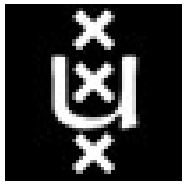
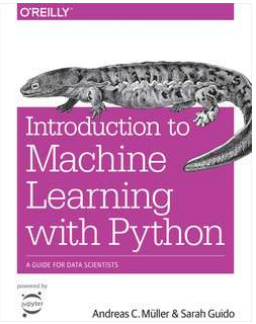


Applied Machine Learning
Classification Evaluation Metrics

BSc course Informatiekunde 2026
https://staff.fnwi.uva.nl/a.visser/education/AML

Arnoud Visser
Intelligent Robotics Lab & Computer Vision Lab
Informatics Institute
Universiteit van Amsterdam
A.Visser@uva.nl

Illustrations courtesy of Maarten Marx, Sarah Guido, Yolanda Hagar, and many others.



Section 5.3

Realistic dataset

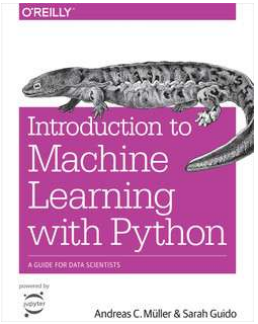
- Iris dataset



Iris Setosa, Versicolor and Virginica

```
X_train, X_test, y_train, y_test = train_test_split(
    iris_dataset['data'], iris_dataset['target'], random_state=0)

knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)

print("Test set score: {:.2f}".format(knn.score(X_test, y_test)))
```

```
Test set score: 0.97
```



Realistic dataset

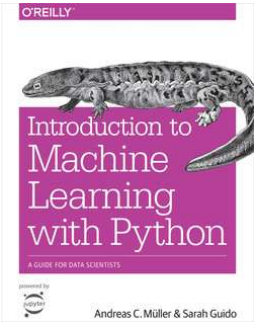
- Iris dataset



Iris Setosa, Versicolor and Virginica

```
…
knn.fit(X_train, y_train)

y_test_predictions = knn.predict(X_test[0:9])
print('knn predicts: ' + str(iris.target_names[y_test_predictions])
```

```
knn predicts: ['versicolor' 'versicolor' 'versicolor' 'virginica' ,'setosa' 'virginica' 'versicolor' 'setosa'
'versicolor' 'versicolor']
```



Section 1.7

Realistic dataset

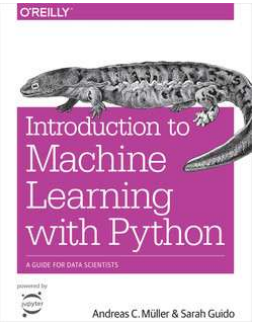
- Iris dataset



Iris Setosa, Versicolor and Virginica

```
…
knn.fit(X_train, y_train)

y_test_predictions = knn.predict(X_test[0:9])

errors = (y_test_predictions != y_test[0:9])
error_est = sum(errors)/errors.size

print('The error rate estimate is: {:.2f}'.format(error_est) + '\n’
print( 'The accuracy is: {:.2f}'.format(1-error_est))
```

```
The error rate estimate is: 0.10
The accuracy is: 0.90
```



Section 1.7

Realistic dataset

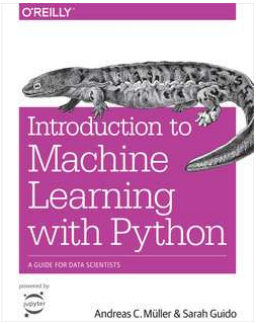
- Iris dataset



Iris Setosa, Versicolor and Virginica

```
from sklearn.metrics import accuracy_score
…

knn.fit(X_train, y_train)

y_test_predictions = knn.predict(X_test[0:9])

print('The accuracy is {:.2f}'.format(accuracy_score(y_test[0:9], ,y_test_predictions)))
```

```
The accuracy is: 0.90
```



Section 1.7

Realistic dataset

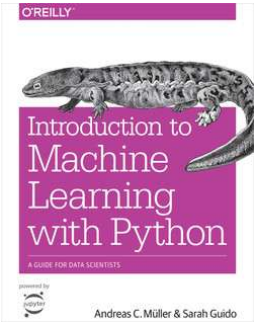
- Iris dataset



Iris Setosa, Versicolor and Virginica

```
…

knn.fit(X_train, y_train)

method_score = knn.score(X_test[0;9], y_test[0:9])

print('The accuracy is {:.2f}'.format(method_score )))
```

```
The accuracy is: 0.90
```



Section 1.7

What happens with more classes?

- Model = EstimatorObject()
- Model.fit(dataset.data, dataset.target)
  - dataset.data = dataset ← X
  - dataset.target = labels ← y
- Model.predict(dataset.data)

Class

	class
0	unacc
1	unacc
2	unacc
3	unacc
4	unacc

vector ↓

Matrix ↓

$$y = f(X)$$

↓ output

↑ input

Feature

	buying	maint	doors	persons	lug_boot	safety
0	vhigh	vhigh	2	2	small	low
1	vhigh	vhigh	2	2	small	med
2	vhigh	vhigh	2	2	small	high
3	vhigh	vhigh	2	2	med	low
4	vhigh	vhigh	2	2	med	med

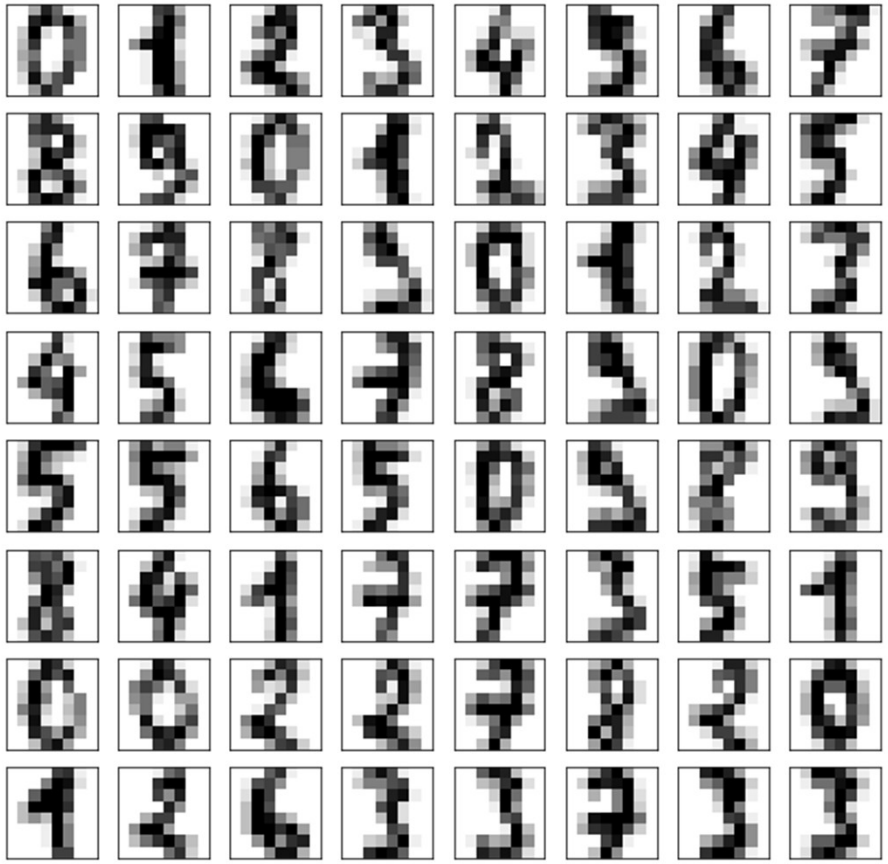
Instance

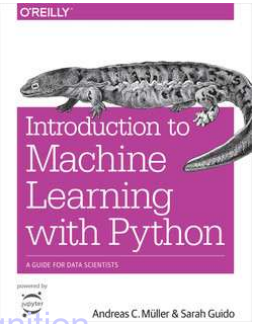
Slide idea from Sharon Guido

Realistic dataset

- Digits dataset

```
from sklearn.datasets import load_digits
digits = load_digits()
```





Kaynak, C. (1995) Methods of combining multiple classifiers and their applications to handwritten digit recognition
Master thesis, .Bogazici University

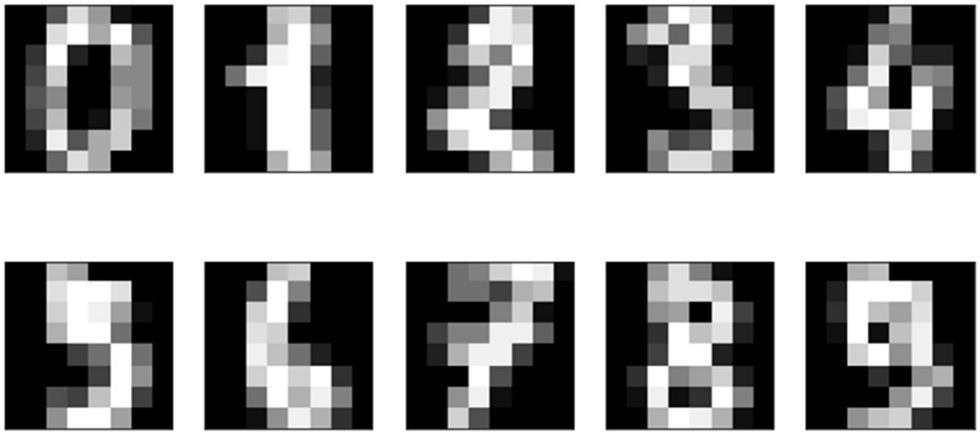
Section 3.4.3

Realistic dataset

- Digits dataset

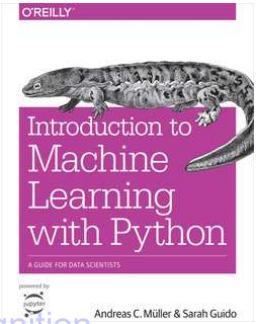
64 pixels, 64 features

```
from sklearn.datasets import load_digits

digits = load_digits()
y = digits.target == 9

X_train, X_test, y_train, y_test = train_test_split(
    digits.data, y, random_state=0)
```



```
import pandas as pd

df = pd.DataFrame(digits.data,
columns=digits.feature_names)
df['target'] = digits.target
df.head()
```

pixel_7_3	pixel_7_4	pixel_7_5	pixel_7_6	pixel_7_7	target
13.0	10.0	0.0	0.0	0.0	0
11.0	16.0	10.0	0.0	0.0	1
3.0	11.0	16.0	9.0	0.0	2
13.0	13.0	9.0	0.0	0.0	3
2.0	16.0	4.0	0.0	0.0	4



