Chapter3. Classification

2.1 MNIST

- 可以使用sklearn中的fetch_openml从www.openml.org网站下载各种数据集
- MNIST包含7000个数据，其中前6000个为训练集，后1000个为测试集，已经进行shuffle了，确保在进行交叉验证时，每一次fold都包含全部数字。

2.2 Training a Binary Classifer

- 使用sklearn中的SGDClassifier。此方法实现了随机梯度下降，可以通过指定损失函数使用哪种模型（SVM、LR, etc.），默认为损失函数hinge，Linear SVM。

2.3 Performance Measures

2.3.1 Measuring Accuracy Using Cross-Validation

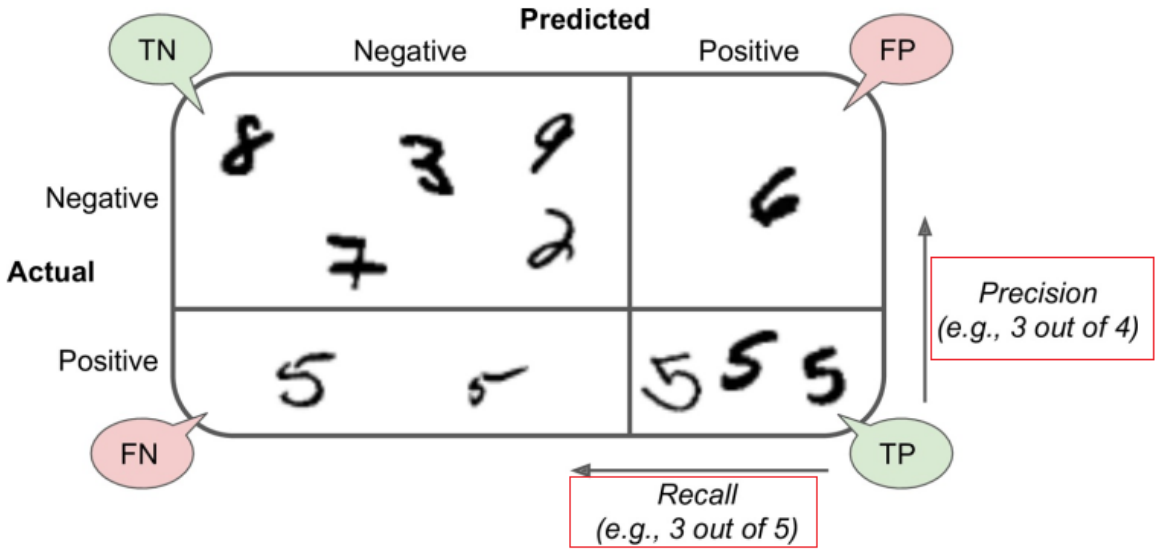
- 使用sklearn中的cross_val_score()函数。
- StratifiedKFold使用了分层抽样，确保每个Fold都包含每个类中一定比例的样本。

2.3.2 Confusion Matrix

- sklearn中的cross_val_predict()实现K折交叉验证，返回的是每个样本的score，该score和threshold决定样本被分为正类还是负类。
- 调用confusion_matrix(groud_true_labels, predict_labels) 方法生成混淆矩阵。

True Negative	**False Positive**
False Negative	**True Positive**

- 完美的分类器只有TN和TP样本，只有对角线上为非零元素。
- Precision = TP / (TP+FP)，指示该分类器找到的正类中，有多少为真正的正类，即针对分类结果而言。
- Recall (也称作Sensitivity或True Positive Rate(TPR)) = TP / (TP+FN)，指示在所有的正类中，分类器找到了多少个，即针对原数据集而言。



2.3.3 Precision and Recall

- F1 score = 2*precision*recall / (precision+recall)，可以使用sklearn的f1_score()函数计算。
- precision升高会导致recall下降，反之亦然。因此要找到precision和recall的平衡点。

