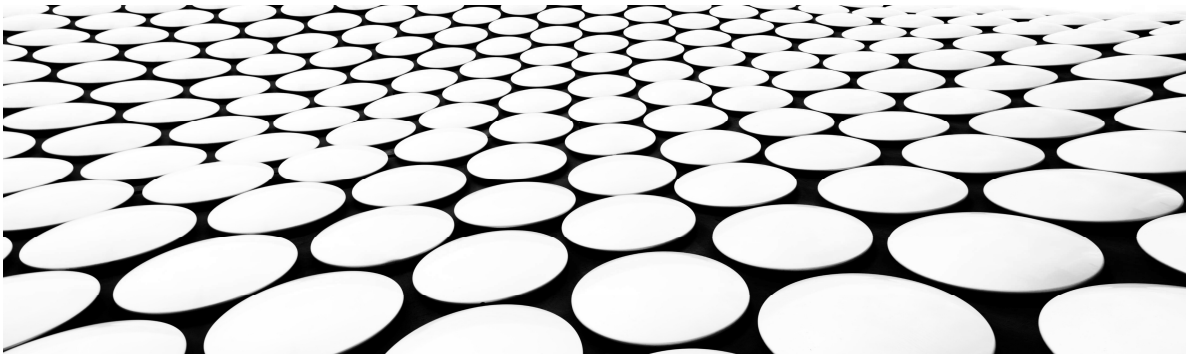
CSE353 – MACHINE LEARNING
CLASSIFICATION

PRAVIN PAWAR, SUNY KOREA

BASED ON CHAPTER 3 – HANDS-ON ML WITH SCIKIT-LEARN, KERAS AND TENSORFLOW BY AURÉLIEN GÉRON



1

CONTENTS

2

SOME SCIKIT-LEARN DATASETS

- Boston house prices dataset (regression)
- Iris plants dataset (classification)
- Diabetes dataset (regression)
- Optical recognition of handwritten digits dataset (classification)
- UCI breast cancer dataset (classification)
- The Olivetti faces dataset (classification)
- 20 newsgroups dataset (classification)
- RCV1 multilabel dataset (classification)
- Labeled Faces in the Wild (LFW) people dataset (classification)
- Labeled Faces in the Wild (LFW) pairs dataset (classification)
- California housing dataset (regression)

3

MNIST (MODIFIED NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY) DATASET

- Contains 70,000 small images of digits handwritten by high school students and employees of US Census Bureau
- Each image is 28x28 pixels – thus total 784 features (Each feature represents pixel intensity 0 to 255)
- Image label consists of digits it represents

```
from sklearn.datasets import fetch_openml
mnist = fetch_openml('mnist_784', version=1)
mnist.keys()
```

```
dict_keys(['data', 'target', 'feature_names', 'DESCR',
'details', 'categories', 'url'])
```

```
X, y = mnist["data"], mnist["target"]
X.shape
```

```
(70000, 784)
```

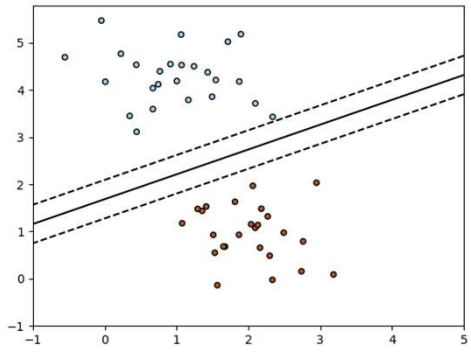
```
y.shape
```

```
(70000,)
```

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

4

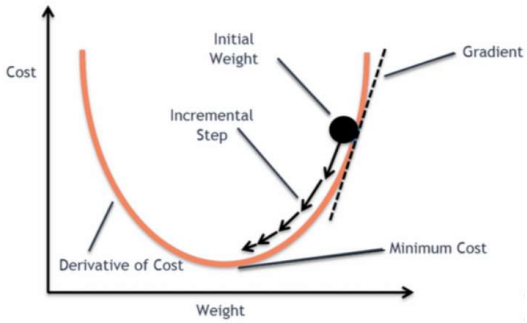
TRAINING A BINARY CLASSIFIER – CONSIDER THE CASE WHERE DIGIT = 5



- Stochastic gradient descent (SGD)
- In SGD, it uses only a single sample, i.e., a batch size of one, to perform each iteration
- The sample is randomly shuffled and selected for performing the iteration

```
y_train_5 = (y_train == 5)
y_test_5 = (y_test == 5)
print(y_train_5.shape)
print(y_test_5.shape)
```

```
(60000,)
(10000,)
```



