KHAI PHÁ DỮ LIỆU

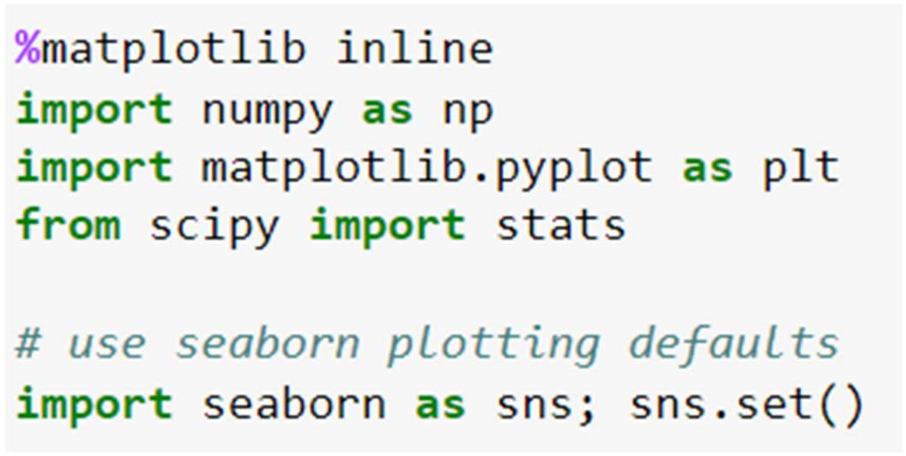
Trường Đại học Nha Trang
Khoa Công nghệ thông tin
Bộ môn Hệ thống thông tin
Giáo viên: TS.Nguyễn Khắc Cường

CHỦ ĐỀ 4

PHÂN LỚP
(Thực hành SVM)

Thực hành SVM

- Multi-class SVM?
  - Bản chất của SVM là chỉ thực hiện Binary classification
  - Mở rộng thành Multi-class SVM theo hai kiểu:
    - One-to-One
    - One-to-Rest
- Tìm hiểu các bước thực hiện SVM binary classification

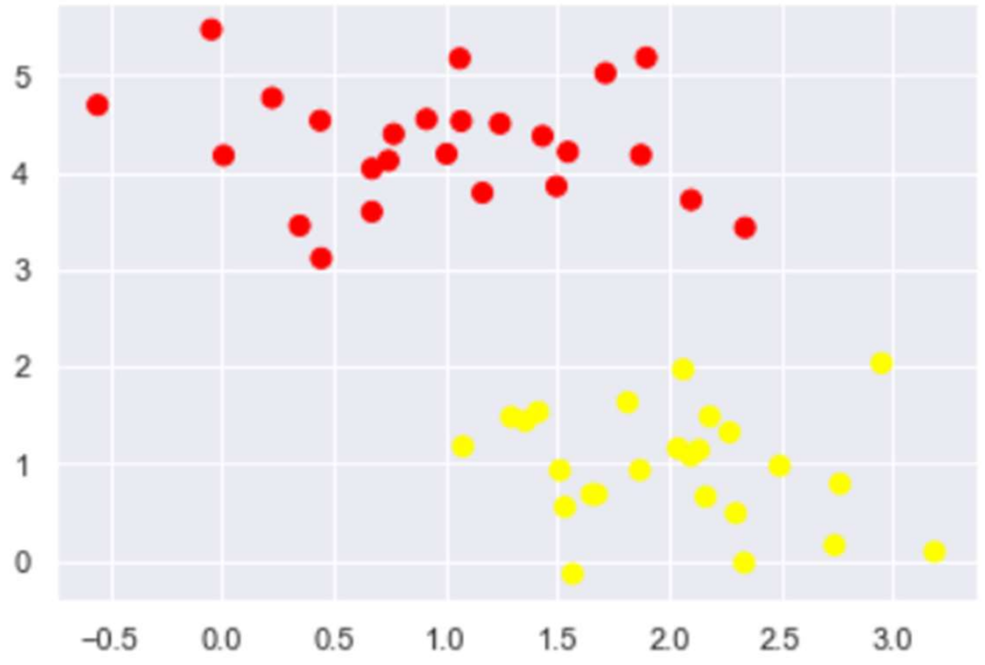


```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats

# use seaborn plotting defaults
import seaborn as sns; sns.set()
```

