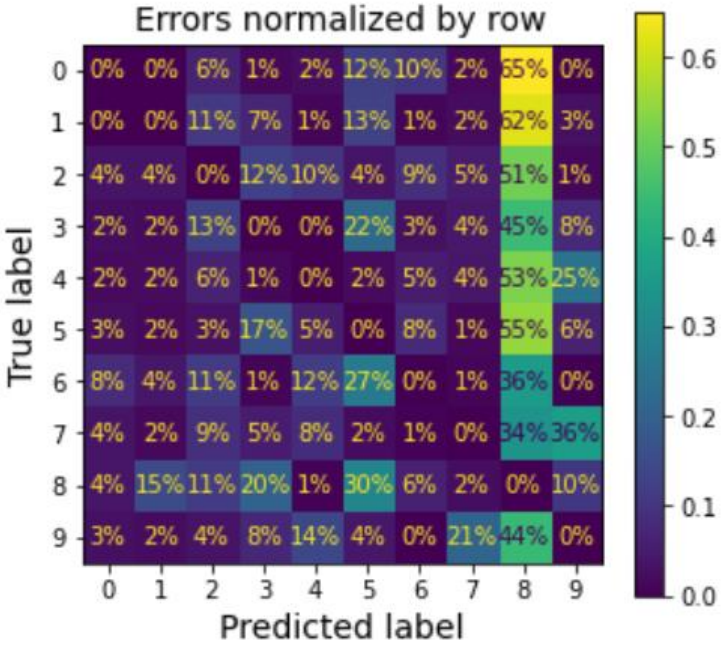
Error Analysis

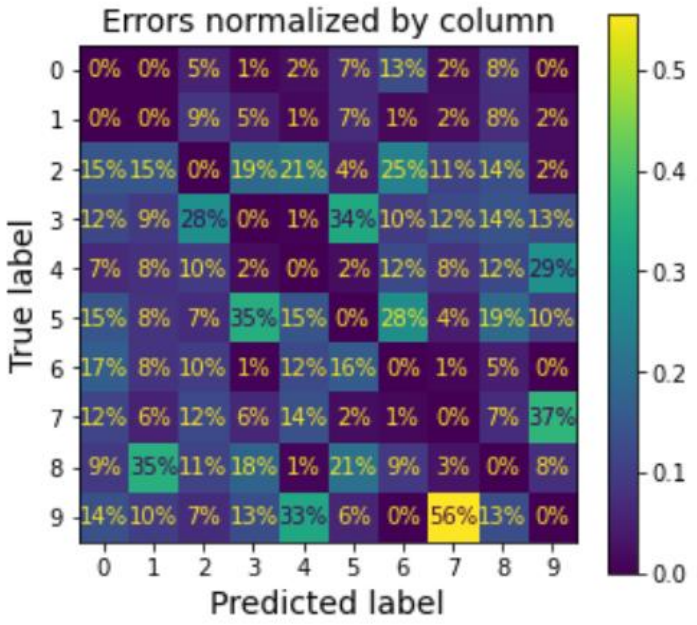
- For example, the 36% in row #7, column #9 does not mean that 36% of all images of 7s were misclassified as 9s.
- It means that 36% of the errors the model made on images of 7s were misclassifications as 9s.
- In reality, only 3% of images of 7s were misclassified as 9s



Error Analysis

- It is also possible to normalize the confusion matrix by column rather than by row
- you can see that 56% of misclassified 7s are actually 9s.