Kaynak, C. (1995) Methods of combining multiple classifiers and their applications to handwritten digit recognition Master thesis, .Bogazici University

Section 3.4.3

Binary Classification

- Digits dataset

[9] versus [0,1,2,3,4,5,6,7,8]

```
from sklearn.datasets import load_digits

digits = load_digits()
y = digits.target == 9

X_train, X_test, y_train, y_test = train_test_split(
    digits.data, y, random_state=0)
```

```
from sklearn.dummy import DummyClassifier

dummy_majority = DummyClassifier(strategy='most_frequent')
dummy_majority.fit(X_train, y_train)
accuracy = dummy_majority.score(X_test, y_test)))
print("Test score: {:.2f}".format(accuracy))
```

```
Test score: 0.90
```

```
from sklearn.metrics import precision_recall_fscore_support

pred_most_frequent = dummy_majority.predict(X_test)

all_metrics = precision_recall_fscore_support(
y_test, pred_most_frequent)



print("support: ", all_metrics[3])
```

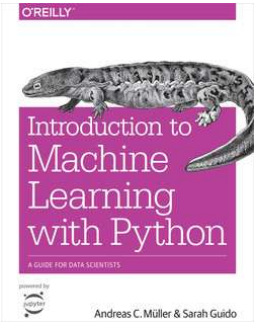
```
support:  [403  47]
```

Is there a [9]? -> predict() = False

All 450 X_test samples!

All 403 correct not-nine!

All 47 wrong a-nine!



Section 5.3.2

Binary Classification

- Digits dataset

[9] versus [0,1,2,3,4,5,6,7,8]

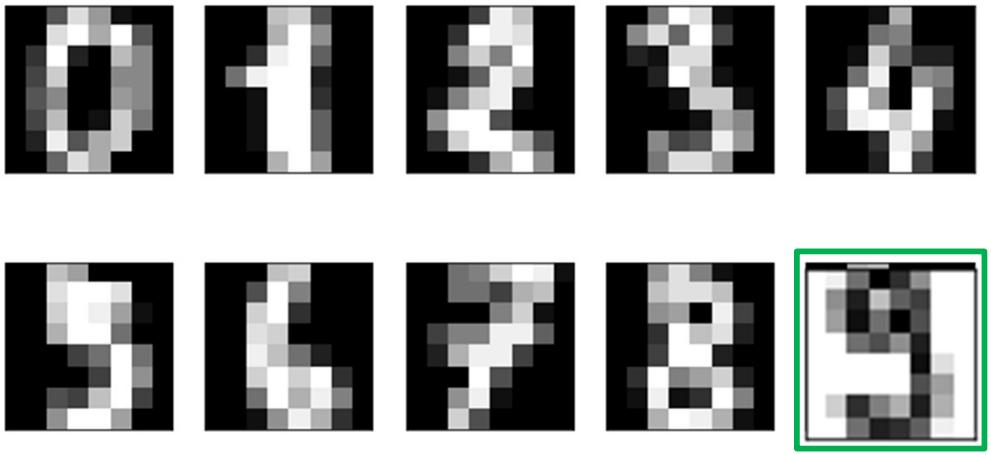
```
dummy_majority = DummyClassifier(strategy='most_frequent')
```

```
from sklearn.metrics import precision_recall_fscore_support

pred_most_frequent = dummy_majority.predict(X_test)

all_metrics = precision_recall_fscore_support(
y_test, pred_most_frequent)


print("support: ", all_metrics[3])
print("recall: ", all_metrics[1])
```

Is there a [9]? -> predict() = False

```
Test score: 0.90
```

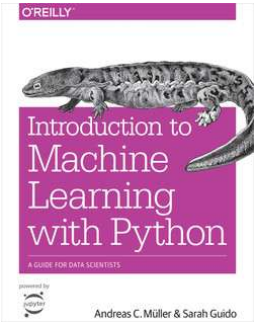
```
support:  [403  47]
recall:     [1.0  0.0]
```



All 450 X_test samples!

All 403 correct not-nine!

All 47 wrong a-nine!



Section 5.3.2

Binary Classification

- Digits dataset

[9] versus [0,1,2,3,4,5,6,7,8]

```
knn = KNeighborsClassifier(n_neighbors=5)
```

Is there a [9]? -> predict() = False

```
from sklearn.metrics import precision_recall_fscore_support

pred_most_frequent = dummy_majority.predict(X_test)

all_metrics = precision_recall_fscore_support(
y_test, pred_most_frequent)


print("support: ", all_metrics[3])
print("recall: ", all_metrics[1])
```

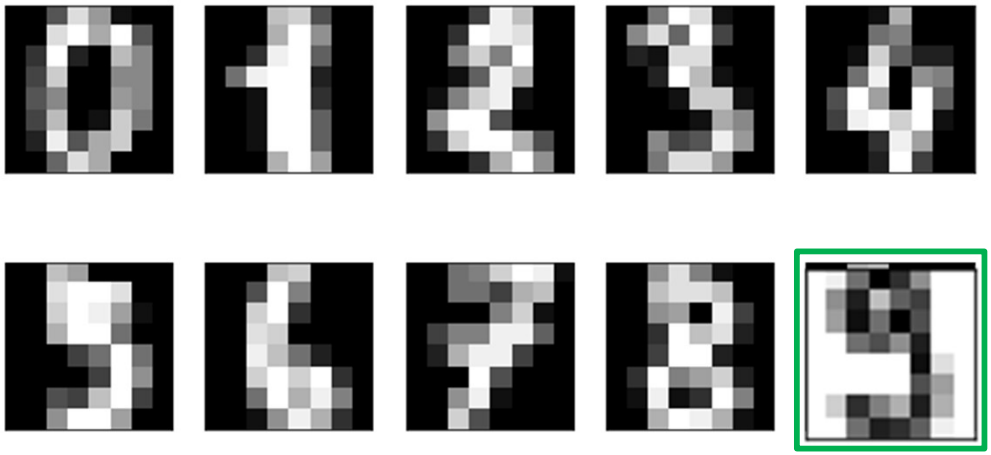
Test score: 0.996

448/450 = 0.9956

```
support:  [403  47]
recall:   [0.998  0.979]
```

~~402/403~~ = 0.9975

~~46/47~~ = 0.9787

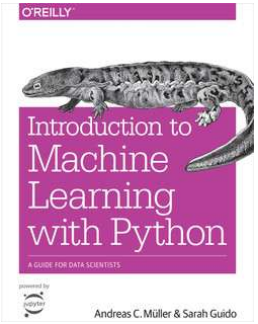


All 450 X_test samples!

402 correct not-nine!

46 correct a-nine!

448 correct!



Section 5.3.2

Binary Classification

- Digits dataset

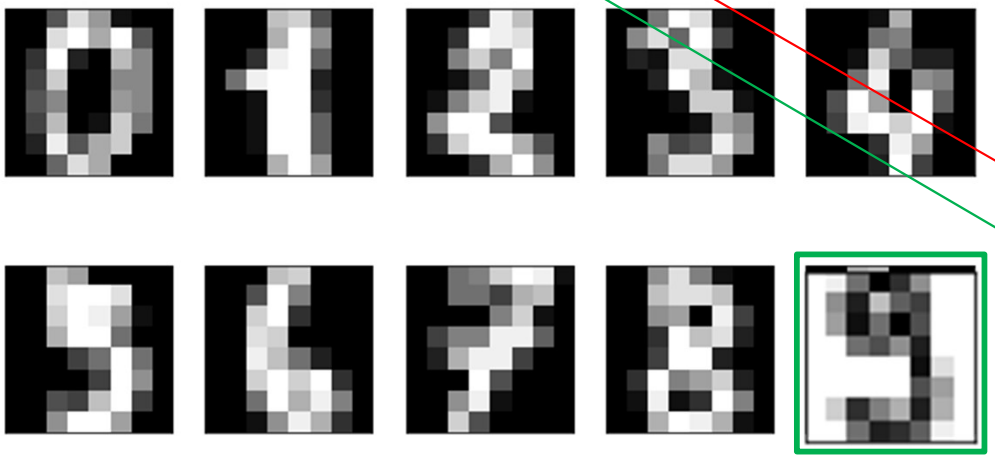
[9] versus [0,1,2,3,4,5,6,7,8]

```
knn = KNeighborsClassifier(n_neighbors=5)
```

Is there a [9]? -> predict() = False

```
from sklearn.metrics import confusion_matrix

tn, fp, fn, tp = confusion_matrix(y_test, pred_knn).ravel()

print("support: ", [tn+fp,fn+tp])
print("True Negatives: ", tn)
print("True Positives: ", tp)
```

```
support:  [403  47]
True Negatives:  402
True Positives:     46
```



```
support:  [403  47]
recall:     [0.998  0.979]
```

```
Test score: 0.996
```

All 450 X_test samples!