5

APPLYING STOCHASTIC GRADIENT DESCENT CLASSIFIER

```
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)
```

```
SGDClassifier(alpha=0.0001, average=False, class_weight=None,
       early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
       l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=1000,
       n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
       power_t=0.5, random_state=42, shuffle=True, tol=0.001,
       validation_fraction=0.1, verbose=0, warm_start=False)
```

```
some_digit = X[0]
sgd_clf.predict([some_digit])
```

```
array([ True])
```

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

```
array([0.96355, 0.93795, 0.95615])
```

6

COMPARING CROSS VALIDATION ACCURACY WITH A DUMB CLASSIFIER – NOT 5

```
from sklearn.base import BaseEstimator
class Never5Classifier(BaseEstimator):
    def fit(self, X, y=None):
        pass
    def predict(self, X):
        return np.zeros((len(X), 1), dtype=bool)
```

```
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])
```

- A dumb classifier has over 90% accuracy!!
- Because only about 10% of the images are 5s
- Hence accuracy is generally not a preferred performance measure for classifiers – specially dealing with skewed datasets

7

CONFUSION MATRIX

		Predicted Class: Positive	Predicted Class: Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

Confusion matrix is a better approach to evaluate performance of a classifier

The general idea is to count the number of times instances of class A are classified as class B

8

CONFUSION MATRIX

```
from sklearn.model_selection import cross_val_predict
y_train_pred = cross_val_predict(sgd_clf, X_train, y_train_5, cv=3)
```

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_train_5, y_train_pred)
```

```
array([[53057,  1522],
       [ 1325,  4096]])
```

```
y_train_perfect_predictions = y_train_5  # pretend we reached perfection
confusion_matrix(y_train_5, y_train_perfect_predictions)
```

```
array([[54579,     0],
       [    0,  5421]])
```

9

MORE CONCISE METRICS

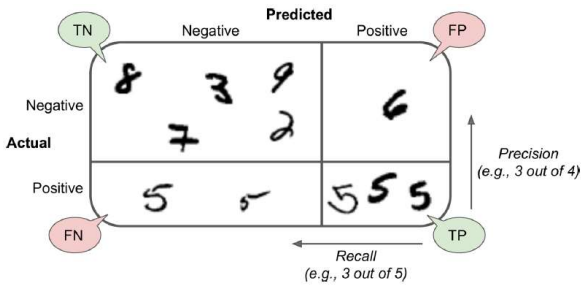
Precision

$$\frac{TP}{(TP + FP)}$$

Sensitivity

$$\frac{TP}{(TP + FN)}$$

- Precision: Accuracy of positive predictions
- Recall (sensitivity): True positive rate represents the ratio of positive instances that are correctly detected by the classifier



10

BINARY CLASSIFIER – PRECISION & RECALL

```
from sklearn.metrics import precision_score, recall_score
precision_score(y_train_5, y_train_pred)
```

```
0.7290850836596654
```

```
4096 / (4096 + 1522)
```

```
0.7290850836596654
```

```
recall_score(y_train_5, y_train_pred)
```

```
0.7555801512636044
```

```
4096 / (4096 + 1325)
```

```
0.7555801512636044
```

11

COMBINE PRECISION AND RECALL INTO F_1 SCORE

$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{TP}{TP + \frac{FN + FP}{2}}$$

- The F1 score is the harmonic mean of precision and recall
- Whereas the regular mean treats all values equally, the harmonic mean gives much more weight to low values
- Hence the classifier will get a high F1 score only if both recall and precision are high.

