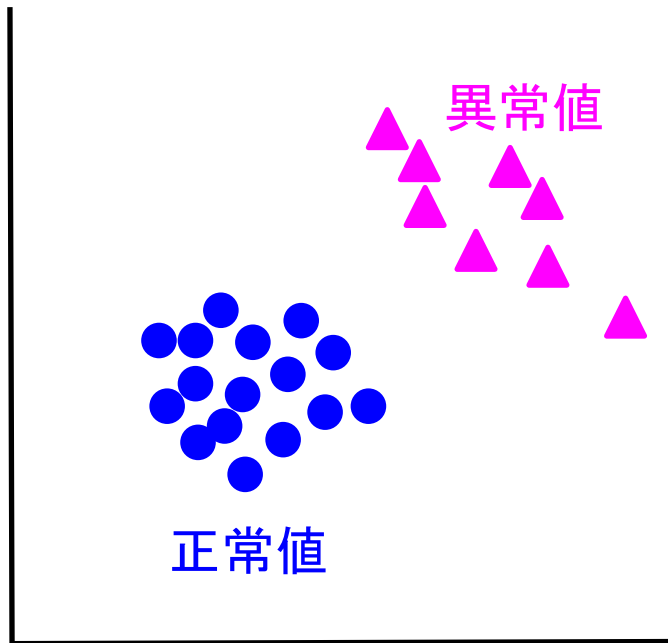


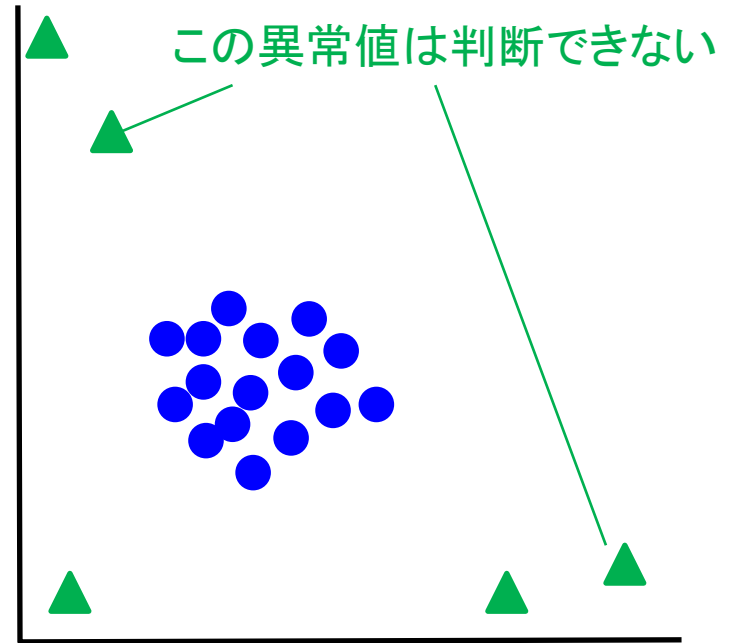
異常検知

- ① Novelty detection
訓練データに異常データが含まれない
- ② Outer detection
 - 訓練データに異常データがあるかもしれない
 - 異常があれば特定する

訓練データに全ての異常
が含まれない場合



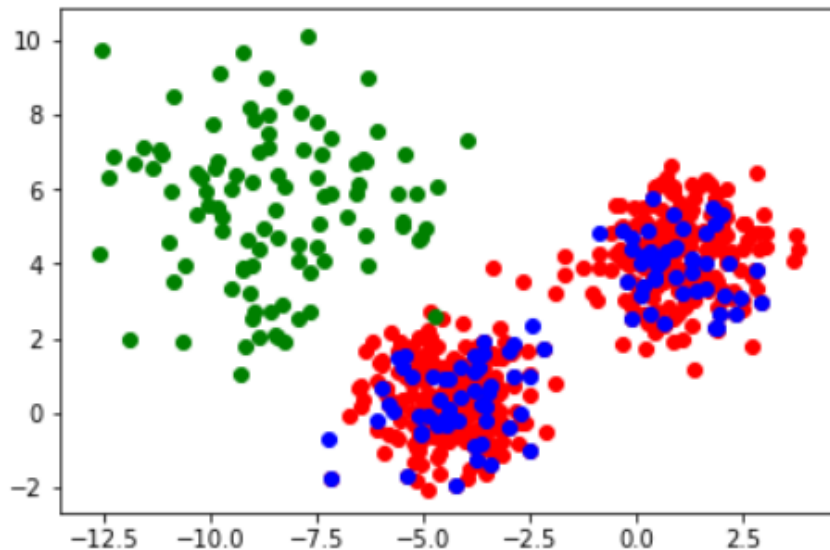
事前学習データ



Novelty detection

One-class SVMを実行する前の準備

```
%matplotlib inline
from sklearn import datasets
import matplotlib.pyplot as plt
X_train,_ = datasets.make_blobs(centers=2,random_state=3,cluster_std=1,n_samples=500)
X_inlier,_ = datasets.make_blobs(centers=2,random_state=3,cluster_std=1,n_samples=600)
X_inlier =X_inlier[500:600]
X_outlier,_ = datasets.make_blobs(centers=1,random_state=7,cluster_std=2,n_samples=100)
plt.scatter(X_train[:,0],X_train[:,1],c='red')
plt.scatter(X_inlier[:,0],X_inlier[:,1],c='blue')
plt.scatter(X_outlier[:,0],X_outlier[:,1],c='green')
plt.show()
```

