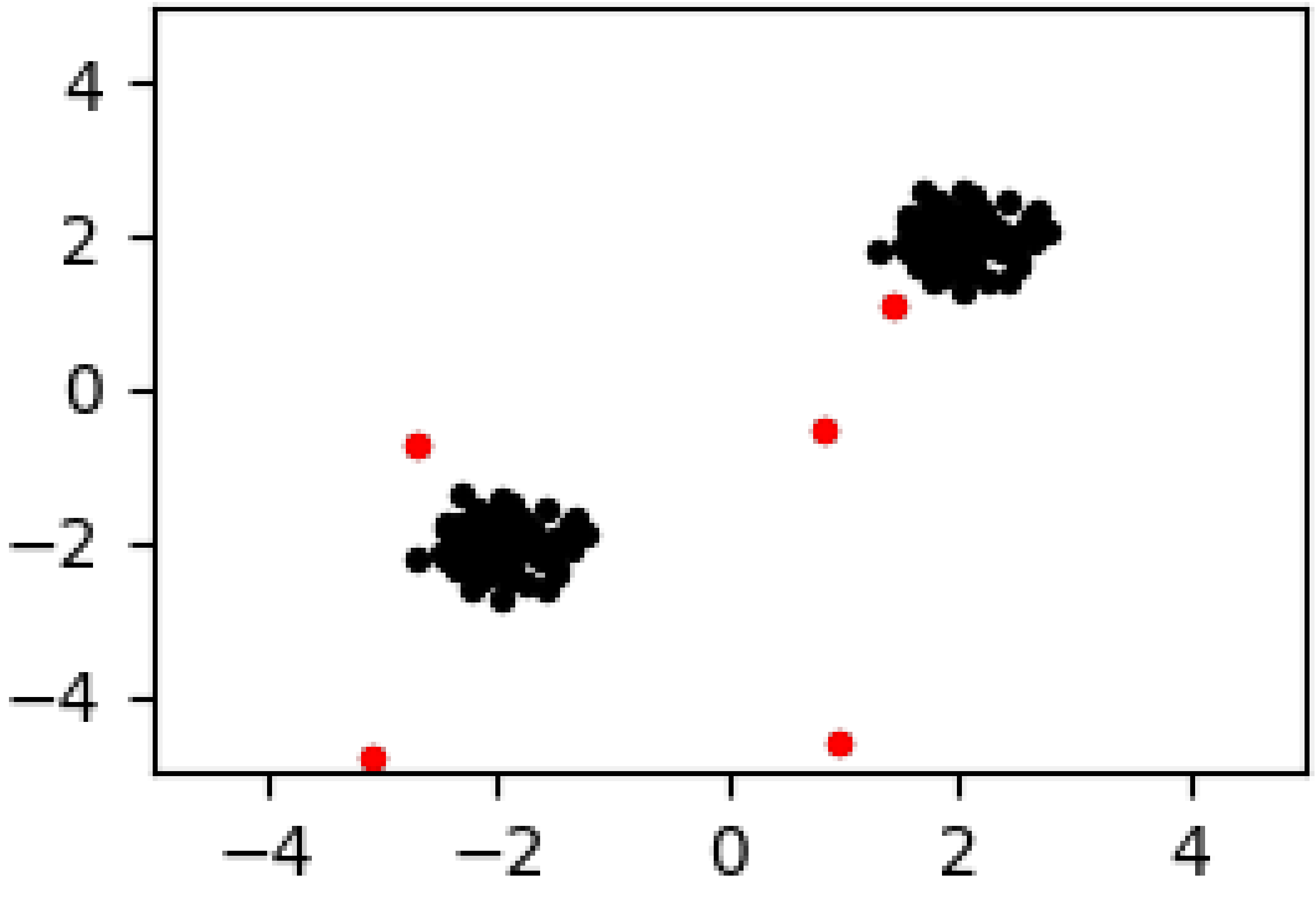
Anomaly detection

DESIGNING MACHINE LEARNING WORKFLOWS IN PYTHON

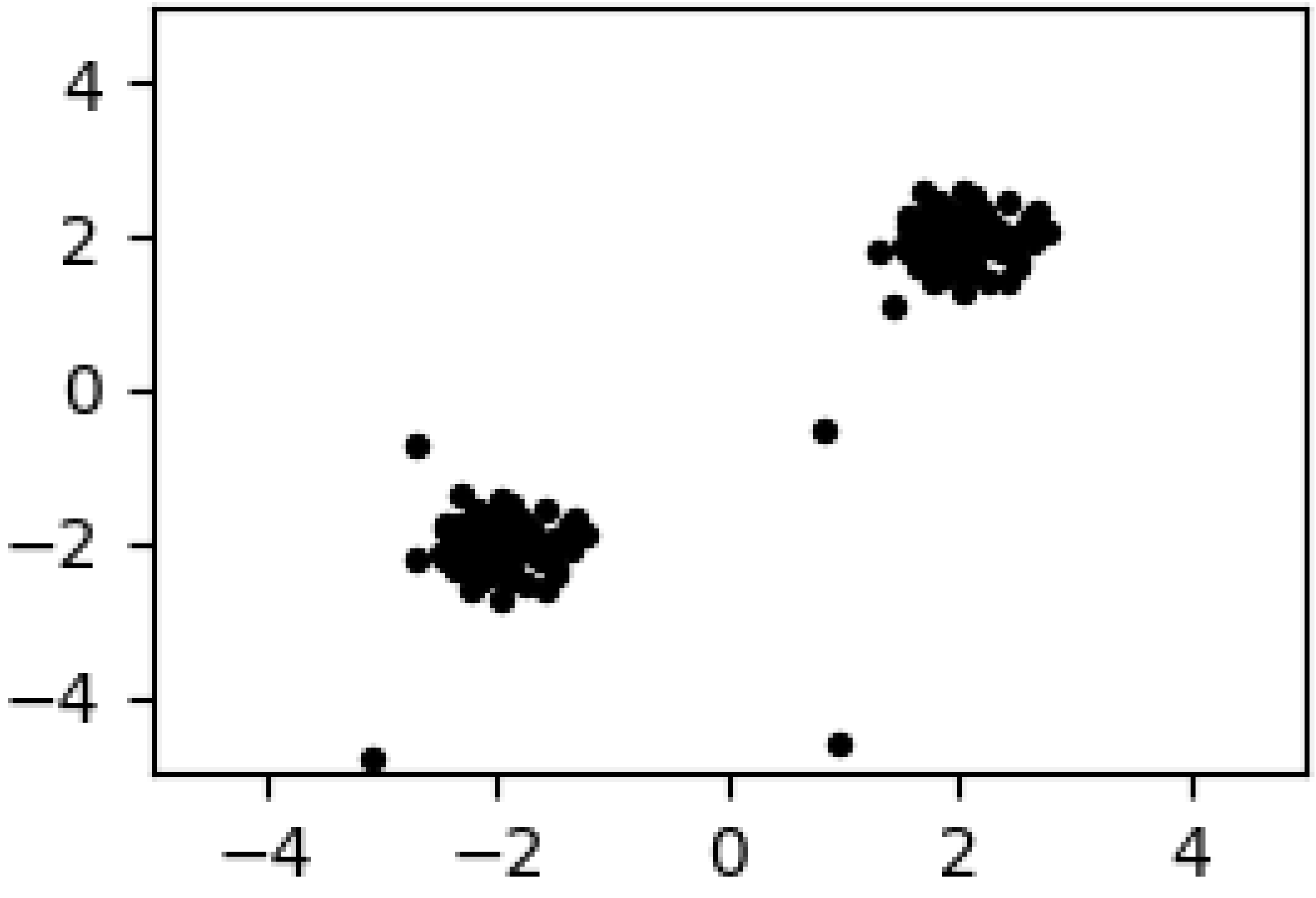
Dr. Chris Anagnostopoulos
Honorary Associate Professor