Thực hành SVM

- Tìm hiểu các bước thực hiện SVM binary classification

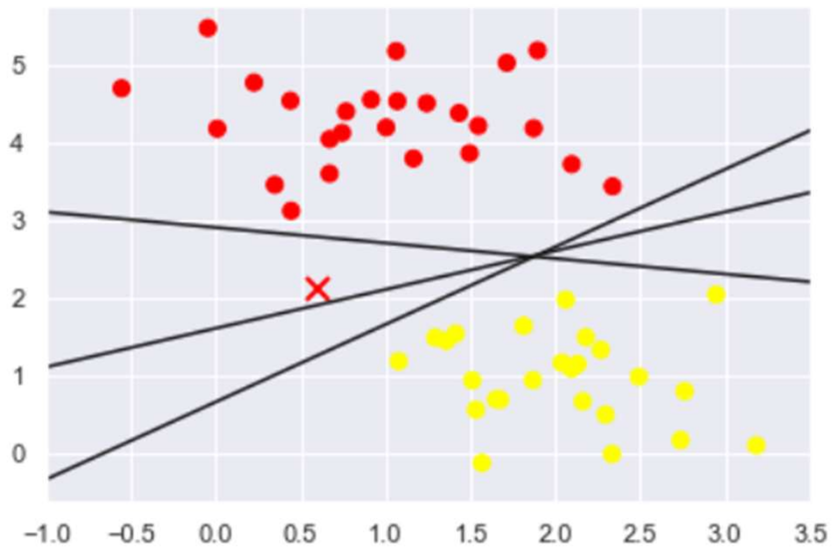
```
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples=50, centers=2, random_state=0, cluster_std=0.60)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='autumn');
```



Thực hành SVM

- Tìm hiểu các bước thực hiện SVM binary classification

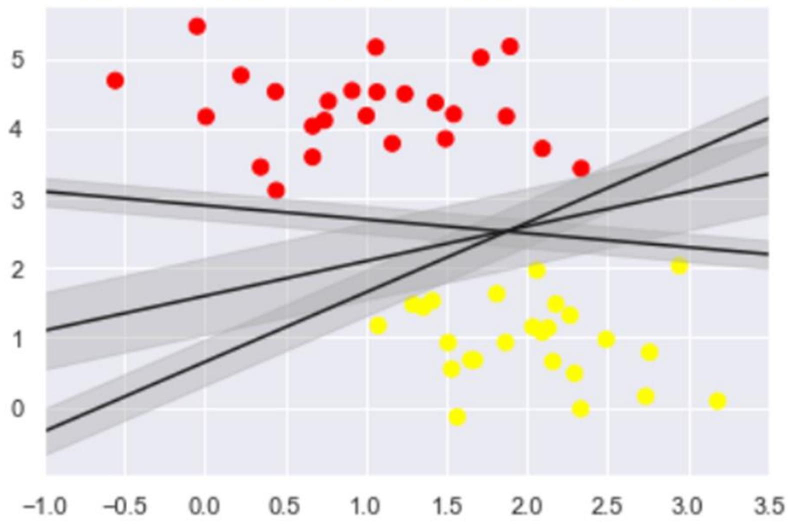
```
xfit = np.linspace(-1, 3.5)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='autumn')
plt.plot([0.6], [2.1], 'x', color='red', markeredgewidth=2, markersize=10)

for m, b in [(1, 0.65), (0.5, 1.6), (-0.2, 2.9)]:
    plt.plot(xfit, m * xfit + b, '-k')

plt.xlim(-1, 3.5);
```



Thực hành SVM

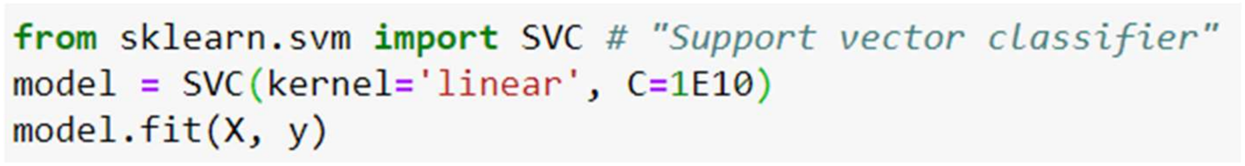
- Tìm hiểu các bước thực hiện SVM binary classification

```
xfit = np.linspace(-1, 3.5)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='autumn')

for m, b, d in [(1, 0.65, 0.33), (0.5, 1.6, 0.55), (-0.2, 2.9, 0.2)]:
    yfit = m * xfit + b
    plt.plot(xfit, yfit, '-k')
    plt.fill_between(xfit, yfit - d, yfit + d, edgecolor='none',
                     color='#AAAAAA', alpha=0.4)

plt.xlim(-1, 3.5);
```