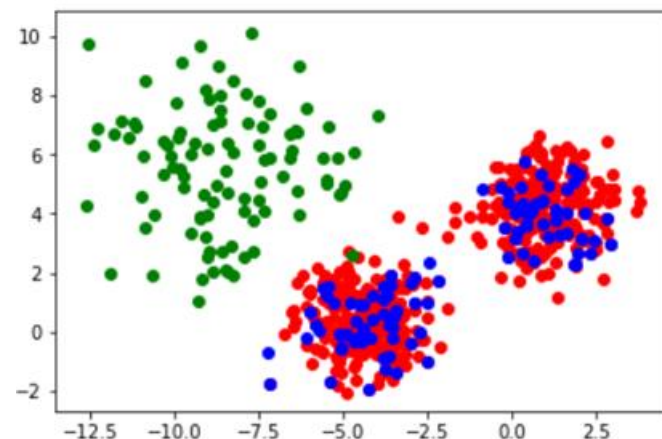
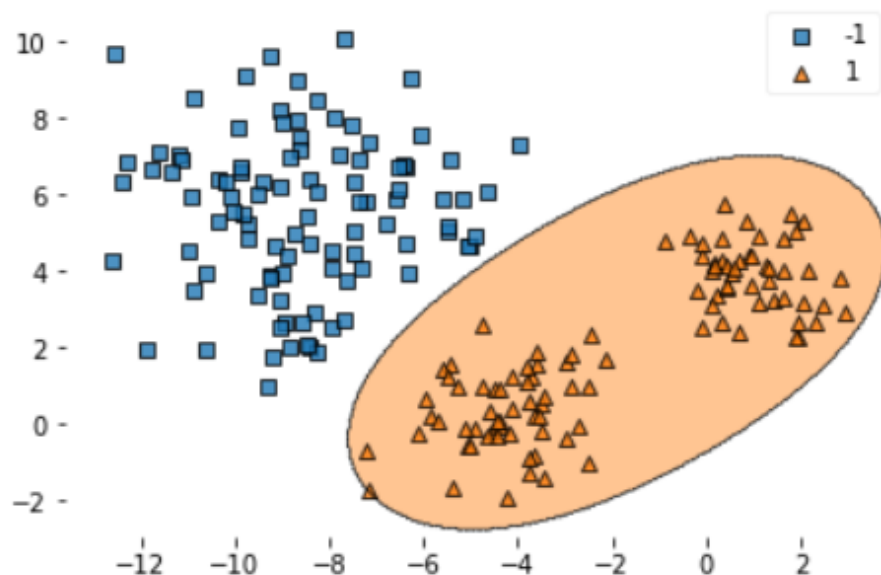