Anomalies and outliers

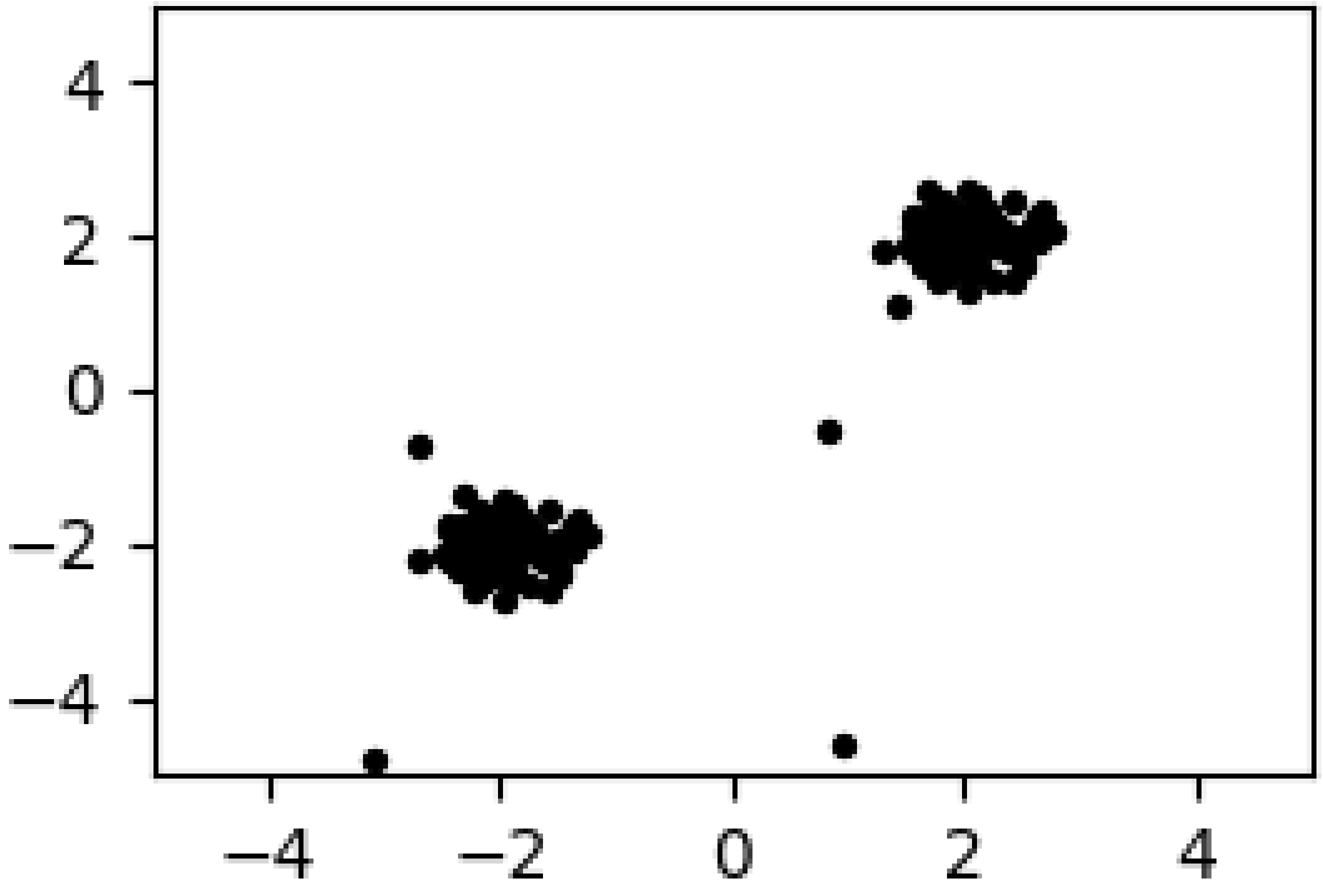
Supervised



Unsupervised

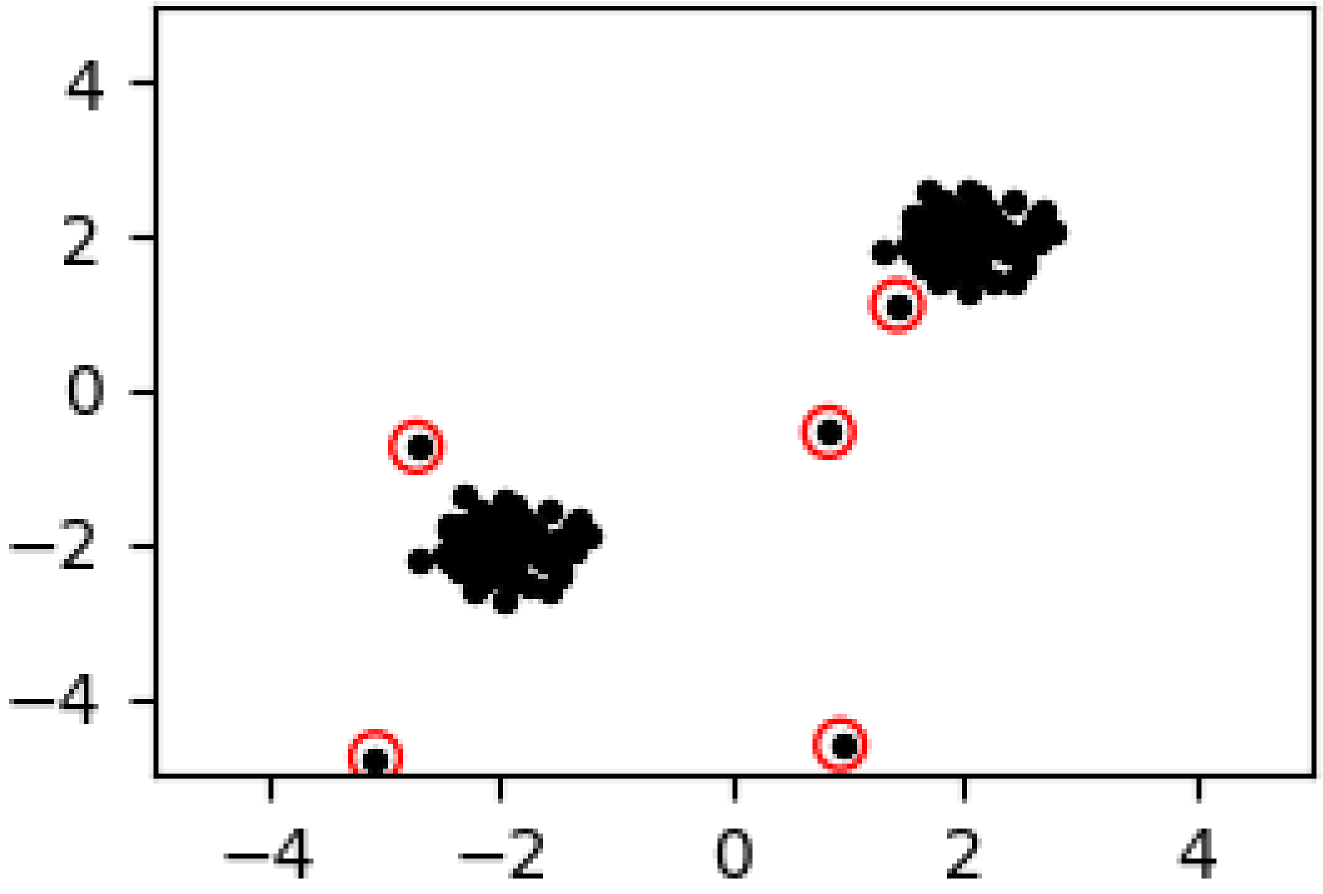


Anomalies and outliers



- One of the two classes is very rare
- Extreme case of dataset shift
- Examples:
  - cybersecurity
  - fraud detection
  - anti-money laundering
  - fault detection