2.3.4 Precision/ Recall Trade-off

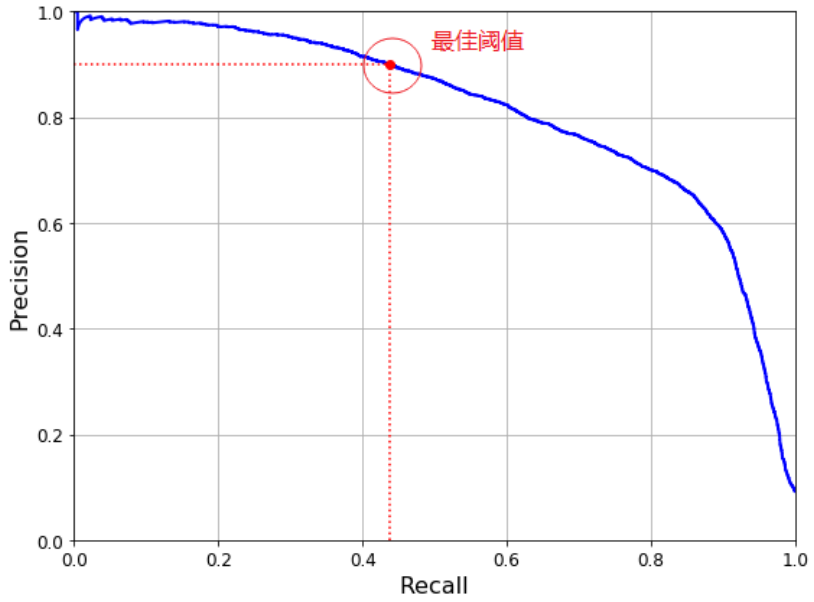
- SGDClassifier是通过threshold和样本的score进行对比来实现分类的。不能直接对SGDClassifier设置分类的threshold。但使用SGDClassifier的decision_function()可以返回每个样本的score，再自定义threshold来对样本实现分类。

```
y_scores = sgd_clf.decision_function([some_digit])
y_scores # output: array([2412.53175101])

threshold = 0
y_some_digit_pred = (y_scores > threshold) # output: array([ True])

threshold = 8000
y_some_digit_pred = (y_scores > threshold)
y_some_digit_pred # output: array([False])
```

- 调用sklearn中的precision_recall_curve方法绘制PR曲线，函数返回threshold和该阈值对应的precisions, recalls。可以选取PR曲线开始下降的点对应的threshold作为最佳阈值。

```python
def plot_precision_vs_recall(precisions, recalls):
    plt.plot(recalls, precisions, "b-", linewidth=2)
    plt.xlabel("Recall", fontsize=16)
    plt.ylabel("Precision", fontsize=16)
    plt.axis([0, 1, 0, 1])
    plt.grid(True)

plt.figure(figsize=(8, 6))
plot_precision_vs_recall(precisions, recalls)
plt.plot([0.4368, 0.4368], [0., 0.9], "r:")
plt.plot([0.0, 0.4368], [0.9, 0.9], "r:")
plt.plot([0.4368], [0.9], "ro")
save_fig("precision_vs_recall_plot")
plt.show()
```

```
Saving figure precision_vs_recall_plot
```




```python
threshold_90_precision = thresholds[np.argmax(precisions >= 0.90)] # 取得最佳threshold值
```

2.4 The ROC Curve

- 横轴为FPR (false positive rate = 1 - true negative rate)，纵轴为TPR (true positive rate, 即recall)。
- sklearn中的roc_curve可以绘制出ROC曲线。函数返回fpr, tpr, threshold
- 选择PR曲线还是ROC曲线?
    - 当正类样本很少或相对false negtives，更关心false positives时，使用PR曲线。反之选择ROC曲线。
- ROC曲线的AUC面积越接近1，则模型效果越好。可以通过sklearn中的roc_auc_score函数求得。
    - 书中使用了Random Forest和SGDClassifier比较。RandomForestClassifier的predict_proba()方法返回一个数组。一行代表一个样本，一列代表该样本属于该类的概率。