```
from sklearn.metrics import f1_score

f1_score(y_train_5, y_train_pred)
```

```
0.7420962043663375
```

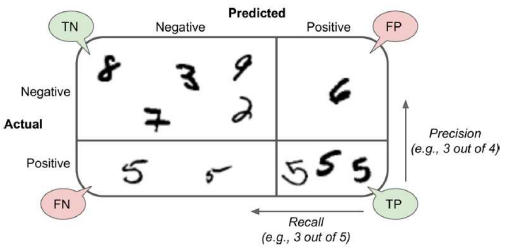
```
4096 / (4096 + (1522 + 1325) / 2)
```

```
0.7420962043663375
```

12

HIGH PRECISION VS HIGH RECALL

- High precision example
  - Sometimes high precision is desired (means less False Positive)
  - E.g. detect videos suitable for children (A suitable video is considered Positive)
  - A classifier that rejects lot of suitable videos, but never shows adult content has high precision
- High recall example
  - Sometimes high recall is desired (means less False Negative)
  - It would be fine id the classifier has only 25% precision (lot of false alarms or False Positive)
  - But should alarm every time when someone breaks in (less False Negative)



Precision

$$\frac{TP}{(TP + FP)}$$

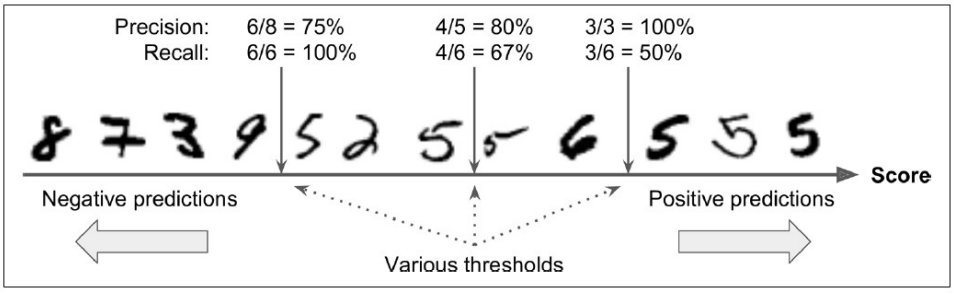
Sensitivity

$$\frac{TP}{(TP + FN)}$$

13

PRECISION-RECALL TRADEOFF IN BINARY CLASSIFIER

- A SGD classifier computes score based on a decision function
- If score greater than a threshold, an instance is positive otherwise it is negative
- Depending on the threshold level, precision and recall values show a tradeoff.
- High precision corresponds to low recall and vice-versa



Precision

$$\frac{TP}{(TP + FP)}$$

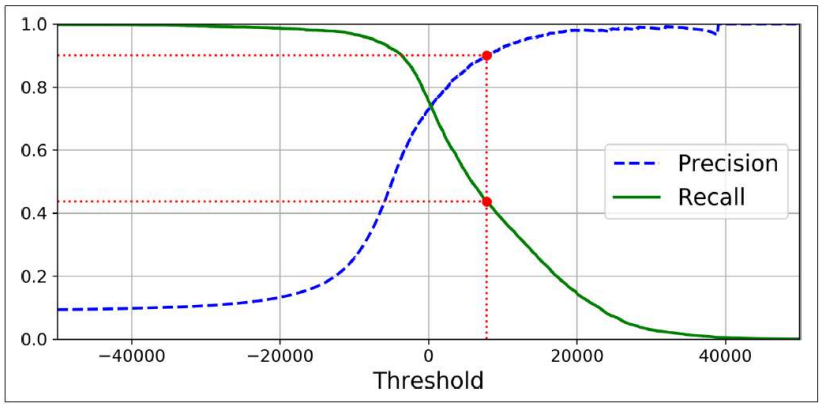
Sensitivity

$$\frac{TP}{(TP + FN)}$$

14

PRECISION-RECALL TRADEOFF IN BINARY CLASSIFIER

- The trade-off between precision and recall can be observed using the precision-recall curve
- It lets you spot which threshold is the best



Precision

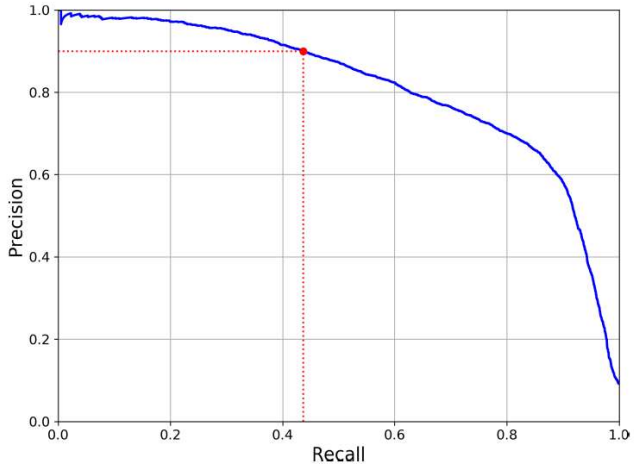
$$\frac{TP}{(TP + FP)}$$