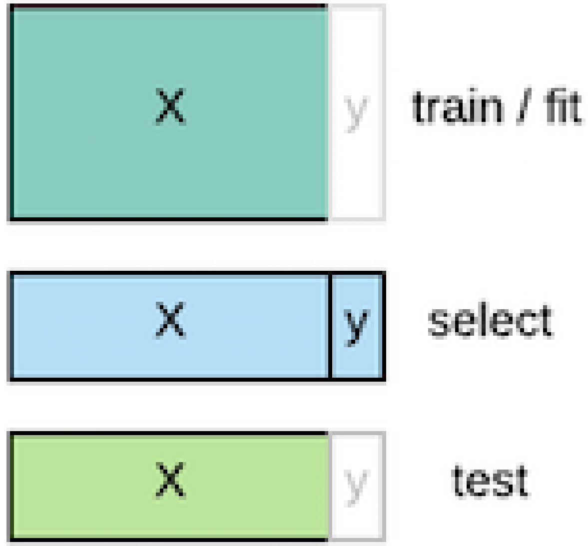
Unsupervised workflows



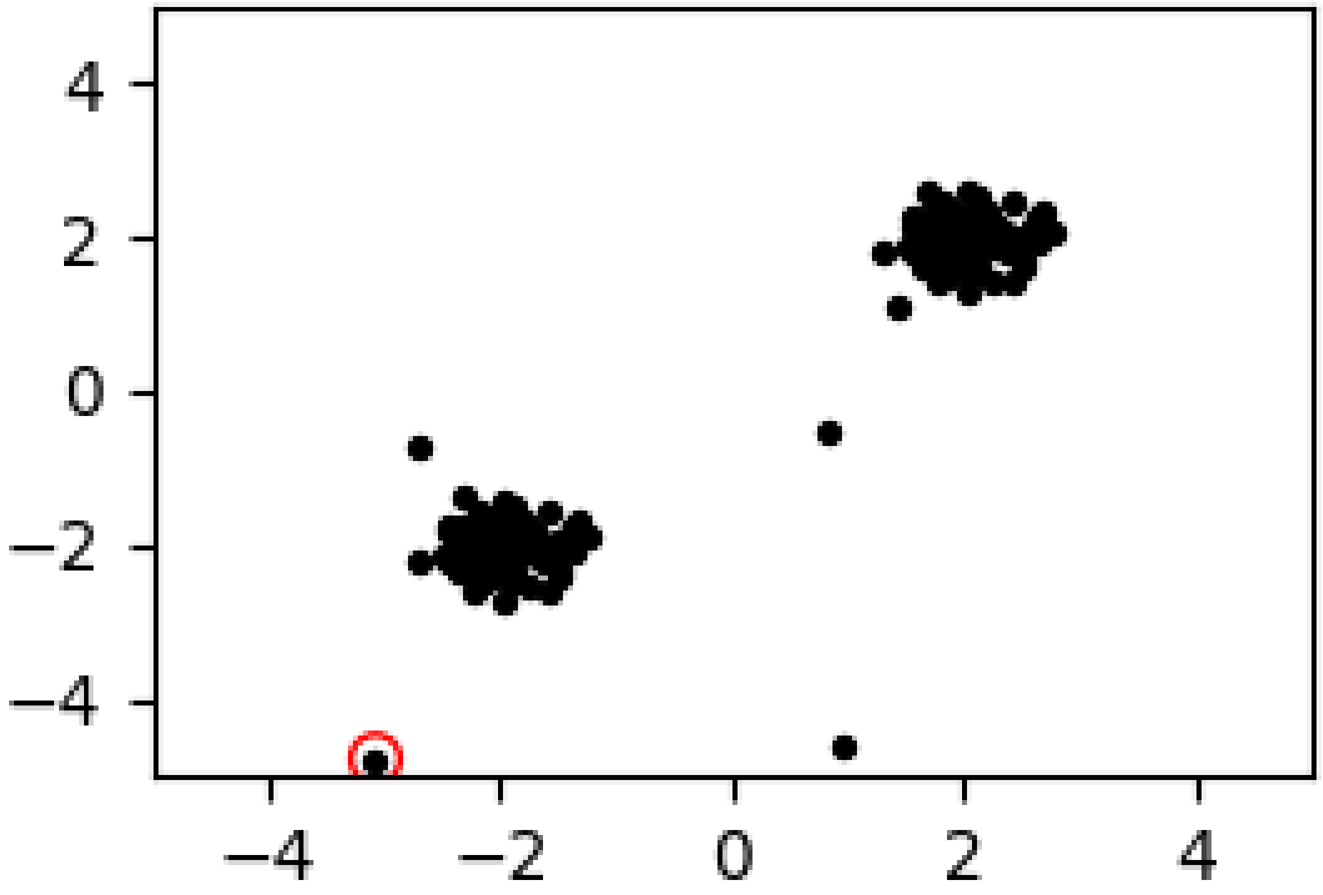
- How to fit an algorithm without labels?
- How to estimate its performance?

Careful use of a handful of labels:

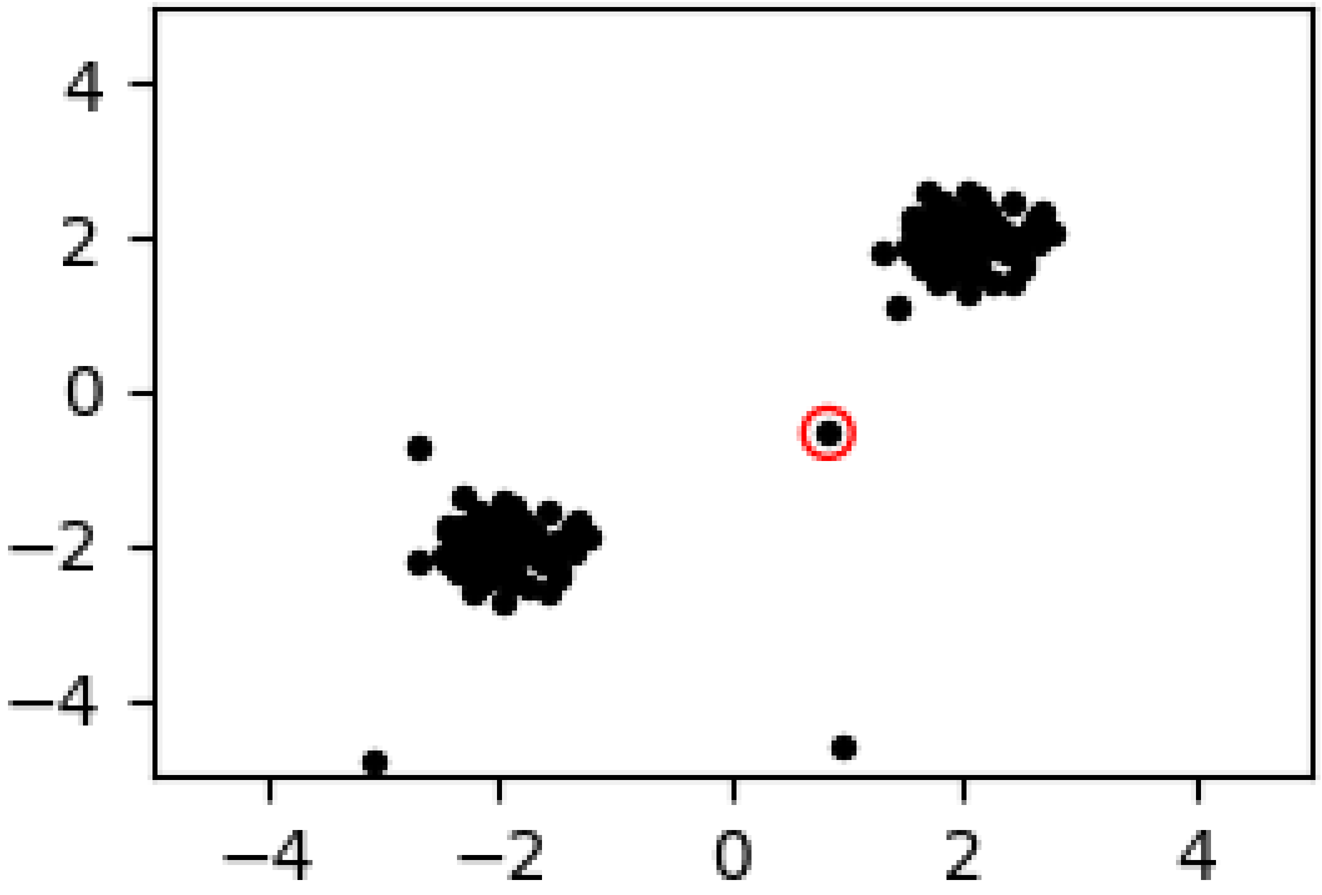
- too few for training without overfitting
- just enough for model selection
- drop unbiased estimate of accuracy