2.5 Multiclass Classification

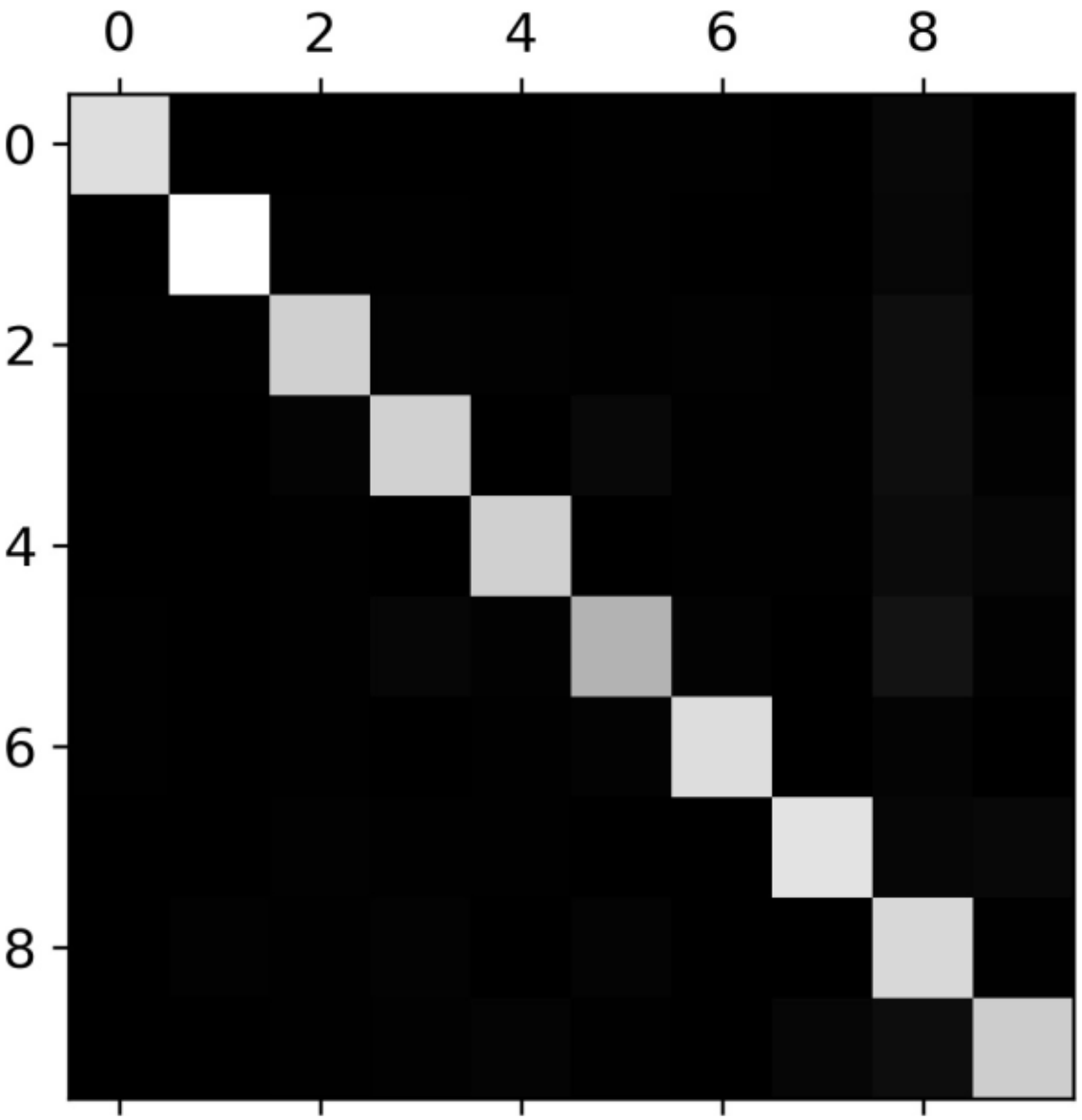
- SGD, Random Forest, Naive Bayes可以处理多分类任务。
- LR, SVM一般只处理二分类任务，但是有些策略可以让让他们实现多分类：
    - one-versus-the-rest（OVR, one-versus-all）策略，需要实现10个类的分类时，可以训练10个分类器，每个分类器只对一个类进行分类
    - one-versus-one（OVO）策略，每个分类器对一对类进行分类（比如一个分类器对0和2进行分类，另一个对1和2分类）。如果有N个类，则总共需要 N*(N-1)/2 个分类器。此策略的优点是每个分类器的训练只需要一部分数据集。
- 有些分类器在大数据量时扩展性不高，此时OVO策略比较有效。
- 在sklearn中，当使用二分类算法实现多分类任务时，会自动选择使用OVR或者OVO策略。
- 如果想强制sklearn使用OVO或OVR策略，可以使用OneVsOneClassifier或OneVsRestClassifier。