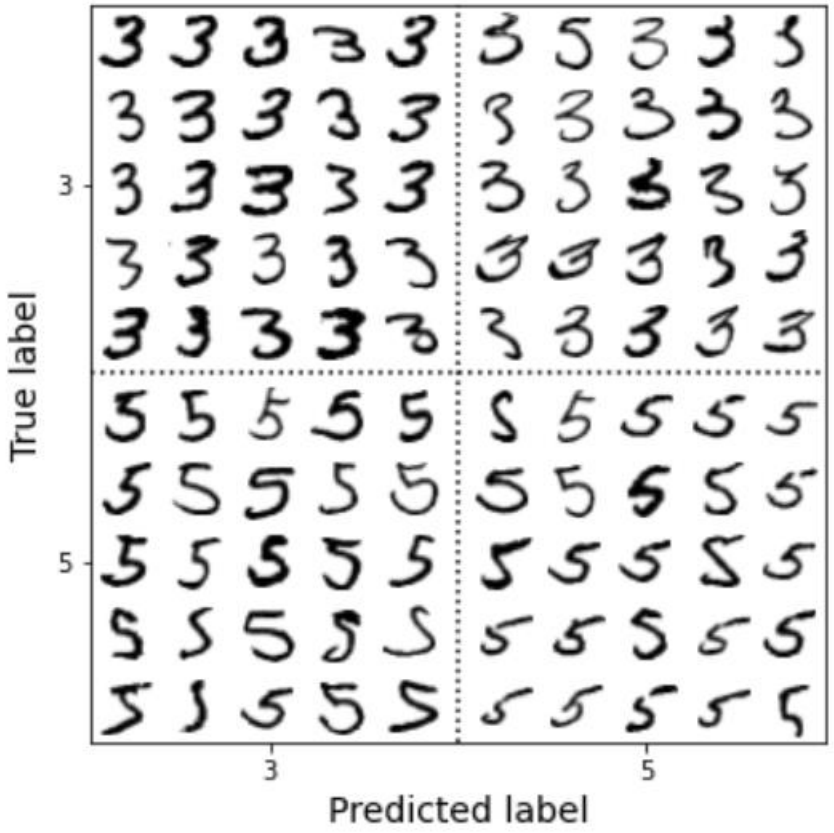


Error Analysis

- Analyzing the confusion matrix often gives you insights into ways to improve your classifier.
- Looking at these plots, it seems that your efforts should be spent on reducing the false 8s. For example, you could try to gather more training data for digits that look like 8s (but are not) so that the classifier can learn to distinguish them from real 8s.
- Or you could engineer new features that would help the classifier—for example, writing an algorithm to count the number of closed loops (e.g., 8 has two, 6 has one, 5 has none). Or you could preprocess the images (e.g., using Scikit-Image, Pillow, or OpenCV) to make some patterns, such as closed loops, stand out more.

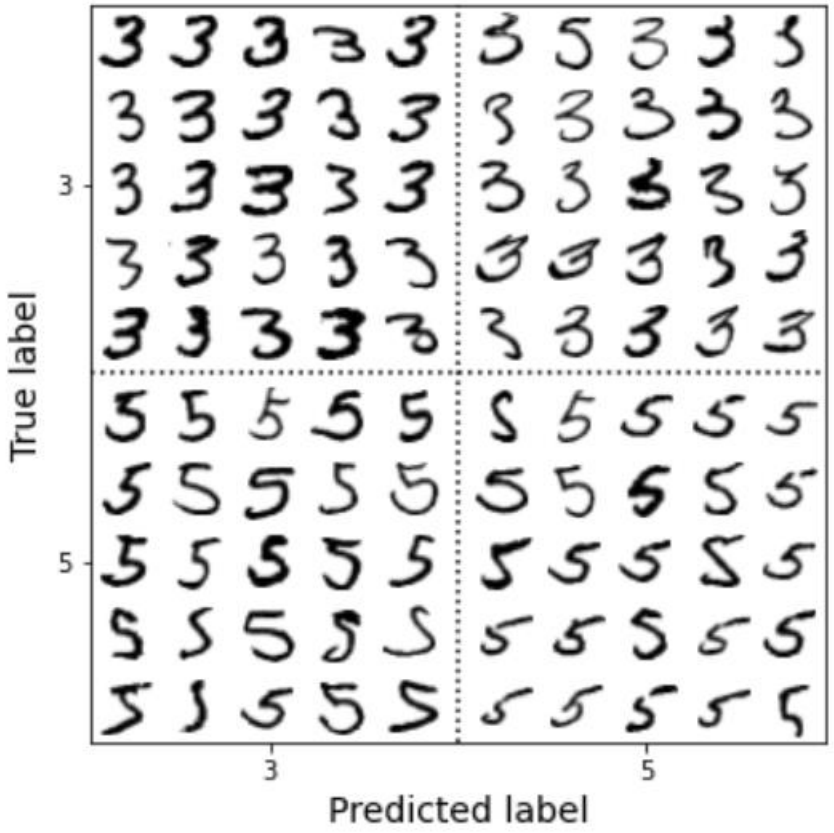
Error Analysis

- Analyzing individual errors can also be a good way to gain insights into what your classifier is doing and why it is failing
- some of the digits that the classifier gets wrong (i.e., in the bottom-left and top-right blocks) are so badly written that even a human would have trouble classifying them