Sensitivity

$$\frac{TP}{(TP + FN)}$$

15

PRECISION-RECALL TRADEOFF IN BINARY CLASSIFIER



- Another way to select a good precision/recall trade-off is to plot precision directly against recall (the same threshold as earlier is highlighted)
- Precision really starts to fall sharply around 80% recall
- Probably want to select a precision/recall trade-off just before that drop
- If someone says, “Let’s reach 99% precision,” you should ask, “At what recall?”

16

THE RECEIVER OPERATING CHARACTERISTIC (ROC) CURVE

- ROC curve is another common tool used with binary classifiers
- Similar to the precision/recall curve
- ROC curve plots the true positive rate (recall) against the false positive rate (FPR)
- The FPR is the ratio of negative instances that are incorrectly classified as positive
- It is equal to 1 – the true negative rate (TNR), which is the ratio of negative instances that are correctly classified as negative
- The TNR is also called specificity
- Hence, the ROC curve plots sensitivity (recall) versus 1 – specificity

Sensitivity

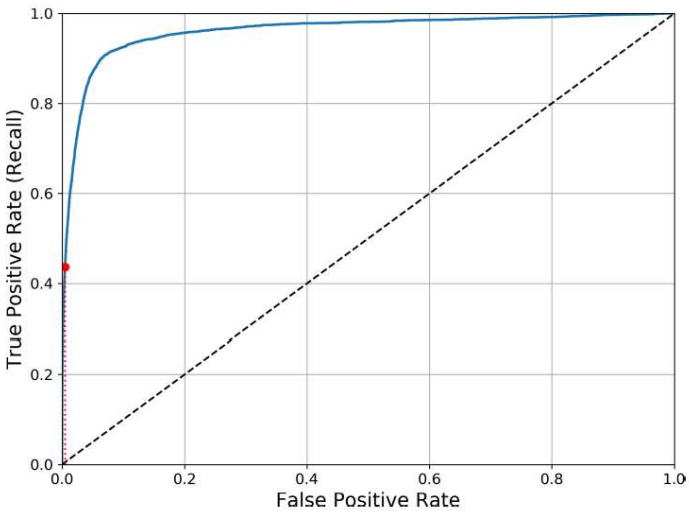
$$\frac{TP}{(TP + FN)}$$

Specificity

$$\frac{TN}{(TN + FP)}$$

17

THE RECEIVER OPERATING CHARACTERISTIC (ROC) CURVE



- Once again there is a trade-off
- The higher the recall (TPR), the more false positives (FPR) the classifier produces
- The dotted line represents the ROC curve of a purely random classifier
- A good classifier stays as far away from that line as possible (toward the top-left corner)
- Red dot represents chosen threshold
- Area Under Curve (AUC) is another criteria for comparing classifiers
- A purely random classifier has ROC AUC = 0.5