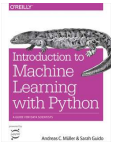
402 correct not-nine! 402/403 = 0.9975

46 correct a-nine! 46/47 = 0.9787

448 correct! 448/450 = 0.9956

Binary Classification

- Digits dataset



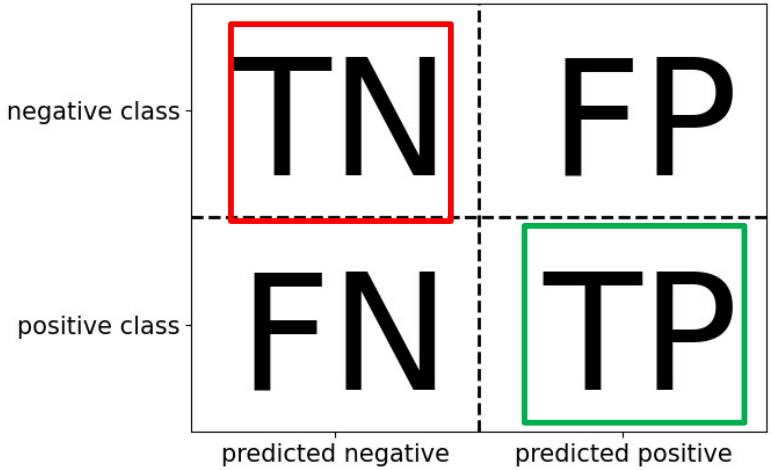
Page 288

[9] versus [0,1,2,3,4,5,6,7,8]

```
knn = KNeighborsClassifier(n_neighbors=5)
```

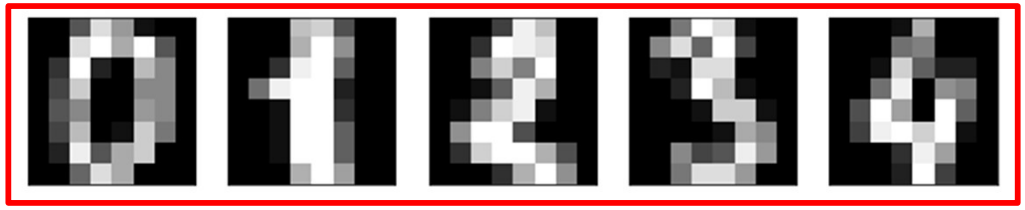
Is there a [9]? -> predict() = False

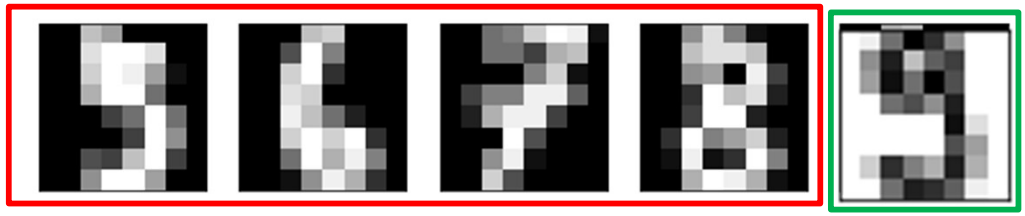
```
mglearn.plots.plot_binary_confusion_matrix()
```




```
from sklearn.metrics import confusion_matrix

tn, fp, fn, tp = confusion_matrix(y_test, pred_knn).ravel()

print("support: ", [tn+fp,fn+tp])
print("True Negatives: ", tn)
print("True Positives: ", tp)
```

```
support:  [403  47]
True Negatives:  402
True Positives:    46
```





All 450 X_test samples!

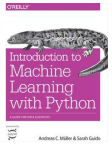
402 correct not-nine! 402/403 = 0.9975

46 correct a-nine! 46/47 = 0.9787

448 correct! 448/450 = 0.9956

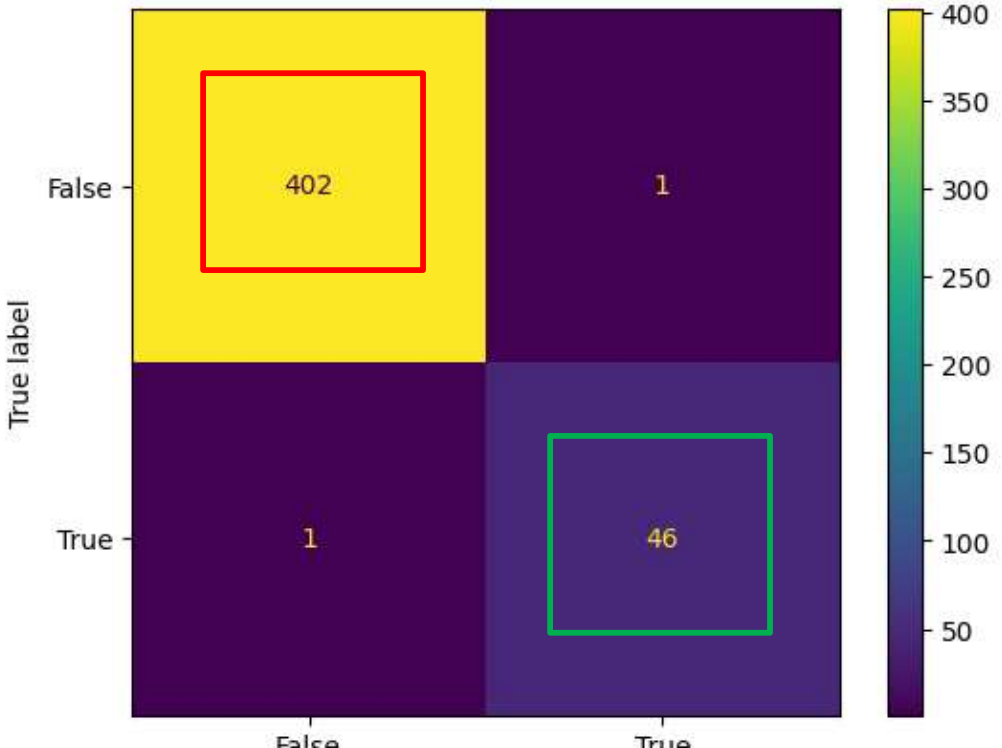
Binary Classification

- Digits dataset



Page 288

```
from sklearn.metrics import ConfusionMatrixDisplay

cm = confusion_matrix(y_test, pred_knn,
labels=knn.classes_)

disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=knn.classes_)
```



[9] versus [0,1,2,3,4,5,6,7,8]

```
knn = KNeighborsClassifier(n_neighbors=5)
```

Is there a [9]? -> predict() = False

```
from sklearn.metrics import confusion_matrix

tn, fp, fn, tp = confusion_matrix(y_test, pred_knn).ravel()

print("support: ", [tn+fp,fn+tp])
print("True Negatives: ", tn)
print("True Positives: ", tp)
```

```
support:  [403  47]
True Negatives:  402
True Positives:    46
```

All 450 X_test samples!

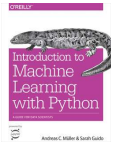
402 correct not-nine! 402/403 = 0.9975

46 correct a-nine! 46/47 = 0.9787

448 correct! 448/450 = 0.9956

Binary Classification

- Digits dataset

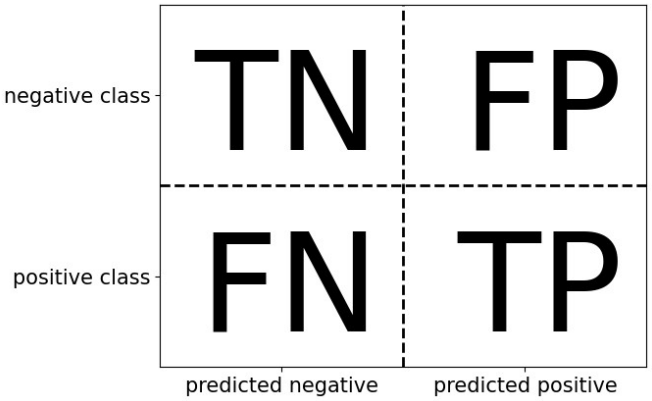


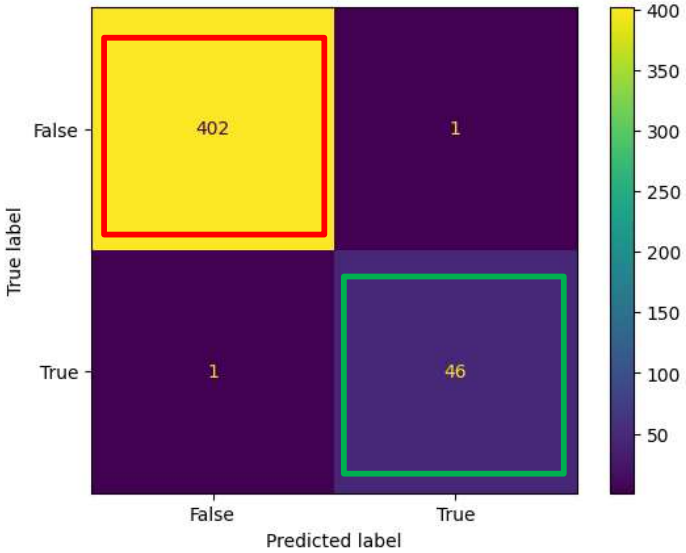
Page 289

[9] versus [0,1,2,3,4,5,6,7,8]

```
knn = KNeighborsClassifier(n_neighbors=5)
```

Is there a [9]? -> predict() = False





$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

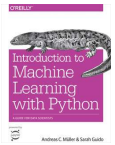
448 correct! 448/450 = 0.9956

$$\text{Recall} = \frac{TP}{TP + FN}$$

46 correct! 46/47 = 0.9787

Binary Classification

- Digits dataset



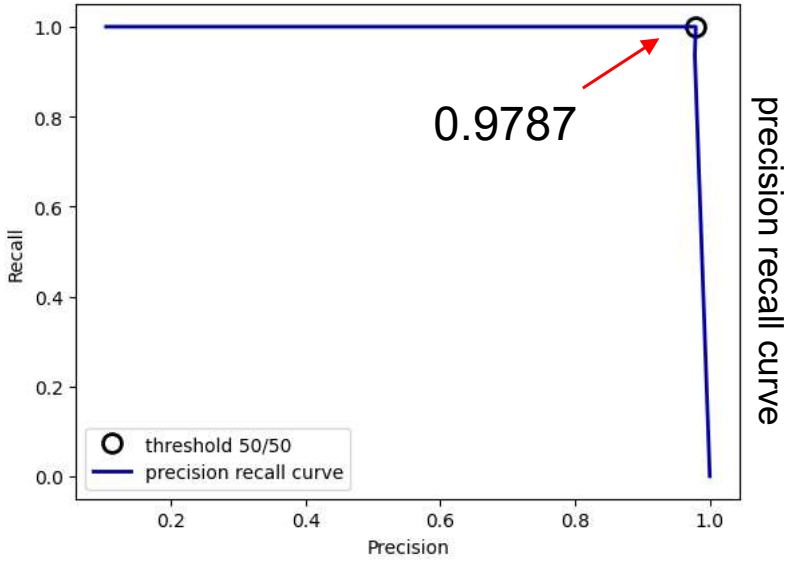
Page 298

[9] versus [0,1,2,3,4,5,6,7,8]