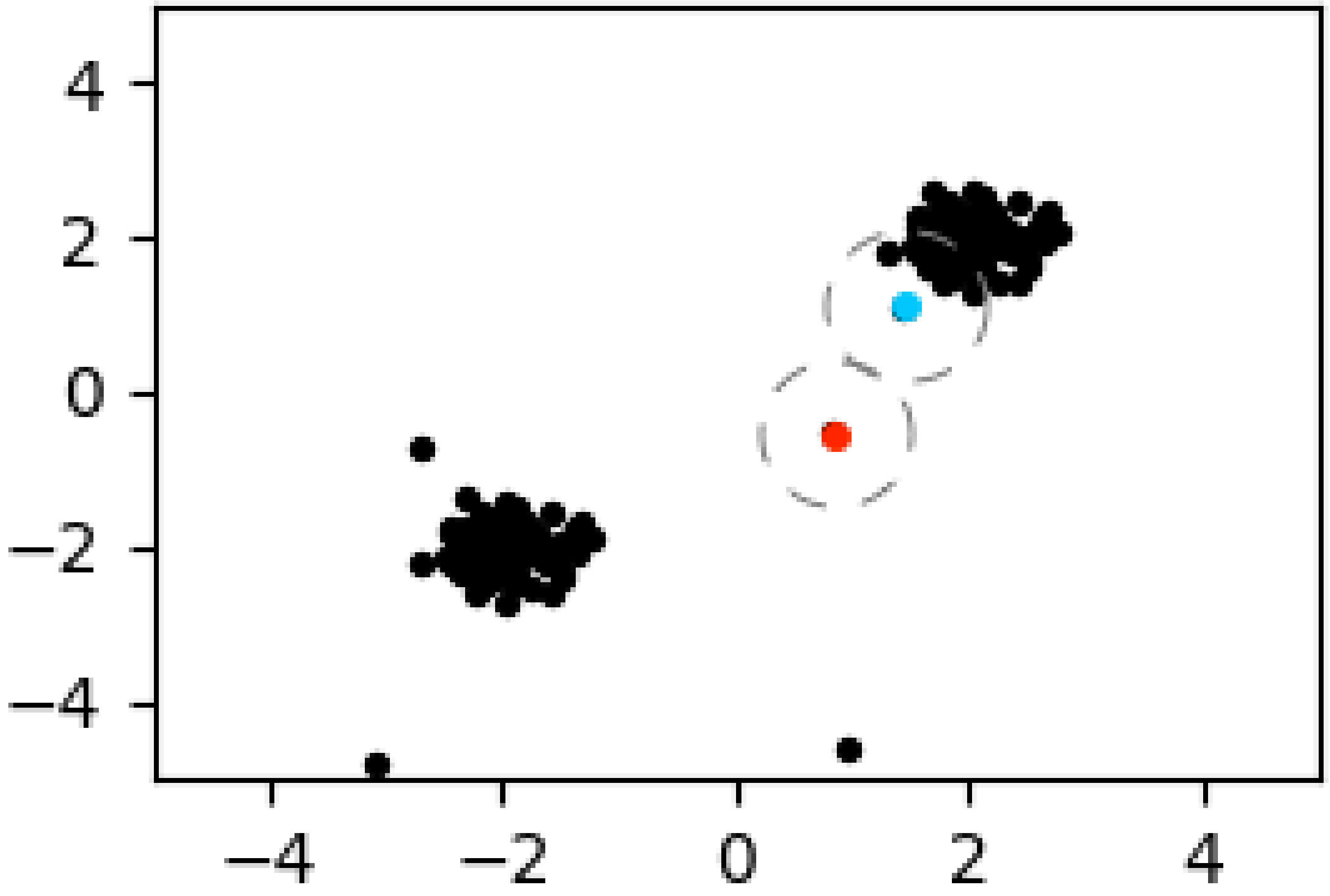
- **Outlier**: a datapoint that lies outside the range of the majority of the data



- **Local outlier**: a datapoint that lies in an isolated region without other data



Local outlier factor (LoF)



Local outlier factor (LoF)

```
from sklearn.neighbors import
   LocalOutlierFactor as lof
clf = lof()
y_pred = clf.fit_predict(X)
```

```
y_pred[:4]
```

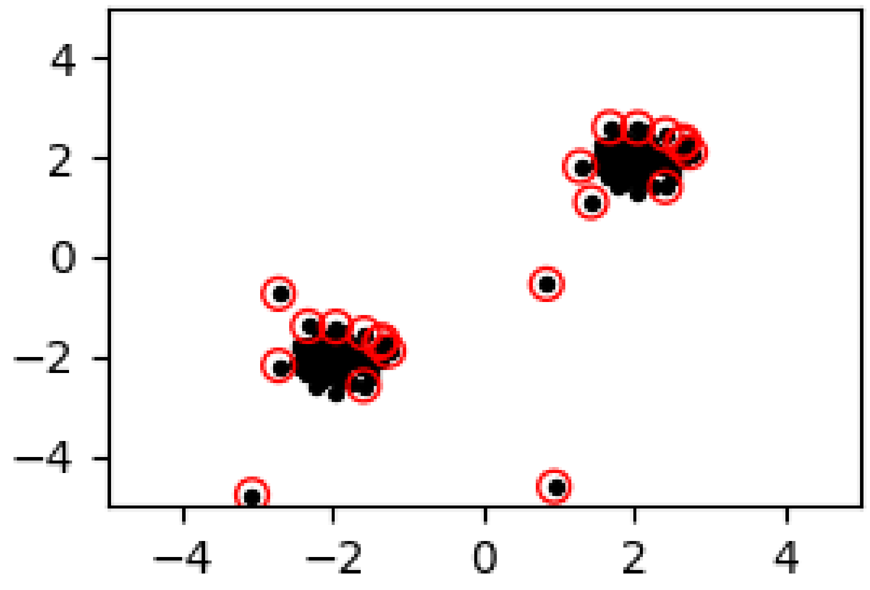
```
array([ 1,  1,  1, -1])
```

```
clf.negative_outlier_factor_[:4]
```

```
array([-0.99, -1.02, -1.08 , -0.97])
```

```
confusion_matrix(
   y_pred, ground_truth)
```

```
array([[  5,  16],
       [  0, 184]])
```

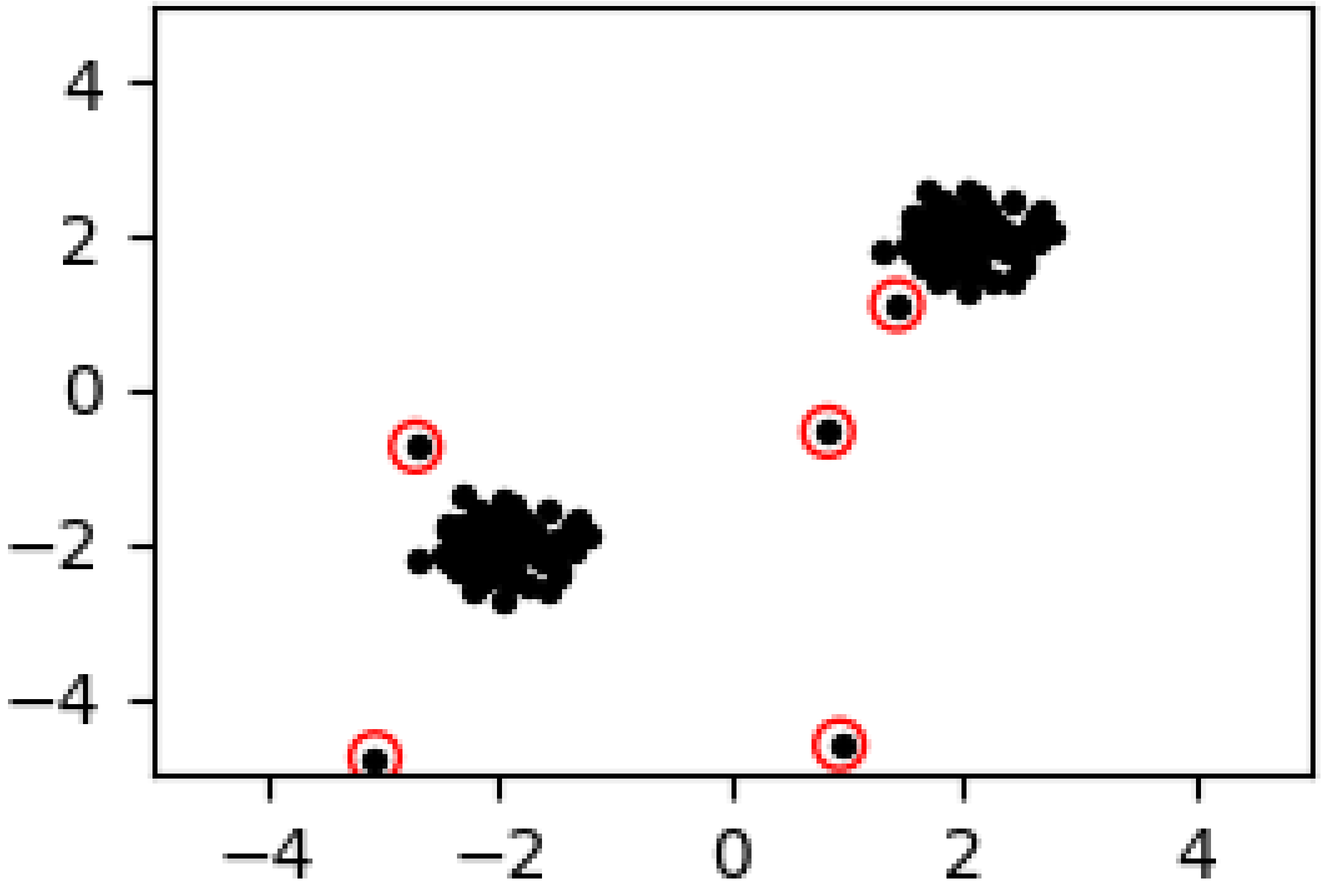


Local outlier factor (LoF)

```
clf = lof(contamination=0.02)
y_pred = clf.fit_predict(X)
```

```
confusion_matrix(
   y_pred, ground_truth)
```

```
array([[  5,   0],
       [  0, 200]])
```



Who needs labels anyway!