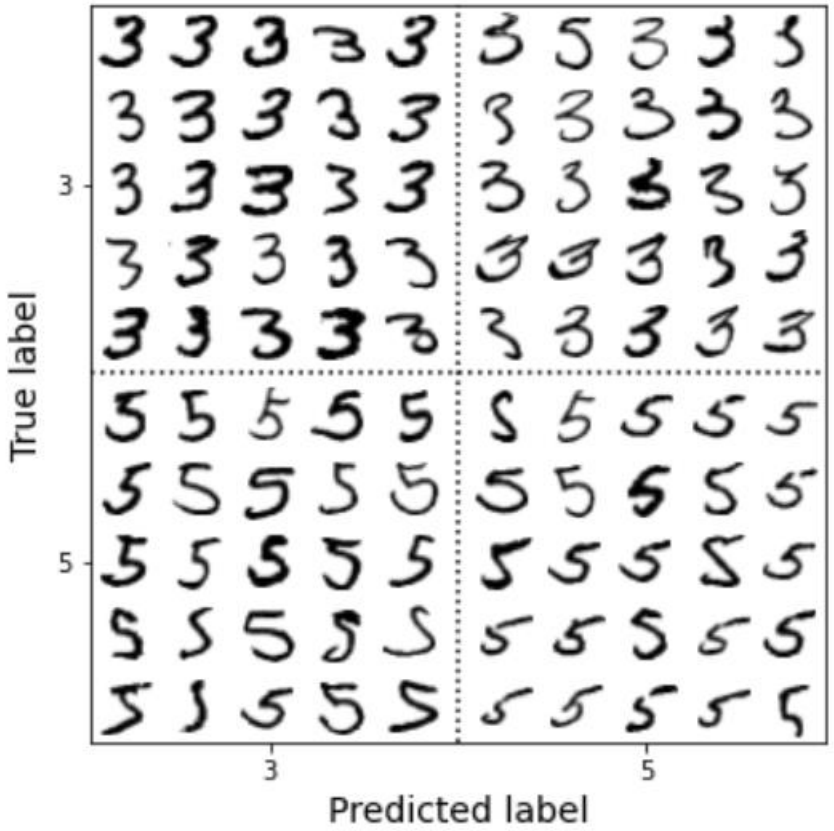
Error Analysis

- Recall that we used a simple SGDClassifier, which is just a linear model: all it does is assign a weight per class to each pixel, and when it sees a new image it just sums up the weighted pixel intensities to get a score for each class.
- Since 3s and 5s differ by only a few pixels, this model will easily confuse them.