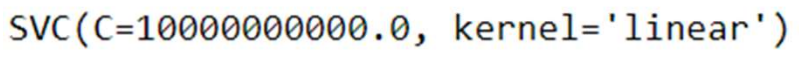


Thực hành SVM

- Tìm hiểu các bước thực hiện SVM binary classification



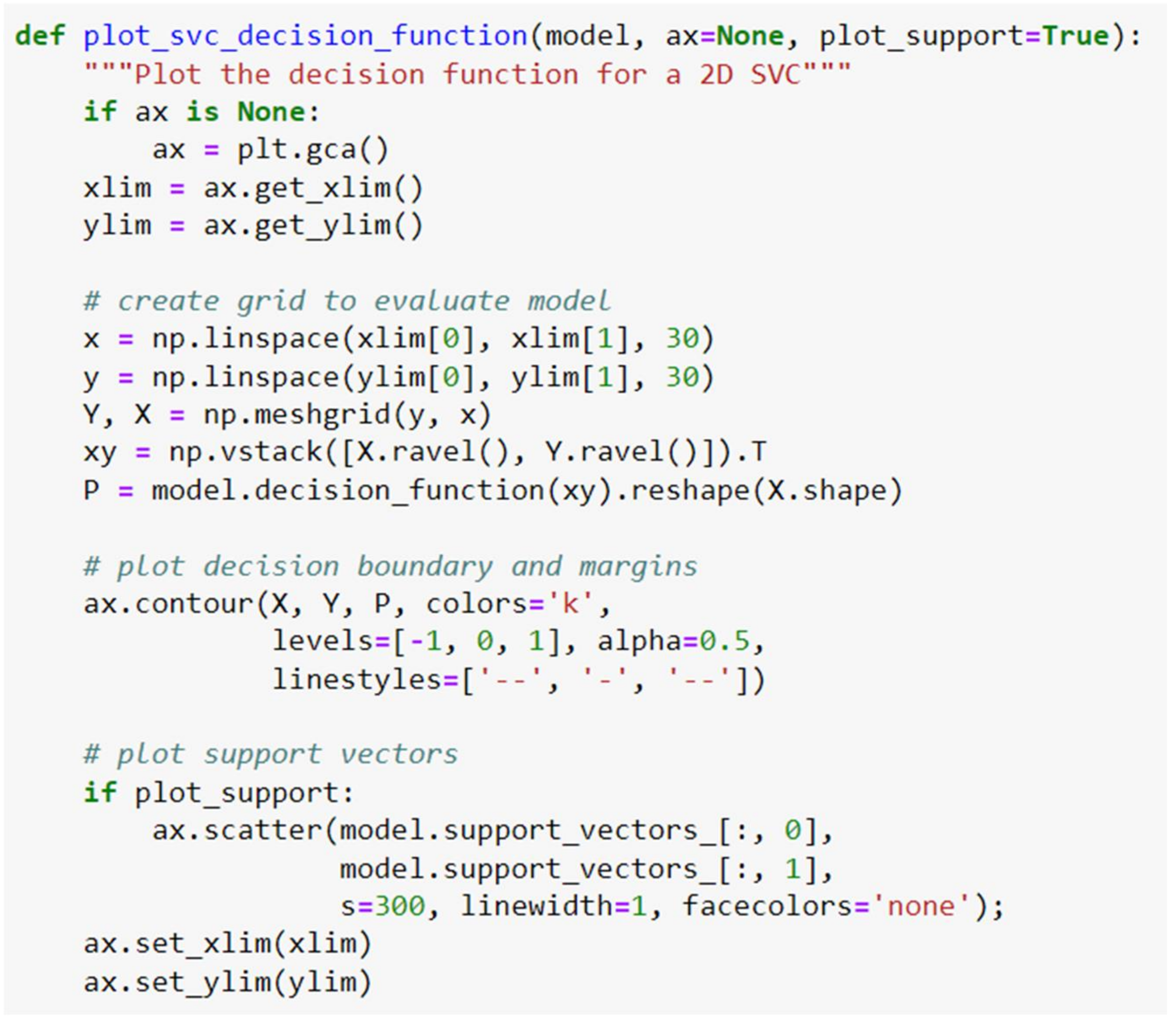
```python
from sklearn.svm import SVC # "Support vector classifier"
model = SVC(kernel='linear', C=1E10)
model.fit(X, y)
```



```
SVC(C=10000000000.0, kernel='linear')
```