赤： 訓練データ(正常)
青： 正常のテストデータ
緑： 異常のテストデータ

コマンドプロンプトで `pip install mlxtend` を入れてインストールしておきます

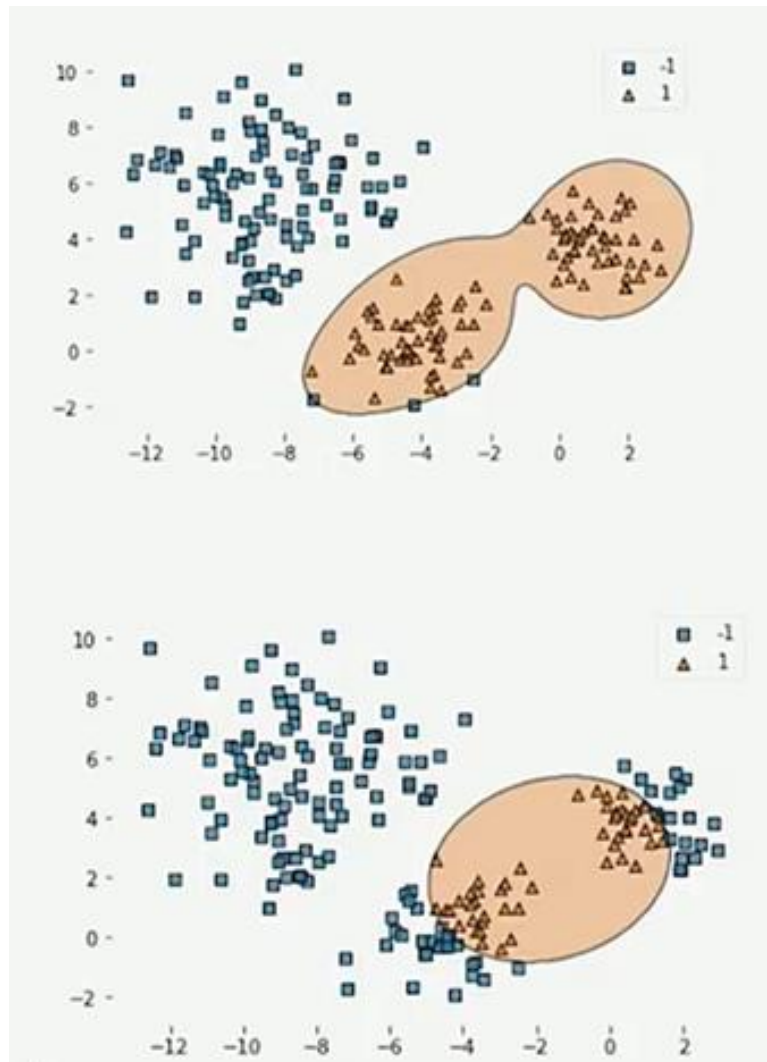
```
%matplotlib inline
from sklearn import svm
import numpy as np
from mlxtend.plotting import plot_decision_regions
clf = svm.OneClassSVM(nu=0.001, gamma=0.01, kernel="rbf")
clf.fit(X_train)
y_pred_inlier = clf.predict(X_inlier)
y_pred_outlier = clf.predict(X_outlier)
X = np.r_[X_inlier, X_outlier]
y = np.r_[y_pred_inlier, y_pred_outlier]
plot_decision_regions(X=X, y=y, clf=clf)
plt.show()
```

nu: 訓練誤差の上限
gamma: マージンを超えるサンプルの割合



nu: 訓練誤差の上限

gamma: マージンを超えるサンプルの割合



nu : 0.001
gamma: 0.1

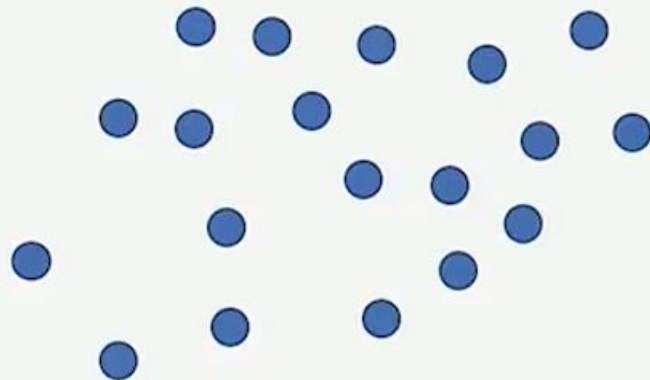
nu : 0.5
gamma: 0.01

Local Outlier Factor



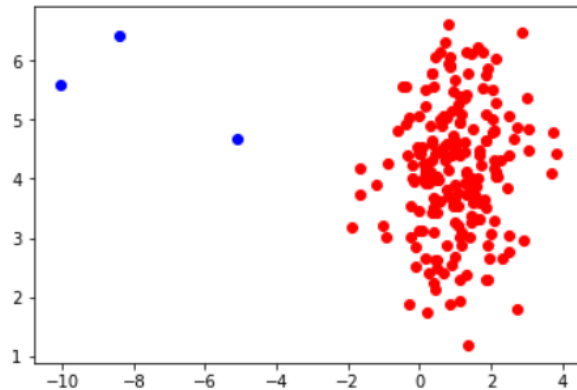
これはOutlierにしたい

これはOutlierにしたくない



他の近くの点との
密度を考慮すれば良さそう

```
%matplotlib inline
from sklearn import datasets
import matplotlib.pyplot as plt
X_train_inlier, _ = datasets.make_blobs(centers=1, random_state=3, cluster_std=1, n_samples=200)
X_train_outlier, _ = datasets.make_blobs(centers=1, random_state=7, cluster_std=2, n_samples=3)
plt.scatter(X_train_inlier[:,0], X_train_inlier[:,1], c='red')
plt.scatter(X_train_outlier[:,0], X_train_outlier[:,1], c='blue')
plt.show()
```



```
import numpy as np
from sklearn.neighbors import LocalOutlierFactor
X = np.r_[X_train_inlier, X_train_outlier]
clf = LocalOutlierFactor(n_neighbors = 10)
y_pred = clf.fit_predict(X)
plt.figure()
plt.scatter(X[y_pred==1,0], X[y_pred==1,1], c='green')
plt.scatter(X[y_pred==-1,0], X[y_pred==-1,1], c='black')
plt.show()
```