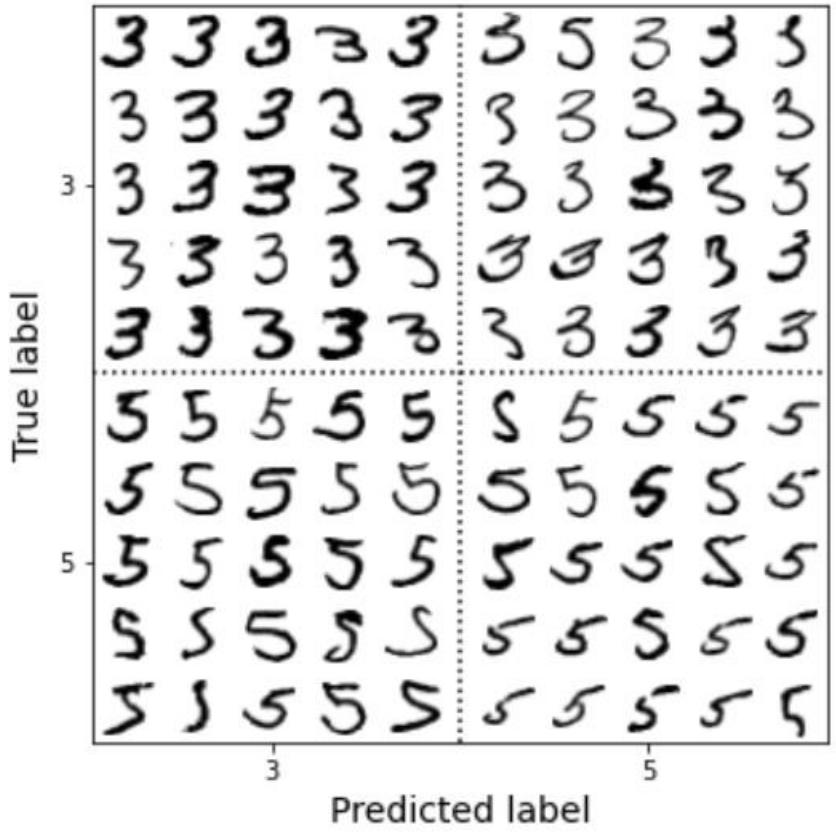
Error Analysis

- The main difference between 3s and 5s is the position of the small line that joins the top line to the bottom arc.
- If you draw a 3 with the junction slightly shifted to the left, the classifier might classify it as a 5, and vice versa.
- In other words, this classifier is quite sensitive to image shifting and rotation



Error Analysis

- One way to reduce the 3/5 confusion is to preprocess the images to ensure that they are well centered and not too rotated.
- However, this may not be easy since it requires predicting the correct rotation of each image.
- A much simpler approach consists of augmenting the training set with slightly shifted and rotated variants of the training images. This will force the model to learn to be more tolerant to such variations. This is called **data augmentation**



Data Augmentation

- To artificially increase the size and diversity of a training dataset without actually collecting new data. The goal is to improve the model's ability to generalize and reduce overfitting.
- It involves applying various transformations to the existing data, such as rotation, flipping, scaling, cropping, and adding noise, to create new, slightly modified versions of the original data.
- These transformations are designed to maintain the essential features and labels of the original data while introducing some variability.

Data Augmentation

- Data augmentation is especially useful when the available training data is limited or when the model needs to be more robust to variations in input data.
- It is commonly used in image classification tasks, where the model needs to learn to recognize objects regardless of their orientation, size, or position in the image.
- Data augmentation can also be applied to other types of data, such as audio (e.g., pitch shifting, time stretching) and text (e.g., synonym replacement, random word insertion/deletion).
- Popular deep learning frameworks, such as TensorFlow and PyTorch, provide built-in utilities for applying data augmentation techniques.

Data Augmentation

- Data augmentation is especially useful when the available training data is limited or when the model needs to be more robust to variations in input data.
- It is commonly used in image classification tasks, where the model needs to learn to recognize objects regardless of their orientation, size, or position in the image.
- Data augmentation can also be applied to other types of data, such as audio (e.g., pitch shifting, time stretching) and text (e.g., synonym replacement, random word insertion/deletion).
- Popular deep learning frameworks, such as TensorFlow and PyTorch, provide built-in utilities for applying data augmentation techniques.

Next Lecture

- **Multilabel classification**—A classification task where each input sample can be assigned multiple labels. For instance, a given image may contain both a cat and a dog and should be annotated both with the “cat” label and the “dog” label. The number of labels per image is usually variable.

Next Lecture

- classify **Reuters newswires** into **46 mutually exclusive topics**. Because we have many classes, this problem is an instance of **multiclass classification**, and because each data point should be classified into **only one category**, the problem is more specifically an instance of **single-label multiclass classification**.
- If each data point could belong to **multiple categories** (in this case, topics), we'd be facing a **multilabel multiclass classification problem**.

Next Lecture

- Reuters dataset, a set of short newswires and their topics, published by Reuters in 1986. It's a simple, widely used toy dataset for text classification. There are 46 different topics; some topics are more represented than others, but each topic has at least 10 examples in the training set.
- Like IMDB and MNIST, the Reuters dataset comes packaged as part of Keras