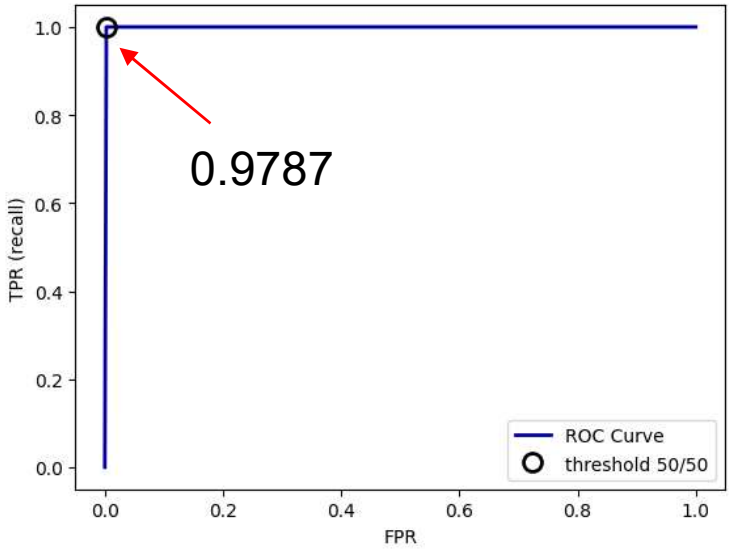
```
knn = KNeighborsClassifier(n_neighbors=5)
```

```
precision, recall, thresholds = precision_recall_curve
                    (y_test, knn.predict_proba(X_test)[:, 1])
```

```
fpr, tpr, thresholds = roc_curve
      (y_test, knn.predict_proba(X_test)[:, 1])
```



precision recall curve



receiver operating characteristics curve

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

46/47 = 0.9787

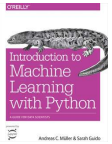
$$\text{FPR} = \frac{FP}{FP + TN}$$

46/47 = 0.9787

1/403 = 0.00248

Binary Classification

- Digits dataset



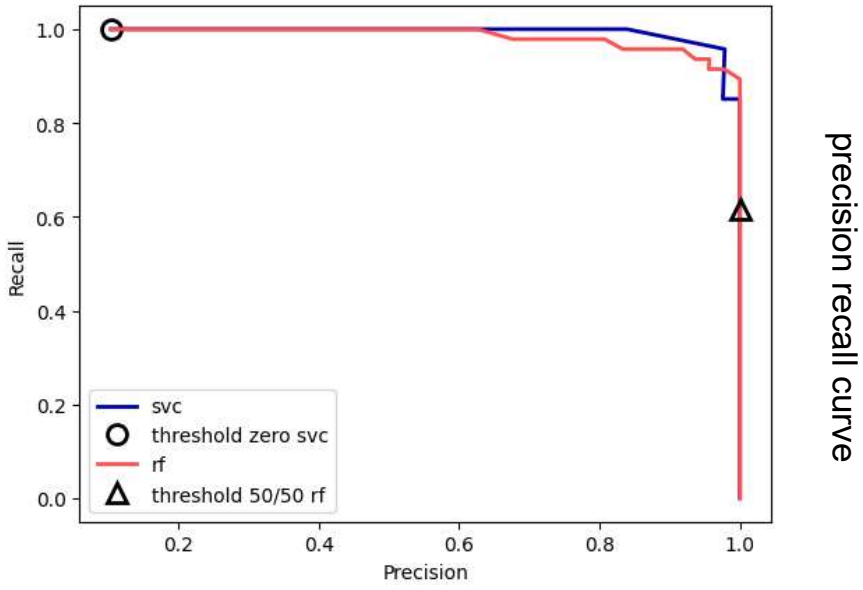
Page 298

[9] versus [0,1,2,3,4,5,6,7,8]

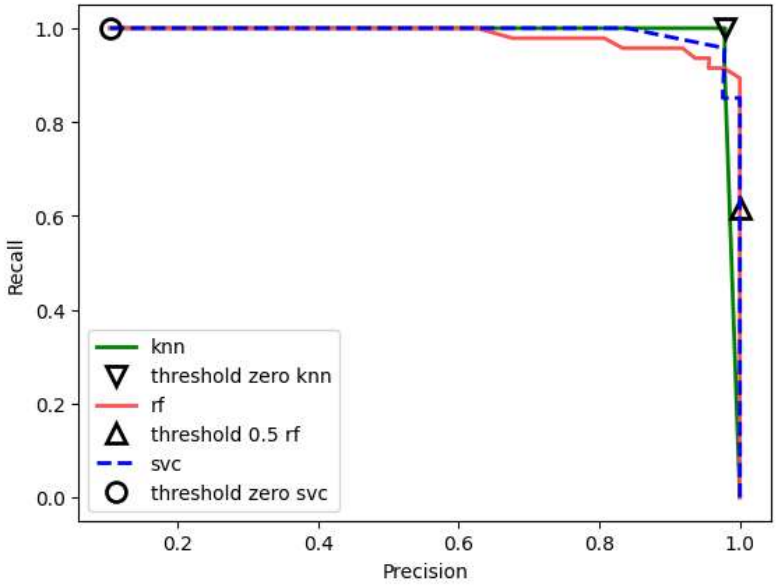
```
knn = KNeighborsClassifier(n_neighbors=5)
```

```
svc = SVC(gamma=.05)
```

```
rf = RandomForestClassifier(n_estimators=100,
random_state=0, max_features=2)
```



precision recall curve



precision recall curve

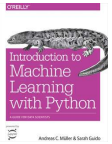
```
from sklearn.metrics import f1_score

print("f1 score knn: {:.2f}".format(
    f1_score(y_test, pred_knn)))
print("f1 score rf: {:.2f}".format(
    f1_score(y_test, rf.predict(X_test))))
print("f1 score svc: {:.2f}".format(
    f1_score(y_test, svc.predict(X_test))))
```

$$f_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

```
f1 score knn: 0.98
f1 score rf:     0.75
f1 score svc: 0.00
```

Binary Classification

- Digits dataset



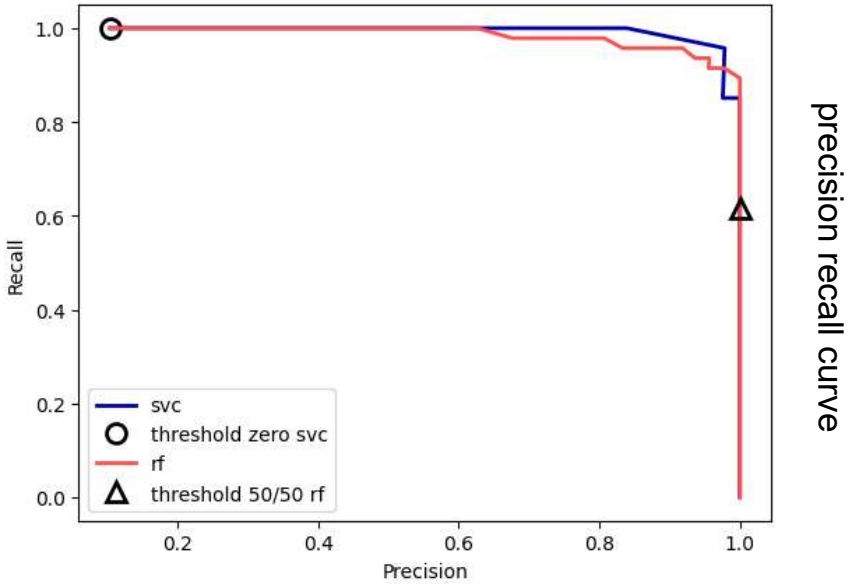
Page 298

[9] versus [0,1,2,3,4,5,6,7,8]

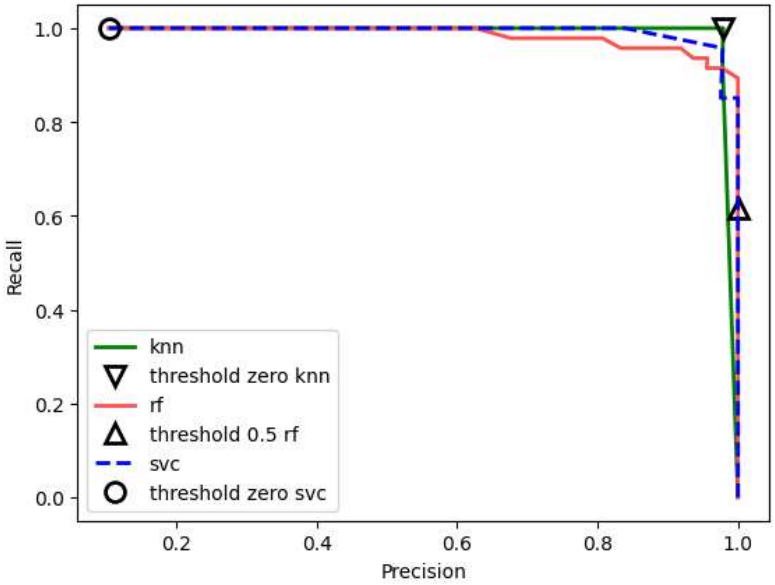
```
knn = KNeighborsClassifier(n_neighbors=5)
```

```
svc = SVC(gamma=.05)
```

```
rf = RandomForestClassifier(n_estimators=100,
random_state=0, max_features=2)
```



precision recall curve



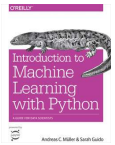
precision recall curve

```
from sklearn.metrics import average_precision_score
```

```
average precision score knn: 0.98
average precision score rf:    0.98
average precision score svc: 0.99
```