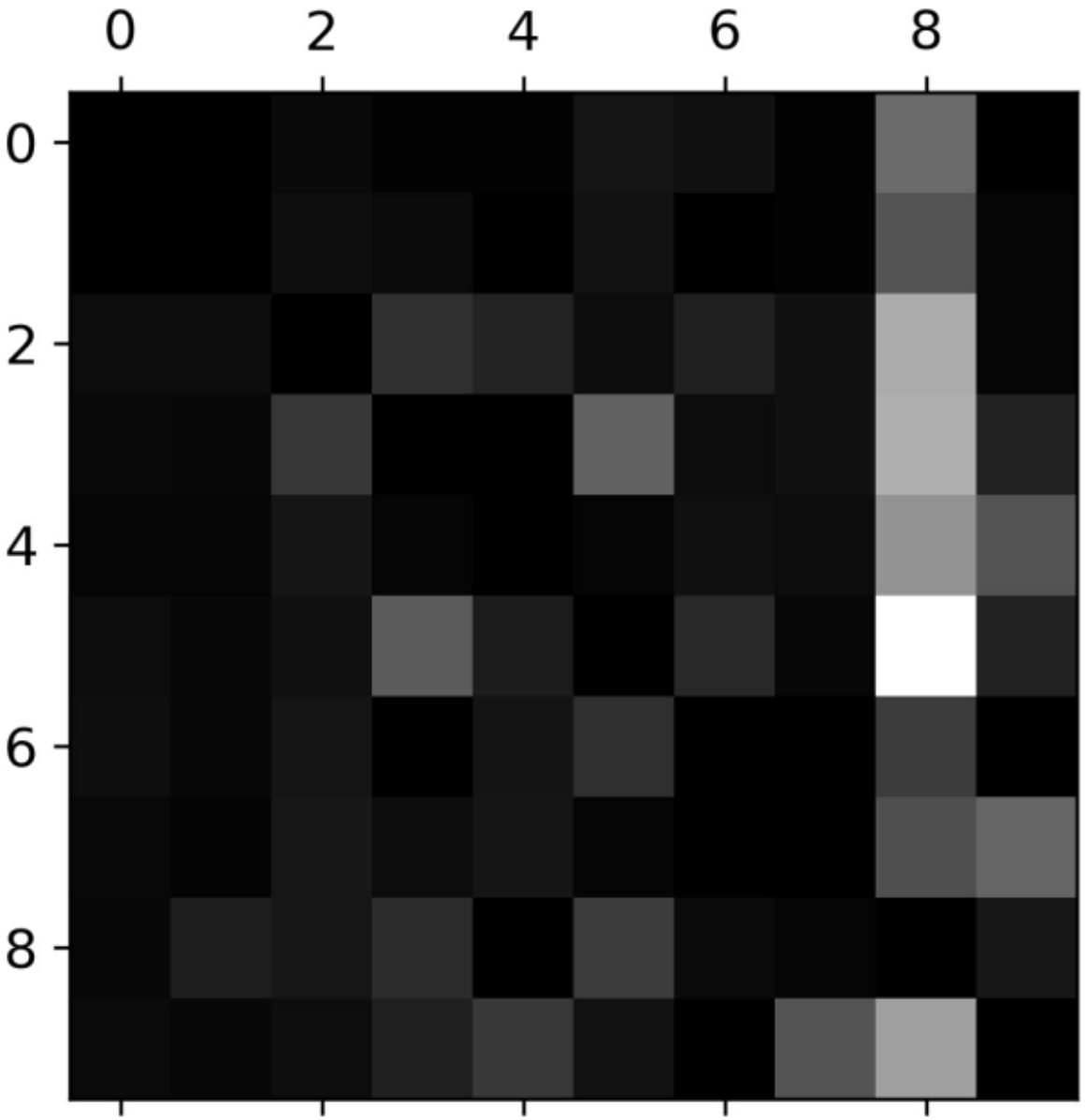
2.6 Error Analysis

- 使用GridSearchCV类实现网格搜索，对超参数进行调优。
- 先调用cross_val_predict()方法获取每个类的正确/ 错误分类情况，调用confusion_matrix()使用混淆矩阵进行可视化分析：
    - 以下图一为例，可以看到对角线上的区域是高亮的，代表大部分样本都得到了正确分类。但是可以看到分类5和分类3下的区域比其他对角线上的区域要暗一点，代表5和3进场被误分类了。
    - 再对误差进行详细的分析。在图二可以看到，第八列的所有区域都比较亮（除了(8,8)对应的区域），代表大部分样本都被误分类到8。针对此情况的方法是可以添加更多看起来像8的样本，让分类器更好的学习如何将真实的8与其他数字分离出来。
    - 还可以将误分类的样本拿出来（比如3和5经常被分类），作处理

```
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)
plt.show() #得到图2
```





2.7 Multilabel Classification

- 标签中包含多个变量，比如数字识别中，Y=[是否为大于7的数，是否为偶数]

```
from sklearn.neighbors import KNeighborsClassifier

y_train_large = (y_train >= 7)
y_train_odd = (y_train % 2 == 1)
y_multilabel = np.c_[y_train_large, y_train_odd]

knn_clf = KNeighborsClassifier()
knn_clf.fit(X_train, y_multilabel)
```

2.8 Multioutput Classification

- 是multilabel classification的更广泛的形式，即标签中的label值可能有多个。
- 书上举了图片去噪声的例子，这样原来的label就变成了每个像素点的像素值。