DESIGNING MACHINE LEARNING WORKFLOWS IN PYTHON

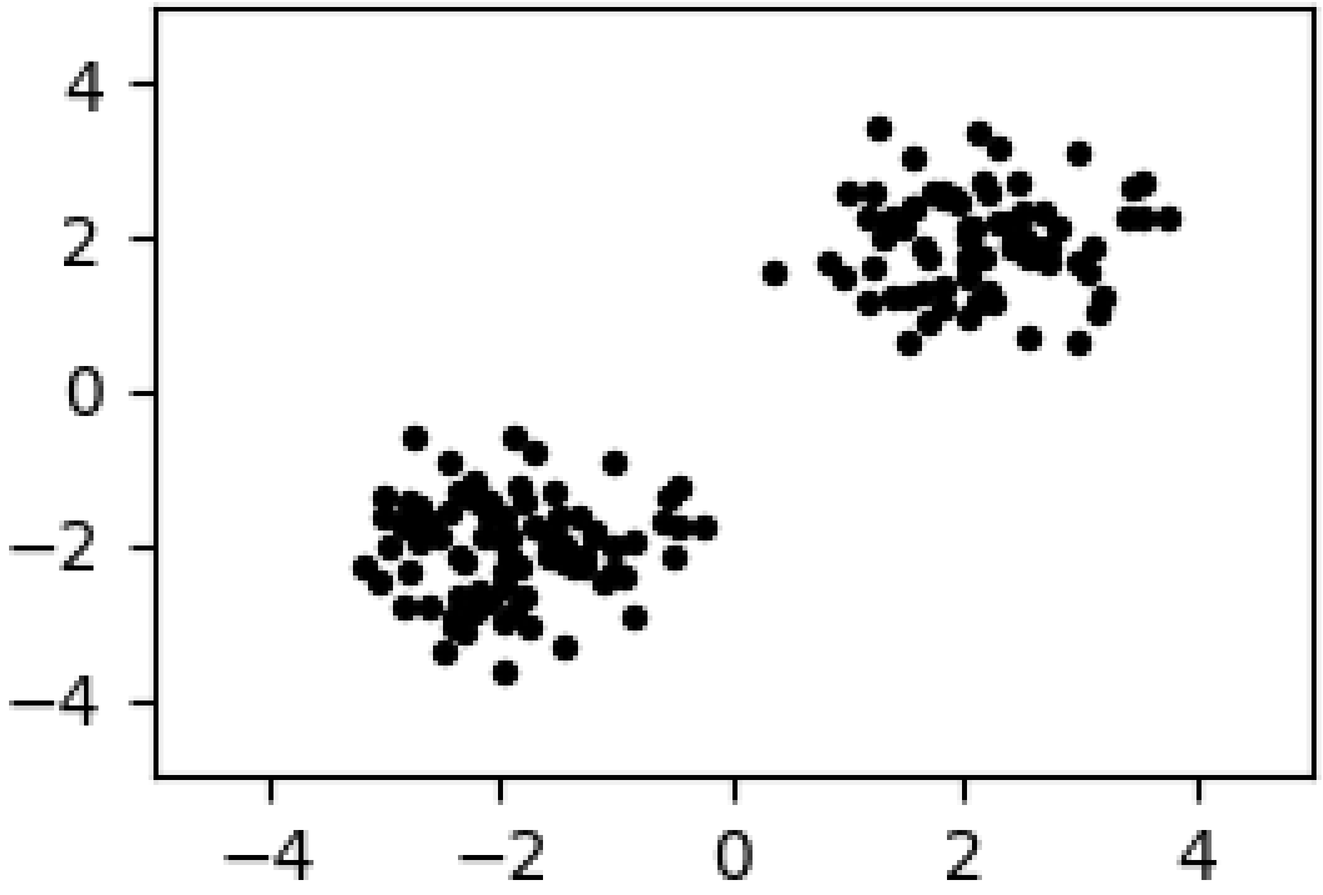
Novelty detection

DESIGNING MACHINE LEARNING WORKFLOWS IN PYTHON

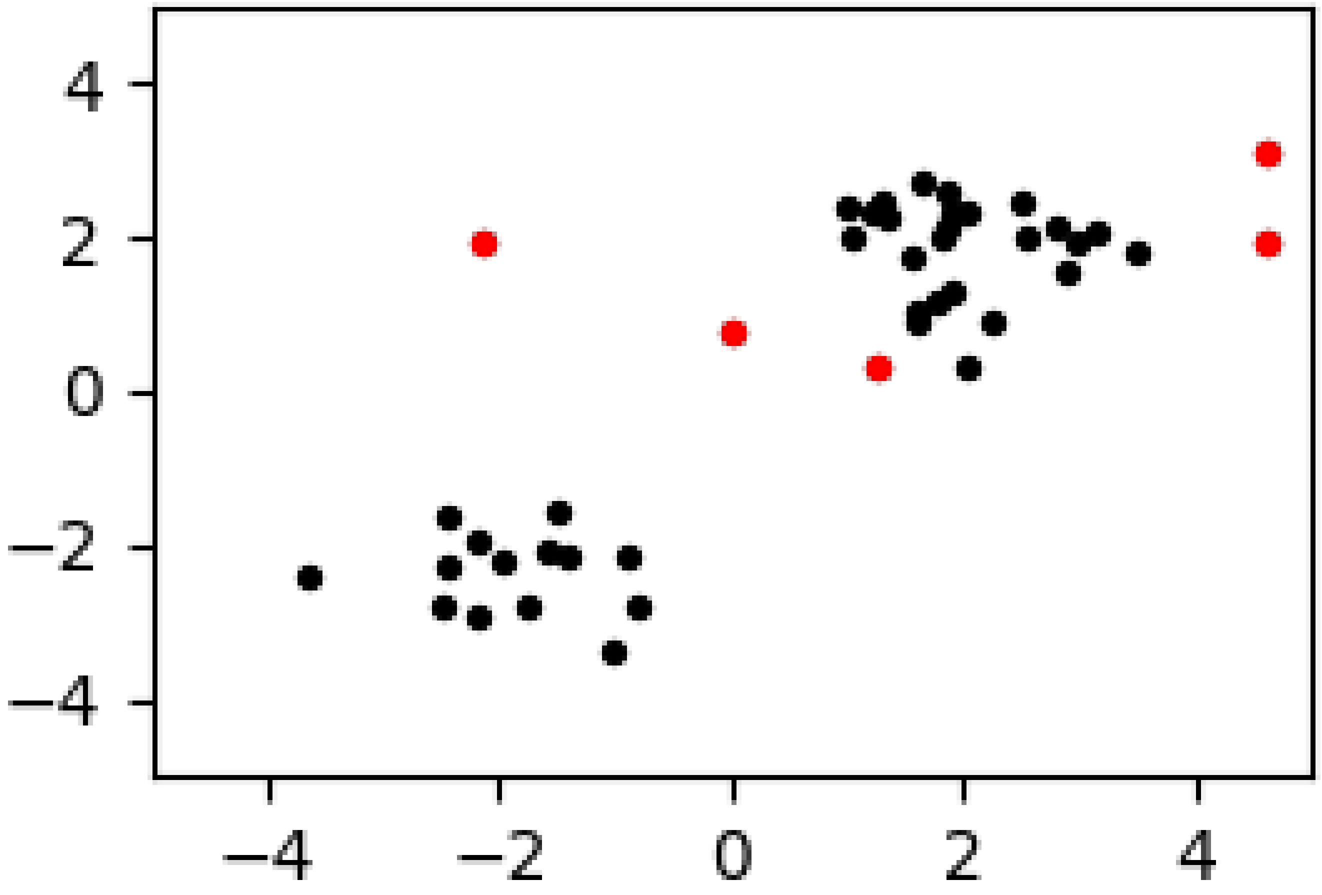
Dr. Chris Anagnostopoulos
Honorary Associate Professor

One-class classification

Training data without anomalies:



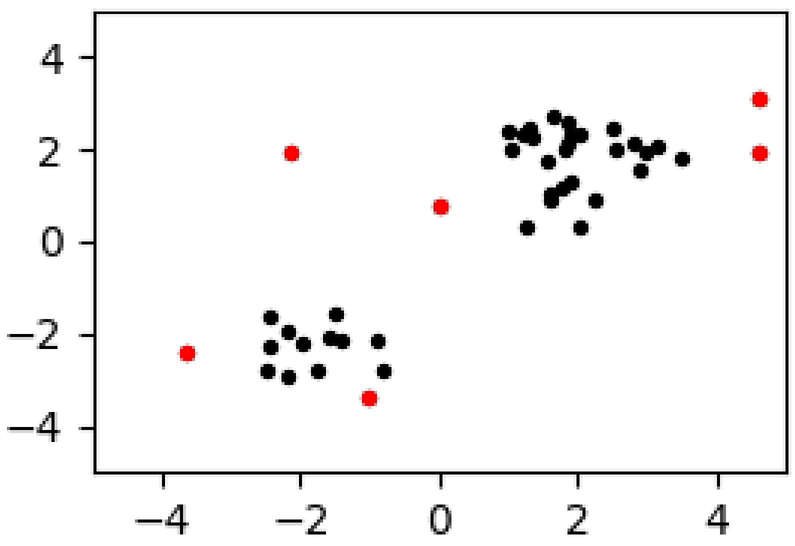
Future / test data with anomalies:



Novelty LoF

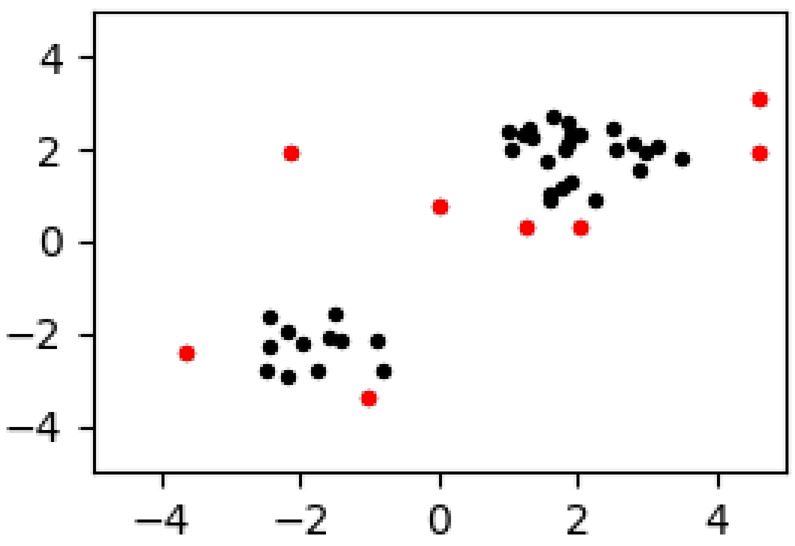
Workaround

```
preds = lof().fit_predict(
    np.concatenate([X_train, X_test]))
preds = preds[X_train.shape[0]:]
```



Novelty LoF

```
clf = lof(novelty=True)
clf.fit(X_train)
y_pred = clf.predict(X_test)
```



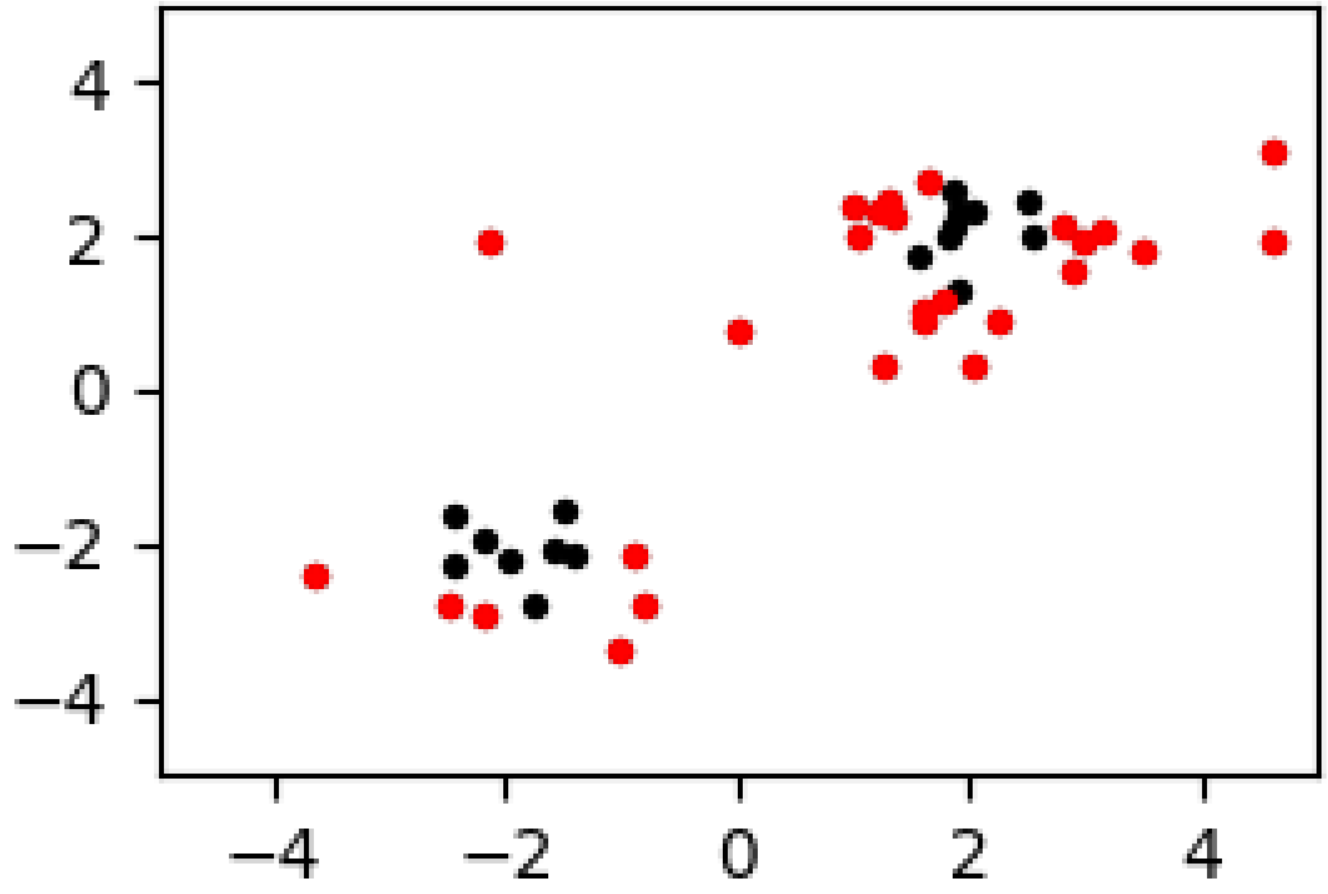
One-class Support Vector Machine

```python
clf = OneClassSVM()

clf.fit(X_train)
y_pred = clf.predict(X_test)
```

```python
y_pred[:4]
```

```
array([ 1,  1,  1, -1])
```