Binary Classification

- Digits dataset

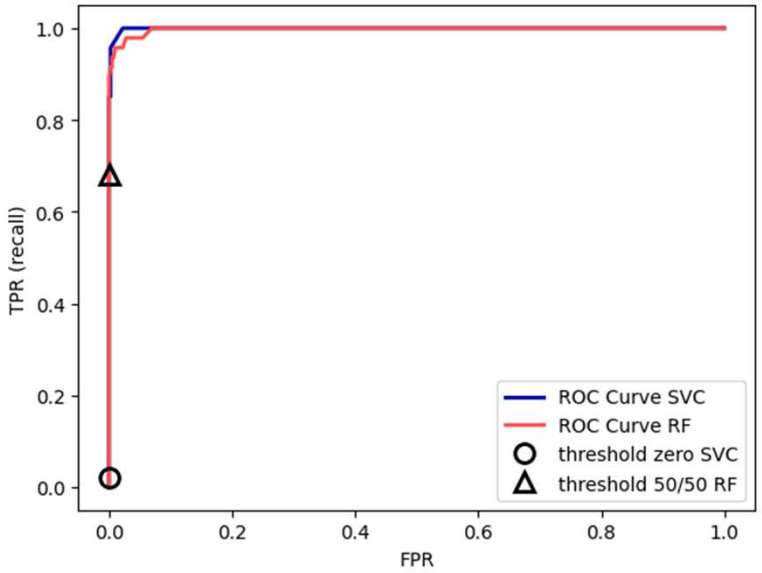


Page '301

[9] versus [0,1,2,3,4,5,6,7,8]

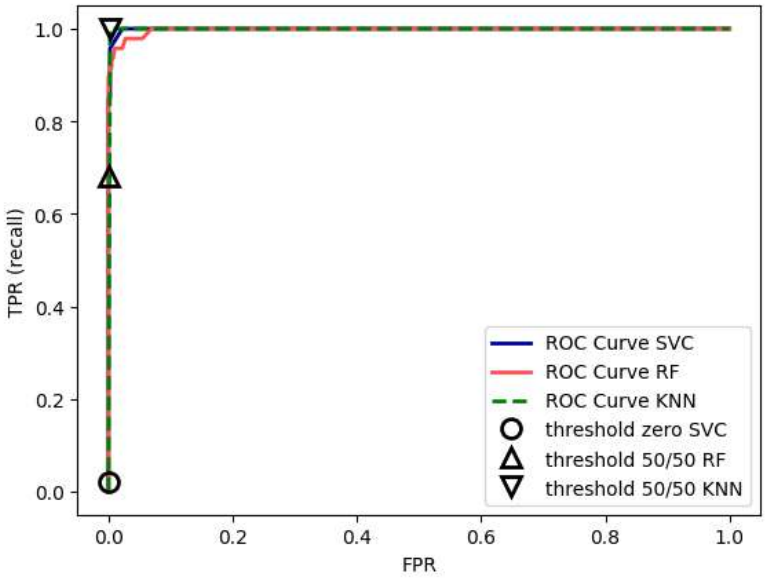
```
knn = KNeighborsClassifier(n_neighbors=5)
```

```
svc = SVC(gamma=.05)
```



receiver operating characteristics curve

```
rf = RandomForestClassifier(n_estimators=100,
random_state=0, max_features=2)
```



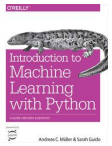
receiver operating characteristics curve

```
from sklearn.metrics import roc_auc_score
```

```
AUC score knn: 1.00
AUC score rf:    1.00
AUC score svc: 1.00
```

Binary Classification

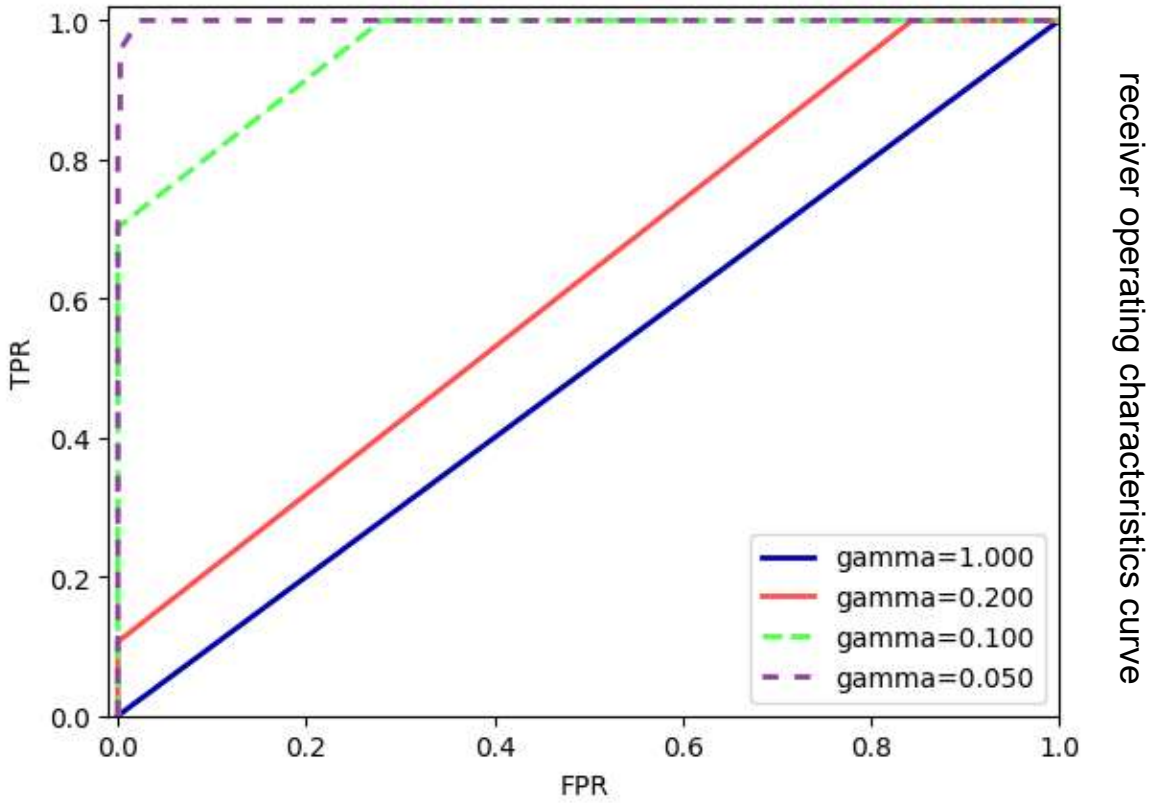
- Digits dataset



Page '302

[9] versus [0,1,2,3,4,5,6,7,8]

```
svc = SVC(gamma=gamma)
```

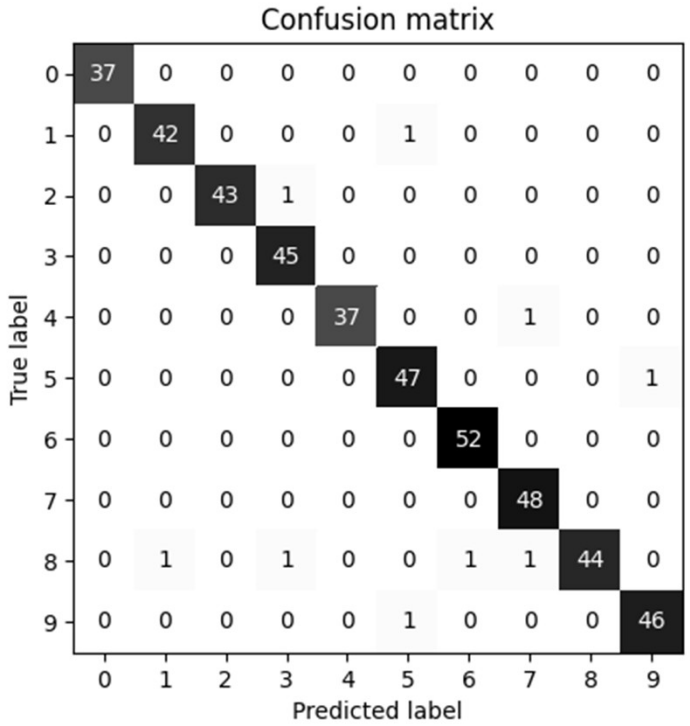


receiver operating characteristics curve

gamma = 1.00 accuracy = 0.90 AUC = 0.50
gamma = 0.20 accuracy = 0.90 AUC = 0.62
gamma = 0.10 accuracy = 0.90 AUC = 0.96
gamma = 0.05 accuracy = 0.90 AUC = 1.00

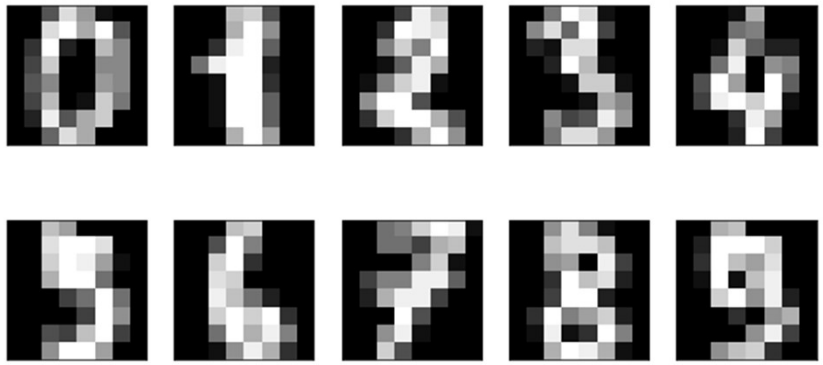
Multi-Class Classification

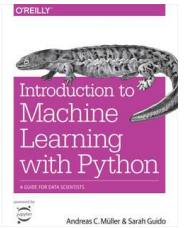
[0,1,2,3,4,5,6,7,8,9] versus [0,1,2,3,4,5,6,7,8,9]




```
from sklearn.metrics import accuracy_score

X_train, X_test, y_train, y_test = train_test_split(
    digits.data, digits.target, random_state=0)

knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
pred = knn.predict(X_test)

print("Accuracy: {:.3f}".format(accuracy_score(y_test, pred)))
```

```
Accuracy: 0.980
```