```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_train_5, y_scores)
```

```
0.9611778893101814
```

18

THE RECEIVER OPERATING CHARACTERISTIC (ROC) CURVE

```
from sklearn.metrics import roc_curve

fpr, tpr, thresholds = roc_curve(y_train_5, y_scores)
```

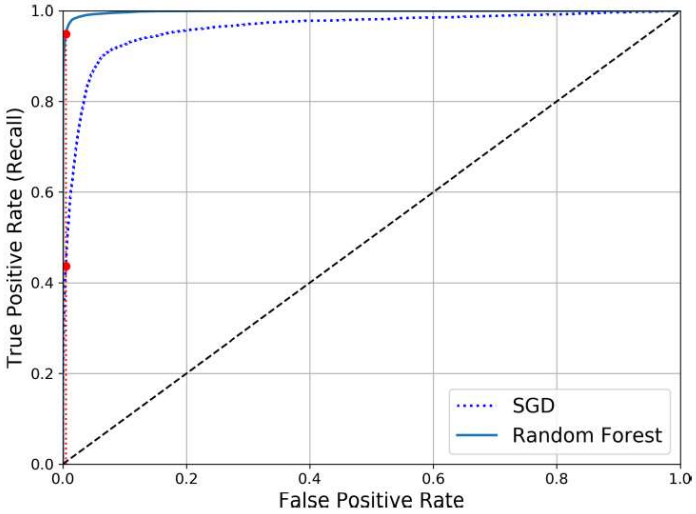
```
def plot_roc_curve(fpr, tpr, label=None):
    plt.plot(fpr, tpr, linewidth=2, label=label)
    plt.plot([0, 1], [0, 1], 'k--') # dashed diagonal
    plt.axis([0, 1, 0, 1])
    plt.xlabel('False Positive Rate (Fall-Out)', fontsize=16)
    plt.ylabel('True Positive Rate (Recall)', fontsize=16)
    plt.grid(True)

plt.figure(figsize=(8, 6))
plot_roc_curve(fpr, tpr)
plt.plot([4.837e-3, 4.837e-3], [0., 0.4368], "r:")
plt.plot([0.0, 4.837e-3], [0.4368, 0.4368], "r:")
plt.plot([4.837e-3], [0.4368], "ro")
save_fig("roc_curve_plot")
plt.show()
```

- **ROC vs PR Curve**
- Since the ROC curve is so similar to the precision/recall (PR) curve, you may wonder how to decide which one to use
- Prefer the PR curve whenever the positive class is rare or when you care more about the false positives than the false negatives
- Otherwise, use the ROC curve
- Looking at the previous ROC curve (and the ROC AUC score), you
- may think that the classifier is really good
- But this is mostly because there are few positives (5s) compared to the negatives (non-5s)
- In contrast, the PR curve makes it clear that the classifier has room for improvement

19

COMPARING SGD VS RANDOM FOREST CLASSIFIER




```
from sklearn.ensemble import RandomForestClassifier
forest_clf = RandomForestClassifier
        (n_estimators=100, random_state=42)
y_probas_forest = cross_val_predict(forest_clf,
        X_train, y_train_5, cv=3, method="predict_proba")
```

```
 # score = proba of positive class
y_scores_forest = y_probas_forest[:, 1]
fpr_forest, tpr_forest, thresholds_forest =
          roc curve(y train 5,y scores forest)
roc_auc_score(y_train_5, y_scores_forest)
```

```
0.9983436731328145
```

```
y_train_pred_forest = cross_val_predict(forest_clf,
                X_train, y_train_5, cv=3)
precision_score(y_train_5, y_train_pred_forest)
```