Thực hành SVM

- Tìm hiểu các bước thực hiện SVM binary classification

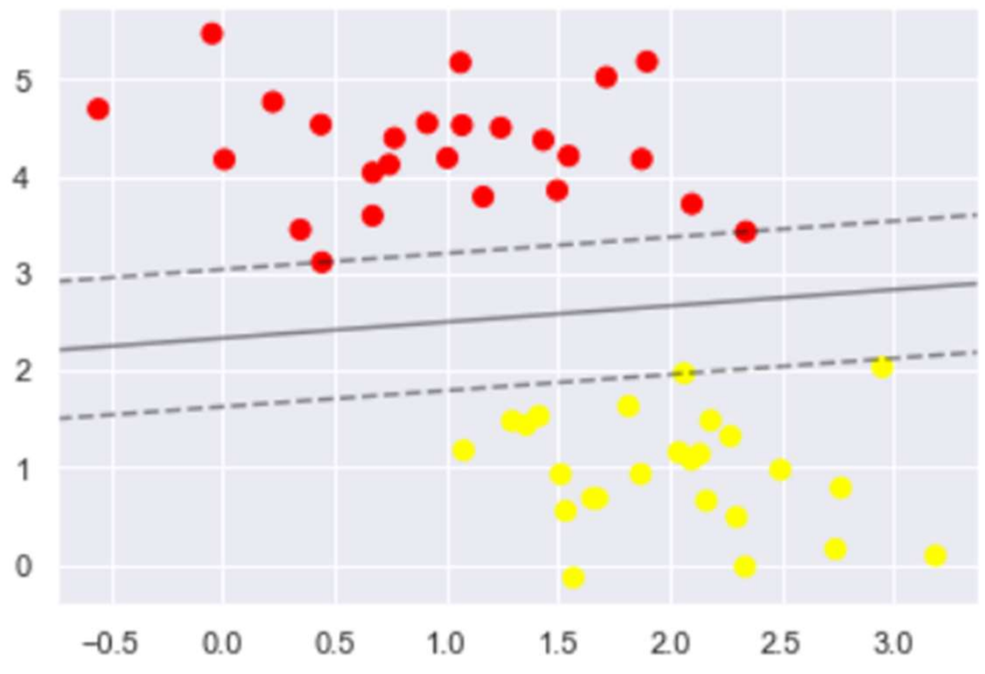


```python
def plot_svc_decision_function(model, ax=None, plot_support=True):
    """Plot the decision function for a 2D SVC"""
    if ax is None:
        ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    # create grid to evaluate model
    x = np.linspace(xlim[0], xlim[1], 30)
    y = np.linspace(ylim[0], ylim[1], 30)
    Y, X = np.meshgrid(y, x)
    xy = np.vstack([X.ravel(), Y.ravel()]).T
    P = model.decision_function(xy).reshape(X.shape)

    # plot decision boundary and margins
    ax.contour(X, Y, P, colors='k',
               levels=[-1, 0, 1], alpha=0.5,
               linestyles=['--', '-', '--'])

    # plot support vectors
    if plot_support:
        ax.scatter(model.support_vectors_[:, 0],
                   model.support_vectors_[:, 1],
                   s=300, linewidth=1, facecolors='none');
    ax.set_xlim(xlim)
    ax.set_ylim(ylim)
```

Thực hành SVM

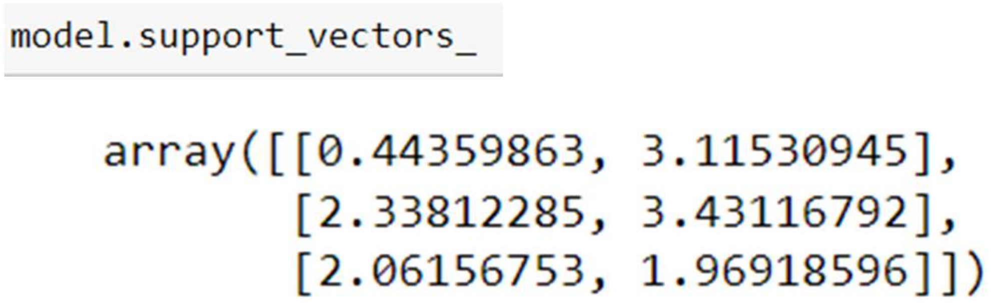
- Tìm hiểu các bước thực hiện SVM binary classification

```
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='autumn')
plot_svc_decision_function(model);
```