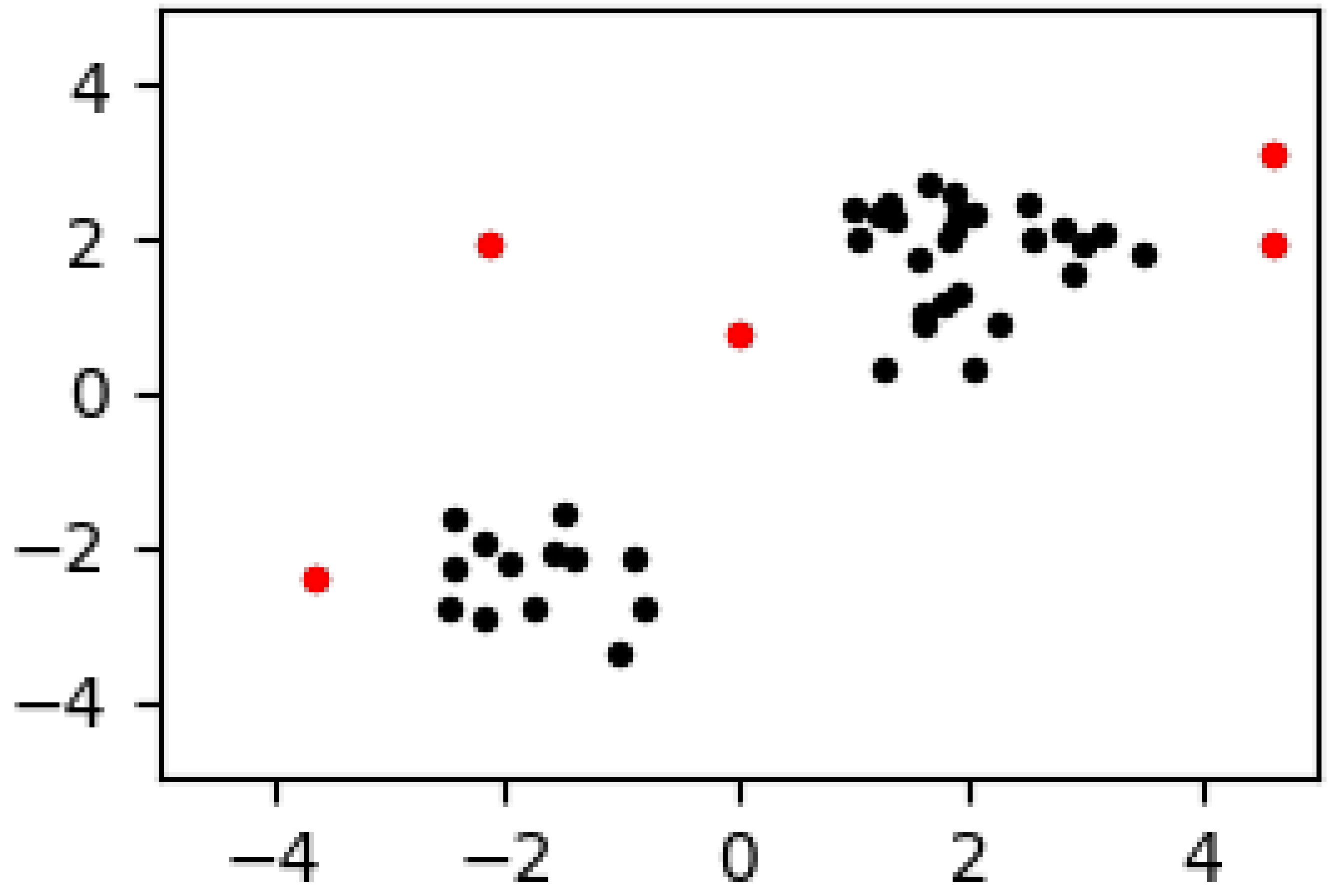


One-class Support Vector Machine

```
clf = OneClassSVM()
clf.fit(X_train)
y_scores = clf.score_samples(X_test)

threshold = np.quantile(y_scores, 0.1)
```

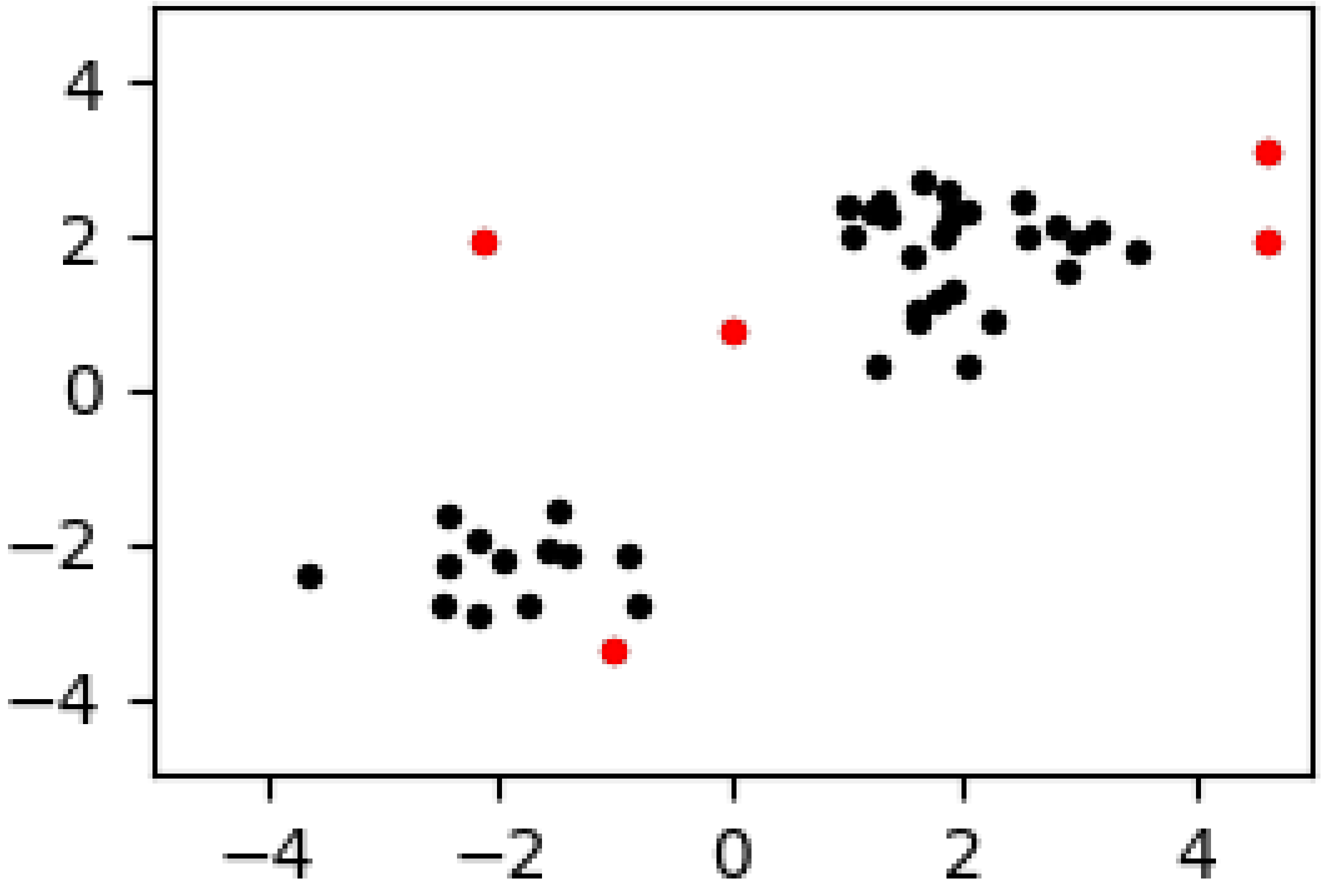
```
y_pred = y_scores <= threshold
```



Isolation Forests

```
clf = IsolationForest()
clf.fit(X_train)
y_scores = clf.score_samples(X_test)
```

```
clf = LocalOutlierFactor(novelty=True)
clf.fit(X_train)
y_scores = clf.score_samples(X_test)
```



```python
clf_lof = LocalOutlierFactor(novelty=True).fit(X_train)
clf_isf = IsolationForest().fit(X_train)
clf_svm = OneClassSVM().fit(X_train)
```

```python
roc_auc_score(y_test, clf_lof.score_samples(X_test)
```

```
0.9897
```

```python
roc_auc_score(y_test, clf_isf.score_samples(X_test))
```

```
0.9692
```

```python
roc_auc_score(y_test, clf_svm.score_samples(X_test))
```

```
0.9948
```

```python
clf_lof = LocalOutlierFactor(novelty=True).fit(X_train)
clf_isf = IsolationForest().fit(X_train)
clf_svm = OneClassSVM().fit(X_train)
```

```python
accuracy_score(y_test, clf_lof.predict(X_test))
```

```
0.9318
```

```python
accuracy_score(y_test, clf_isf.predict(X_test))
```

```
0.9545
```

```python
accuracy_score(y_test, clf_svm.predict(X_test))
```

```
0.5
```

What's new?