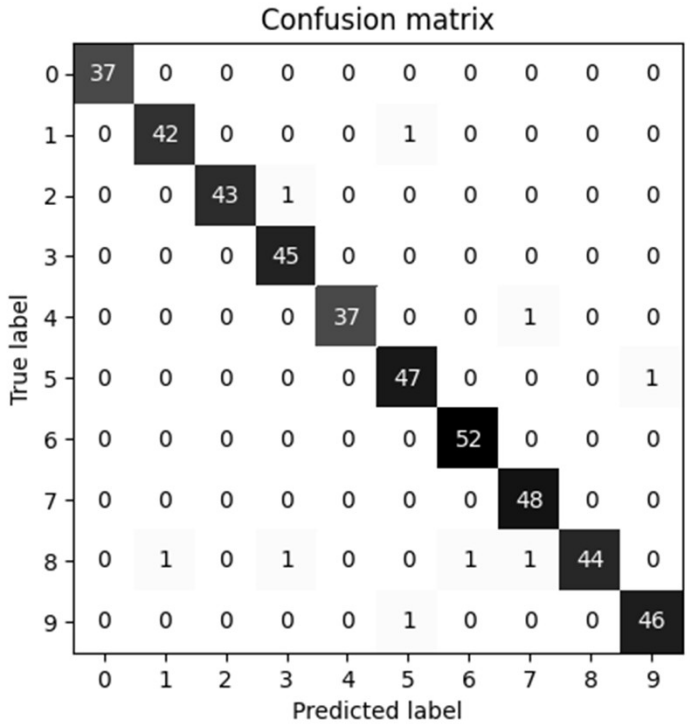


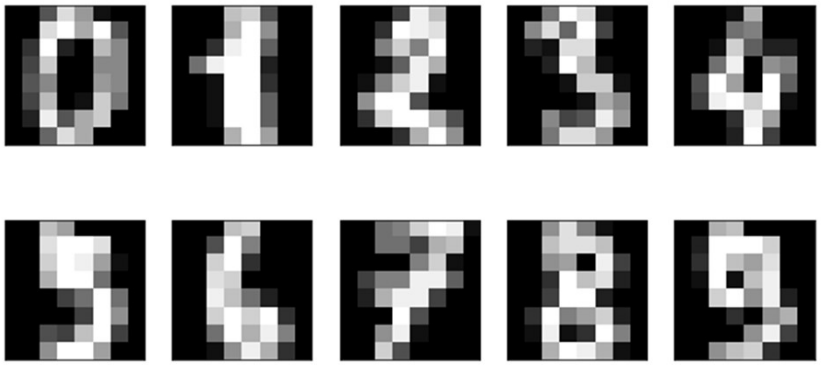


Section 5.3.3

Multi-Class Classification

[0,1,2,3,4,5,6,7,8,9] versus [0,1,2,3,4,5,6,7,8,9]



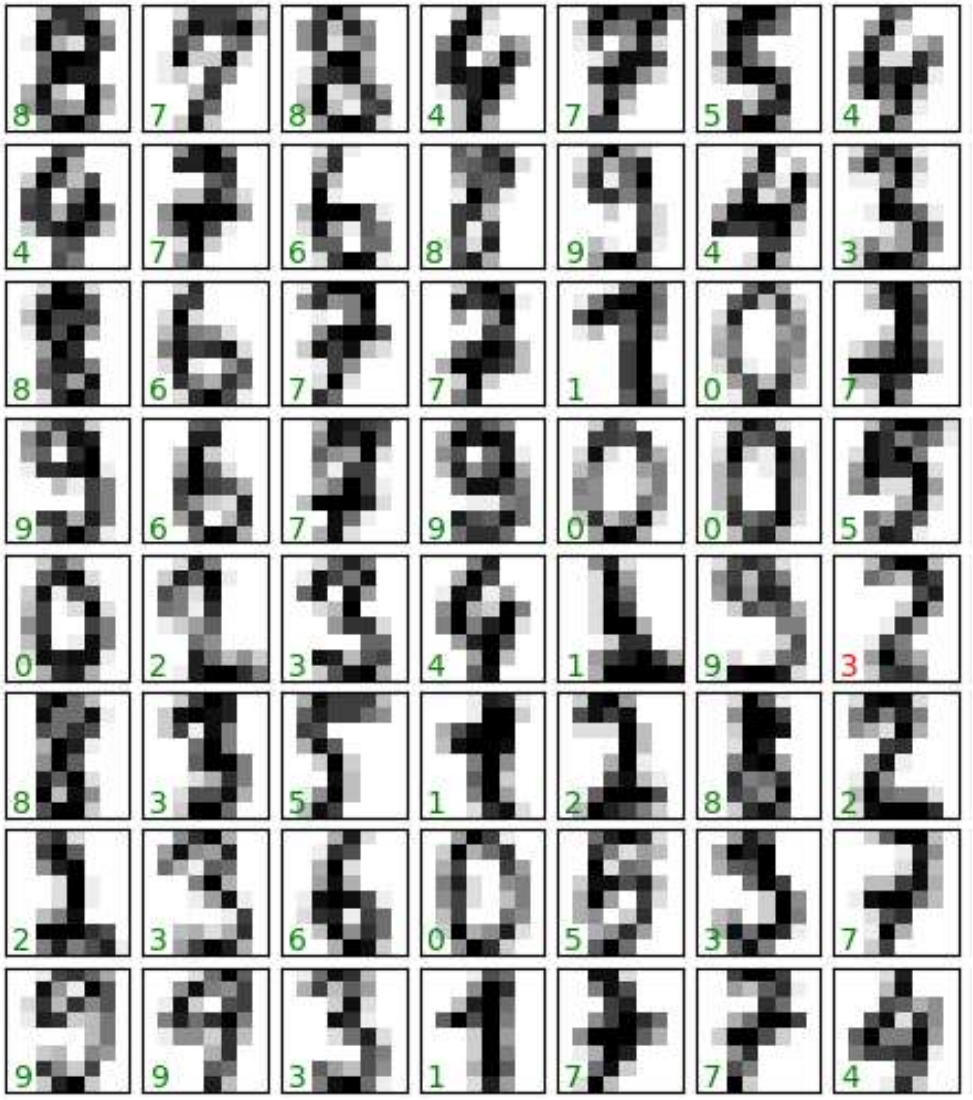


```
from sklearn.metrics import classification_report
print(classification_report(y_test, pred))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	37
1	0.98	0.98	0.98	43
2	1.00	0.98	0.99	44
3	0.96	1.00	0.98	45
4	1.00	0.97	0.99	38
5	0.96	0.98	0.97	48
6	0.98	1.00	0.99	52
7	0.96	1.00	0.98	48
8	1.00	0.92	0.96	48
9	0.98	0.98	0.98	47
accuracy			0.98	450
macro avg	0.98	0.98	0.98	450
weighted avg	0.98	0.98	0.98	450

Multi-Class Classification

[0,1,2,3,4,5,6,7,8,9] versus [0,1,2,3,4,5,6,7,8,9]

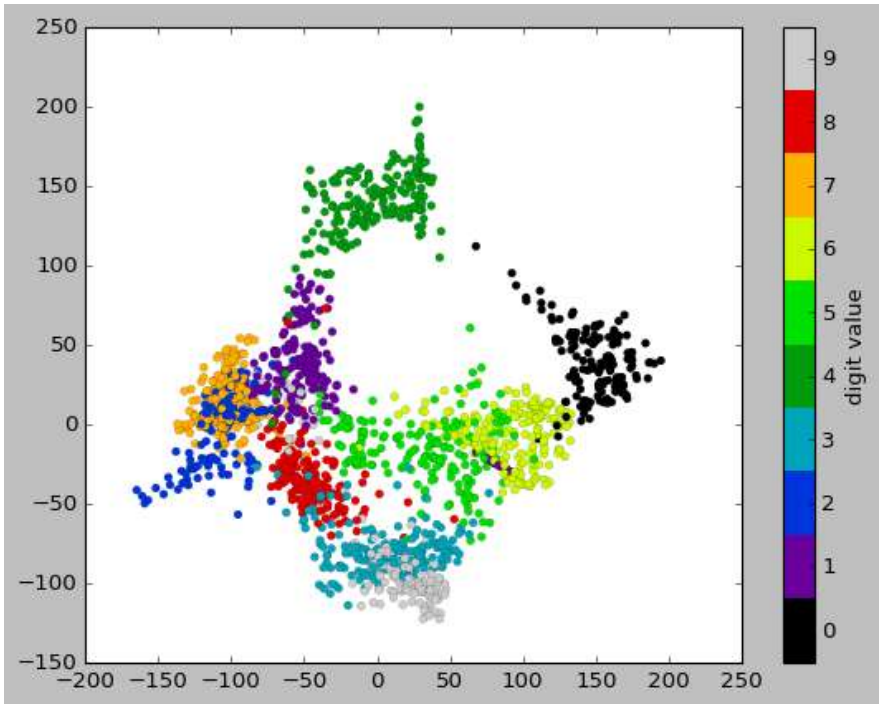