```
0.9905083315756169
```

```
recall_score(y_train_5, y_train_pred_forest)
```

```
0.8662608374838591
```

- 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

20

MULTICLASS CLASSIFICATION

- Multiclass classifiers – aka multinomial classifiers are capable of distinguishing between more than two classes
- Algorithms such as SGD classifiers, Random Forest classifiers and Naïve Bayes classifiers can handle multiple classes natively
- Algorithms such as Logistic Regression or Support Vector Machines are strictly binary classifiers
- One-versus-the-rest (OvR) strategy for multiclass classification of MNIST dataset
  - Train 10 binary classifiers – one for each digit ((a 0-detector, a 1-detector, a 2-detector and son on)
  - While classifying an image, get a decision score from each classifier for that image
  - Select the class whose classifier outputs the highest score
- One-versus-One (OvO) strategy for multiclass classification of MNIST dataset
  - Train a binary classifier for every pair of digits (one for distinguishing 0s and 1s, another for distinguishing 0s and 2s, and so on)
  - If there are N classes, in total N x (N – 1) / 2 classifiers required
  - For MNIST, this is training 45 binary classifiers!!
  - A classifier needs to be trained on the part of the training set for the two classes it must distinguish

21

MULTICLASS CLASSIFICATION – SVM CLASSIFIER

```
from sklearn.svm import SVC

svm_clf = SVC(gamma="auto", random_state=42)
svm_clf.fit(X_train[:1000], y_train[:1000]) # y_train, not y_train_5
svm_clf.predict([some_digit])
```

```
array([5], dtype=uint8)
```

```
some_digit_scores = svm_clf.decision_function([some_digit])
some_digit_scores
```

```
array([[ 2.92492871,  7.02307409,  3.93648529,  0.90117363,  5.96945908,
         9.5       ,  1.90718593,  8.02755089, -0.13202708,  4.94216947]])
```

- Scikit-Learn automatically detects whether the problem is of binary classification or multiclass classification
- It automatically runs OvR or OvO
- If SVM classifier is used, Scikit-Learn actually uses OvO strategy
- Trained 45 binary classifiers and got their decision score for images

22

MULTICLASS CLASSIFICATION – SGD CLASSIFIER

```
sgd_clf.fit(X_train, y_train)
sgd_clf.predict([some_digit])
```

```
array([5], dtype=uint8)
```

```
sgd_clf.decision_function([some_digit])
```

```
array([[-15955.22627845, -38080.96296175, -13326.66694897,
          573.52692379, -17680.6846644 ,   2412.53175101,
        -25526.86498156, -12290.15704709,  -7946.05205023,
        -10631.35888549]])
```

```
cross_val_score(sgd_clf, X_train, y_train, cv=3, scoring="accuracy")
```

```
array([0.8489802 , 0.87129356, 0.86988048])
```

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train.astype(np.float64))
cross_val_score(sgd_clf, X_train_scaled, y_train, cv=3, scoring="accuracy")
```

```
array([0.89707059, 0.8960948 , 0.90693604])
```

- The decision_function() method now returns one value per class
- Use cross-validation for evaluating this classifier
- A random classifier would give 10% accuracy
- Simply scaling the inputs increases accuracy above 89%

23

ERROR ANALYSIS USING CONFUSION MATRIX – SGD CLASSIFIER

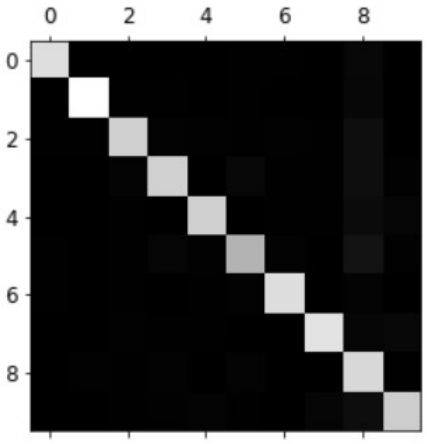
```
y_train_pred = cross_val_predict(sgd_clf, X_train_scaled, y_train, cv=3)
conf_mx = confusion_matrix(y_train, y_train_pred)
conf_mx
```

```
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]])
```