DESIGNING MACHINE LEARNING WORKFLOWS IN PYTHON

Distance-based learning

DESIGNING MACHINE LEARNING WORKFLOWS IN PYTHON

Dr. Chris Anagnostopoulos
Honorary Associate Professor

Distance and similarity

```python
from sklearn.neighbors import DistanceMetric as dm
dist = dm.get_metric('euclidean')
X = [[0,1], [2,3], [0,6]]
dist.pairwise(X)
```

```
array([[0.        , 2.82842712, 5.        ],
       [2.82842712, 0.        , 3.60555128],
       [5.        , 3.60555128, 0.        ]])
```

```python
X = np.matrix(X)
np.sqrt(np.sum(np.square(X[0,:] - X[1,:])))
```

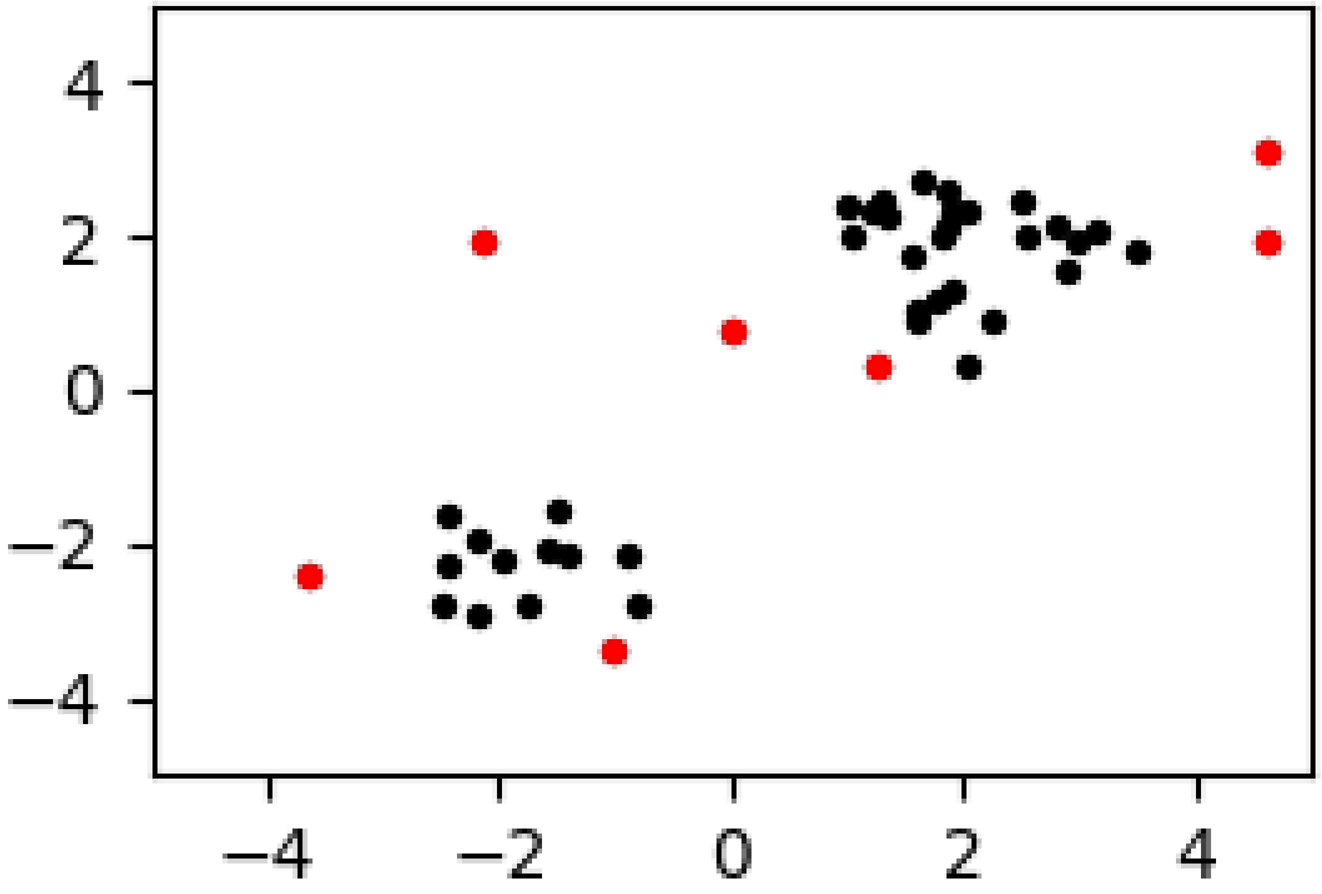
```
2.82842712
```

Non-Euclidean Local Outlier Factor

```
clf = LocalOutlierFactor(
    novelty=True, metric='chebyshev')
clf.fit(X_train)
y_pred = clf.predict(X_test)
```

```
dist = dm.get_metric('chebyshev')
X = [[0,1], [2,3], [0,6]]
dist.pairwise(X)
```

```
array([[0., 2., 5.],
       [2., 0., 3.],
       [5., 3., 0.]])
```



Are all metrics similar?

Hamming distance matrix:

```
dist = dm.get_metric('hamming')
X = [[0,1], [2,3], [0,6]]
dist.pairwise(X)
```

```
array([[0. , 1. , 0.5],
       [1. , 0. , 1. ],
       [0.5, 1. , 0. ]])
```

Are all metrics similar?

```
from scipy.spatial.distance import pdist

X = [[0,1], [2,3], [0,6]]
pdist(X, 'cityblock')
```

```
array([4., 5., 5.])
```

```
from scipy.spatial.distance import \
    squareform
squareform(pdist(X, 'cityblock'))
```

```
array([[0., 4., 5.],
       [4., 0., 5.],
       [5., 5., 0.]])
```

A real-world example

The Hepatitis dataset:

```
   Class   AGE  SEX  STEROID    ...
0    2.0  40.0  0.0      0.0    ...
1    2.0  30.0  0.0      0.0    ...
2    1.0  47.0  0.0      1.0    ...
```

[1] https://archive.ics.uci.edu/ml/datasets/Hepatitis

A real-world example

Euclidean distance:

```
squareform(pdist(X_hep, 'euclidean'))
```

```
[[  0.   127.    64.1]
 [127.     0.   128.2]
 [ 64.1 128.2   0.  ]]
```

- 1 nearest to 3: *wrong* class

Hamming distance:

```
squareform(pdist(X_hep, 'hamming'))
```

```
[[0.  0.5 0.7]
 [0.5 0.  0.6]
 [0.7 0.6 0. ]]
```

- 1 nearest to 2: right class

A bigger toolbox

DESIGNING MACHINE LEARNING WORKFLOWS IN PYTHON

Unstructured data

DESIGNING MACHINE LEARNING WORKFLOWS IN PYTHON

Dr. Chris Anagnostopoulos
Honorary Associate Professor

Structured versus unstructured

```
  Class   AGE  SEX  STEROID    ...
0    2.0  50.0  2.0      1.0    ...
1    2.0  40.0  1.0      1.0    ...
...
```

```
            label                                               sequence
0           VIRUS  AVTVVPDPTCCGTLSFKVPKDAKKGKHLGTFDIRQAIMDYGGLHSQ...
1  IMMUNE SYSTEM  QVQLQQPGAELVKPGASVKLSCKASGYTFTSYWMHWVKQRPGRGLE...
2  IMMUNE SYSTEM  QAVVTQESALTTSPGETVTLTCRSSTGAVTTSNYANWVQEKPDHLF...
3           VIRUS  MSQVTEQSVRFQTALASIKLIQASAVLDLTEDDFDFLTSNKVWIAT...
...
```

Can we build a detector that flags viruses as anomalous in this data?

```python
import stringdist
stringdist.levenshtein('abc', 'acc')
```

```
1
```

```python
stringdist.levenshtein('acc', 'cce')
```

```
2
```

```
             label   sequence
169  IMMUNE SYSTEM  ILSALVGIV
170  IMMUNE SYSTEM  ILSALVGIL
```

```python
stringdist.levenshtein('ILSALVGIV', 'ILSALVGIL')
```

```
1
```

Some debugging

```
# This won't work
pdist(proteins['sequence'].iloc[:3], metric=stringdist.levenshtein)
```

```
Traceback (most recent call last):
ValueError: A 2-dimensional array must be passed.
```

Some debugging

```
sequences = np.array(proteins['sequence'].iloc[:3]).reshape(-1,1)

# This won't work for a different reason
pdist(sequences, metric=stringdist.levenshtein)
```

```
Traceback (most recent call last):
TypeError: argument 1 must be str, not numpy.ndarray
```

Some debugging

```
# This one works!!
def my_levenshtein(x, y):
    return stringdist.levenshtein(x[0], y[0])

pdist(sequences, metric=my_levenshtein)
```

```
array([136.,   2., 136.])
```

Protein outliers with precomputed matrices

```
# This takes 2 minutes for about 1000 examples
M = pdist(sequences, my_levenshtein)
```

LoF detector with a precomputed distance matrix:

```
# This takes 3 seconds
detector = lof(metric='precomputed', contamination=0.1)
preds = detector.fit_predict(M)
```

```
roc_auc_score(proteins['label'] == 'VIRUS', preds == -1)
```

```
0.64
```

Pick your distance

DESIGNING MACHINE LEARNING WORKFLOWS IN PYTHON