```
fig, axes = plt.subplots(10, 10, figsize=(8, 8),
                         subplot_kw={'xticks':[], 'yticks':[]},
                         gridspec_kw=dict(hspace=0.1, wspace=0.1))

test_images = X_test.reshape(-1, 8, 8)

for i, ax in enumerate(axes.flat):
    ax.imshow(test_images[i], cmap='binary',
interpolation='nearest')
    ax.text(0.05, 0.05, str(pred[i]),
            transform=ax.transAxes,
            color='green' if (y_test[i] == pred[i]) else 'red')
```

Page 358

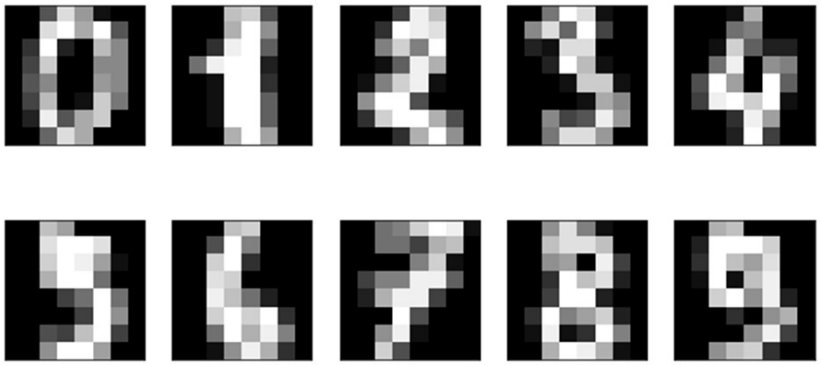
Multi-Class Classification

[0,1,2,3,4,5,6,7,8,9] versus [0,1,2,3,4,5,6,7,8,9]




```
from sklearn.manifold import Isomap
iso = Isomap(n_components=2)
projection = iso.fit_transform(digits.data)
```

```
plt.scatter(projection[:, 0], projection[:, 1], lw=0.1,
            c=digits.target,cmap=plt.cm.get_cmap('nipy_spectral',
10))
plt.colorbar(ticks=range(10), label='digit value')
plt.clim(-0.5, 9.5)
```



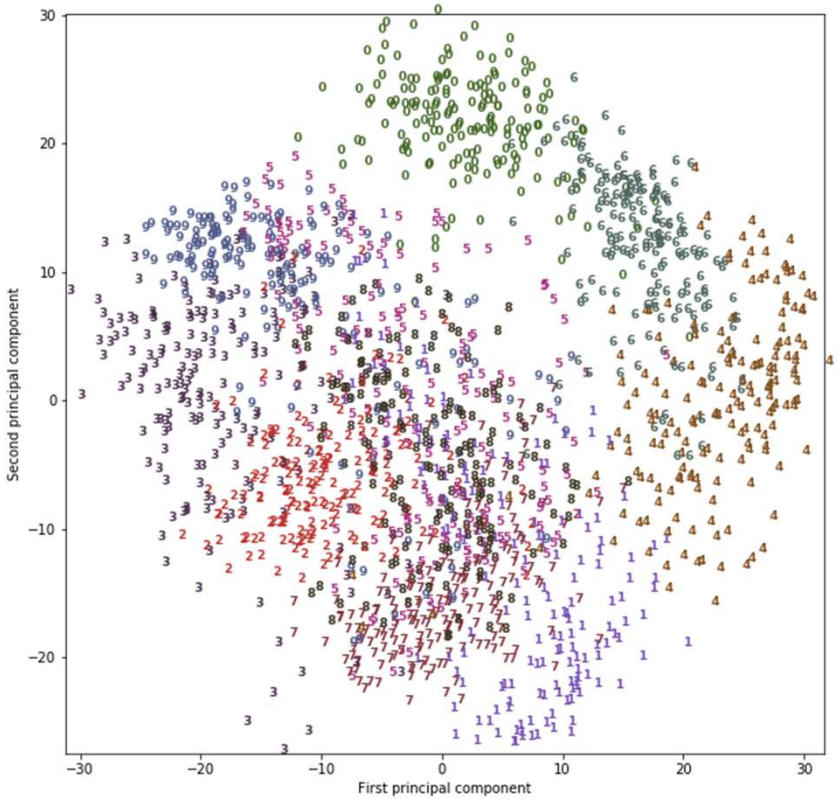
```
from sklearn.pipeline import Pipeline
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=3)

iso_knn_pipe = Pipeline(steps=[("iso", iso), ("knn", knn)])

iso_knn_pipe.fit(X_train, y_train)
pred = iso_knn_pipe.predict(X_test)

print("Accuracy: {:.3f}".format(accuracy_score(y_test, pred)))
```

Accuracy: 0.8733

Page 356

Multi-Class Classification

[0,1,2,3,4,5,6,7,8,9] versus [0,1,2,3,4,5,6,7,8,9]

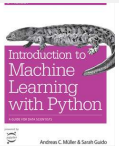



```
pca = PCA(n_components=2)
pca.fit(digits.data)
# transform the digits data onto the first two principal
components
digits_pca = pca.transform(digits.data)
```

```
for i in range(len(digits.data)):
    # actually plot the digits as text instead of using scatter
    plt.text(digits_pca[i, 0], digits_pca[i, 1], str(digits.target[i]),
             color = colors[digits.target[i]],
             fontdict={'weight': 'bold', 'size': 9})
plt.xlabel("First principal component")
plt.ylabel("Second principal component")
```

```
pca_knn_pipe = Pipeline(steps=[("psa", psa), ("knn", knn)])

pca_knn_pipe.fit(X_train, y_train)
pred = pca_knn_pipe.predict(X_test)

print("Accuracy: {:.3f}".format(accuracy_score(y_test, pred)))
```

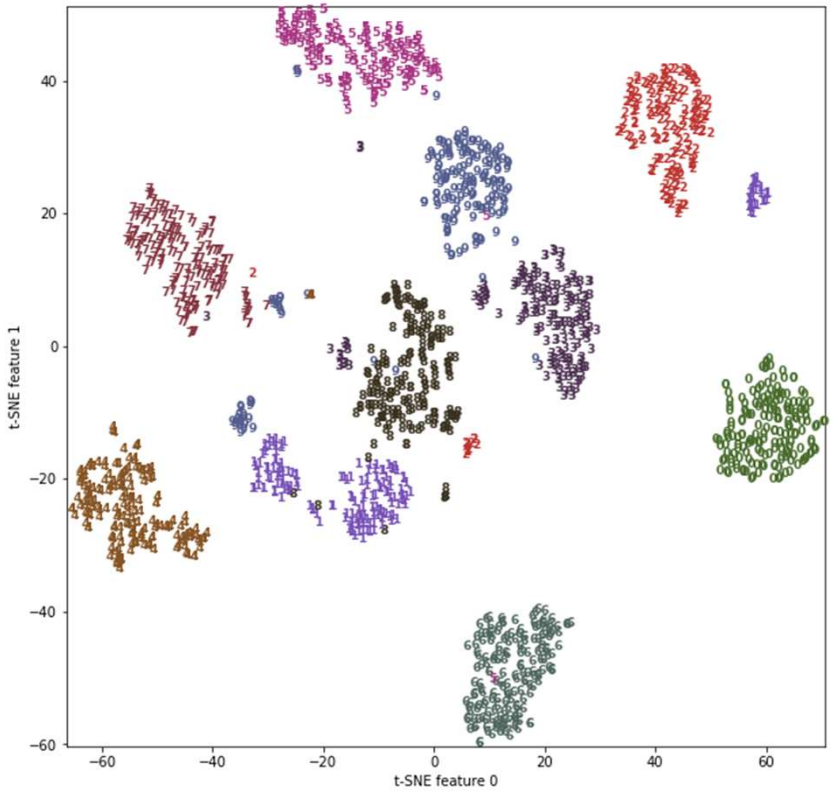
Accuracy: 0.964



Section 3.4.3

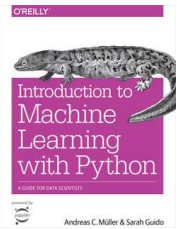
Multi-Class Classification

[0,1,2,3,4,5,6,7,8,9] versus [0,1,2,3,4,5,6,7,8,9]




```
from sklearn.manifold import TSNE
tsne = TSNE(random_state=42)
# use fit_transform instead of fit, as TSNE has no transform
method
digits_tsne = tsne.fit_transform(digits.data)
```

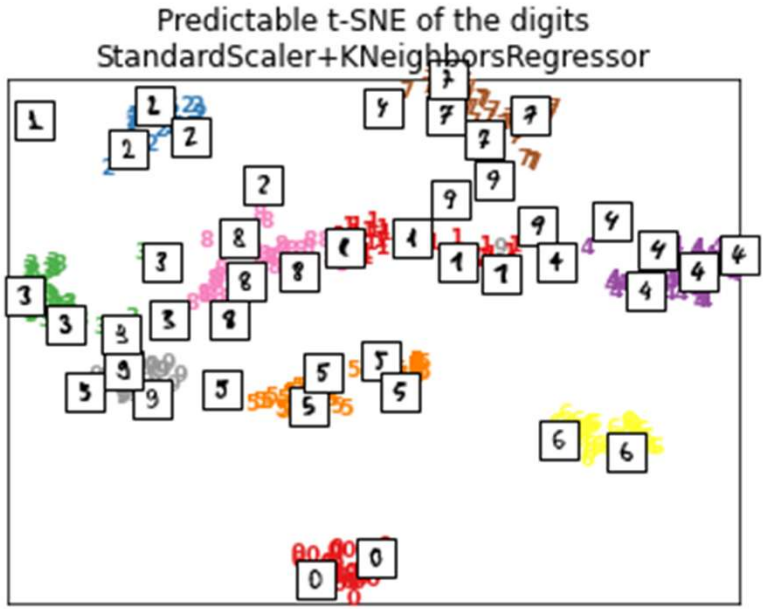
```
for i in range(len(digits.data)):
    # actually plot the digits as text instead of using scatter
    plt.text(digits_tsne[i, 0], digits_tsne[i, 1], str(digits.target[i]),
             color = colors[digits.target[i]],
             fontdict={'weight': 'bold', 'size': 9})
plt.xlabel("t-SNE feature 0")
plt.ylabel("t-SNE feature 1")
```



Section 3.4.3

Multi-Class Classification

[0,1,2,3,4,5,6,7,8,9] versus [0,1,2,3,4,5,6,7,8,9]