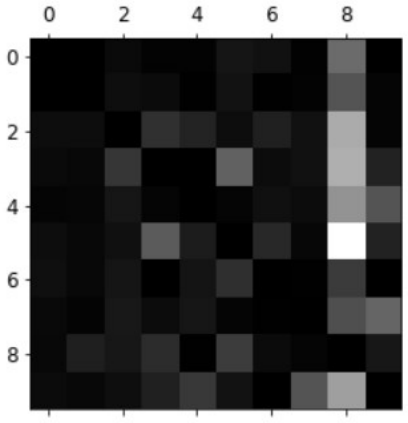
24

ERROR ANALYSIS USING CONFUSION MATRIX – SGD CLASSIFIER

```
plt.matshow(conf_mx, cmap=plt.cm.gray)
save_fig("confusion_matrix_plot",
         tight_layout=False)
plt.show()
```




```
row_sums = conf_mx.sum(axis=1, keepdims=True)
norm_conf_mx = conf_mx / row_sums
np.fill_diagonal(norm_conf_mx, 0)
plt.matshow(norm_conf_mx, cmap=plt.cm.gray)
save_fig("confusion_matrix_errors_plot", tight_layout=False)
plt.show()
```



25

MULTILABEL CLASSIFICATION

- Consider a face recognition classifier which recognizes several people in the same picture
- Say the classifier has been trained to recognize three faces, Alice, Bob, and Charlie
- When the classifier is shown a picture of Alice and Charlie, it should output [1, 0, 1] (meaning “Alice yes, Bob no, Charlie yes”).
- Such a classification system that outputs multiple binary tags is called a *multilabel classification* system
- To evaluate accuracy of multilabel classification, F1 score can be used for each individual label and compute average score
- This assumes that all labels are equally important
- You can use weighted average in case of different level of importance to labels

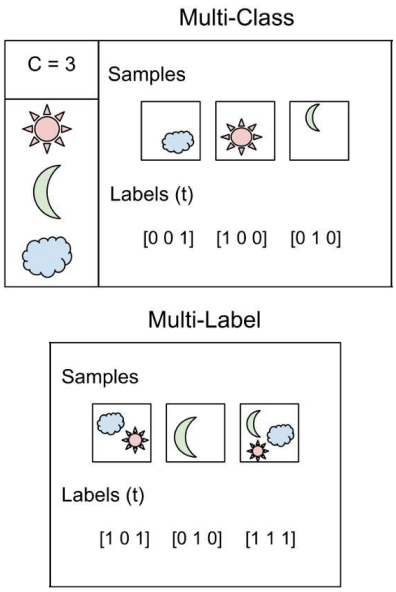
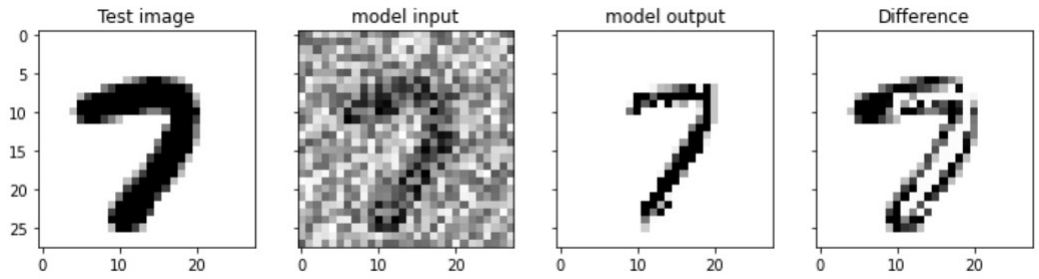


Figure source: https://medium.com/analytics-vidhya/multi-label-classification-using-fastai-a-shallow-dive-into-fastai-data-block-api-54ea57b2c78b

26

MULTIOUTPUT-MULTICLASS CLASSIFICATION



- Multioutput classification is a generalization of multilabel classification
- Each label can be multiclass more than two possible values)
- Consider a system which will take as input a noisy digit image, and will (hopefully) output a clean digit image
- The classifier's output is multilabel (one label per pixel) and each label can have multiple values (pixel intensity 0 to 255)
- The line between classification and regression is sometimes blurry, as in this example
- Predicting pixel intensity is more akin to regression than to classification

27

QUESTIONS?

28