```
ptsne_knn = PredictableTSNE("transformer", TSNE(),
"estimator', KNeighborsRegressor())
ptsne_knn.fit(X_train, y_train)
```

```
X_test_tsne2 = ptsne_knn.transform(X_test)
plot_embedding(
    X_test_tsne2,
    y_test,
    imgs_test,
    "Predictable t-SNE of the digits\n
     StandardScaler+KNeighborsRegressor",
)
```

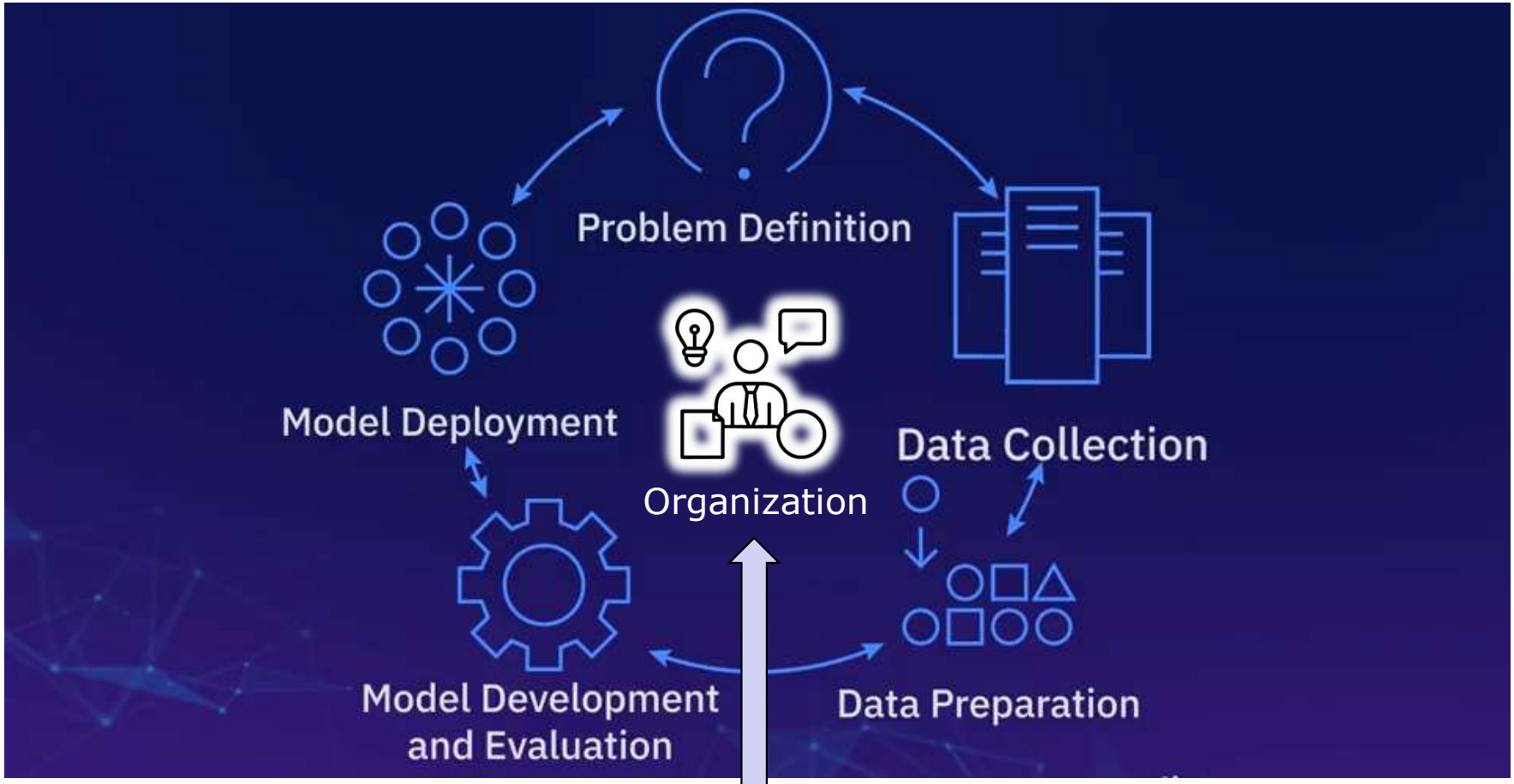
```
tsne_knn_pipe = Pipeline(steps=[("tsne", ptsne_knn), ("knn", knn)])

tsne_knn_pipe.fit(X_train, y_train)
pred = tsne_knn_pipe.predict(X_test)

print("Accuracy: {:.3f}".format(accuracy_score(y_test, pred)))
```

```
Accuracy: 0.984
```

https://onnx.ai/sklearn-onnx/auto_examples/plot_custom_model.html

Introduction to Machine Learning

Building a predictive model is a circular process.

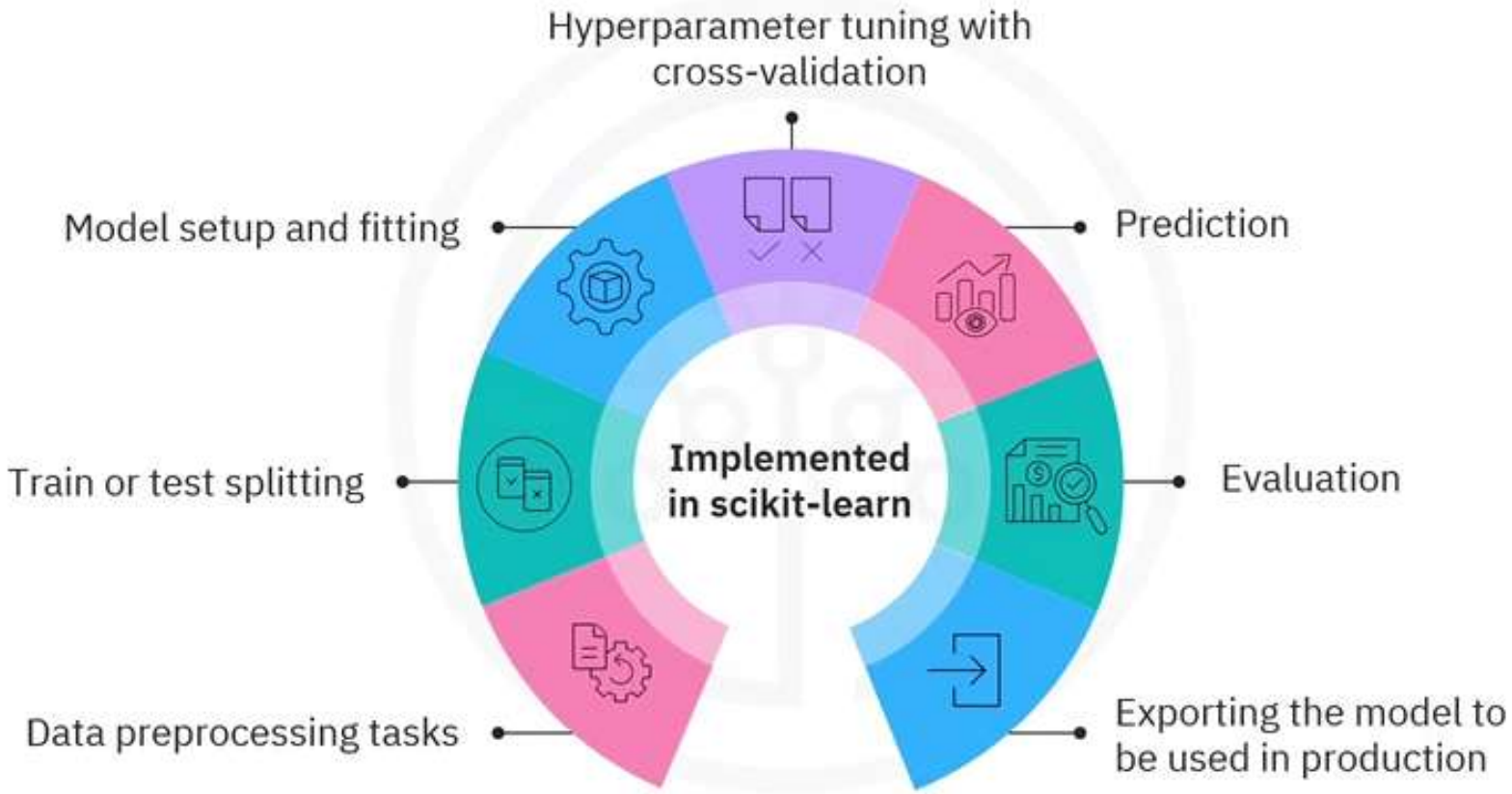


Expertise

- Depends on the organization
- Interpret the information in the right way

Introduction to Machine Learning

Building a predictive model is a circular process.



Joseph Santarcangelo & Joss Grossman, Machine Learning with Python, IBM Skills network, 2025

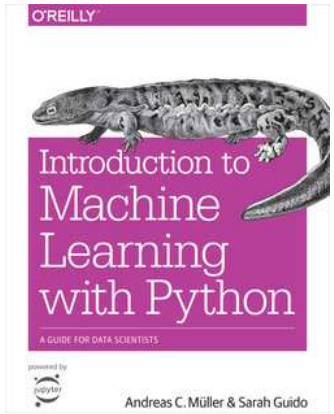
Introduction to Machine Learning

- 5.3 Evaluating Metrics & Scoring

- Strengths:
  - There are multiple ways to measure performance of an algorithm
- Weakness:
  - Accuracy is not always the best metric
  - Precision & Recall have to be balanced

$$f_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Andreas C. Müller, Sarah Guido, Introduction to Machine Learning with Python, O'Reilly Media, October 2016



Conclusion

Learning outcomes of this course covered today

- Classification has both type I & type II errors