Thực hành SVM

- Tìm hiểu các bước thực hiện SVM binary classification

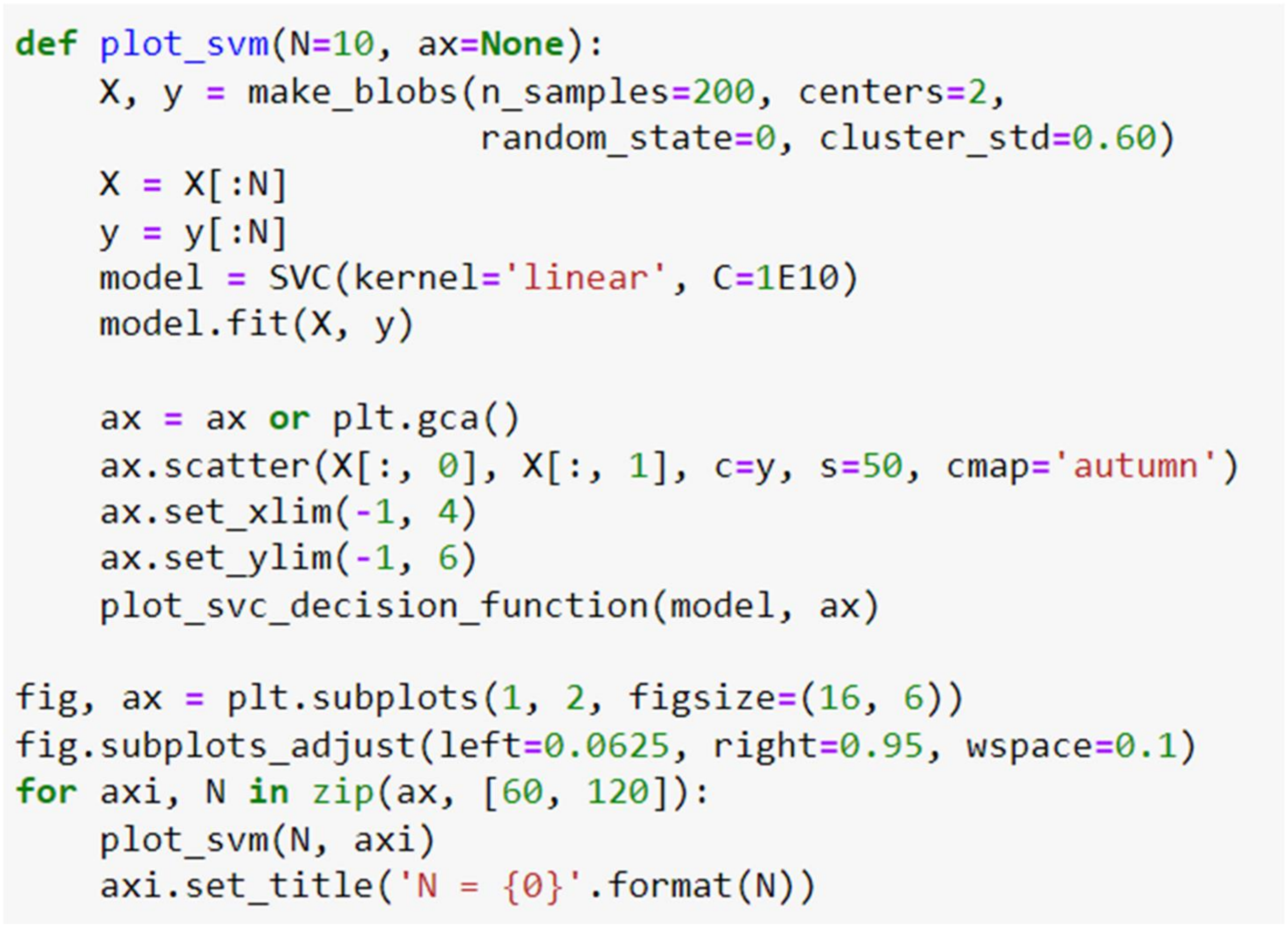


```
model.support_vectors_
```

```
array([[0.44359863, 3.11530945],
       [2.33812285, 3.43116792],
       [2.06156753, 1.96918596]])
```

Thực hành SVM

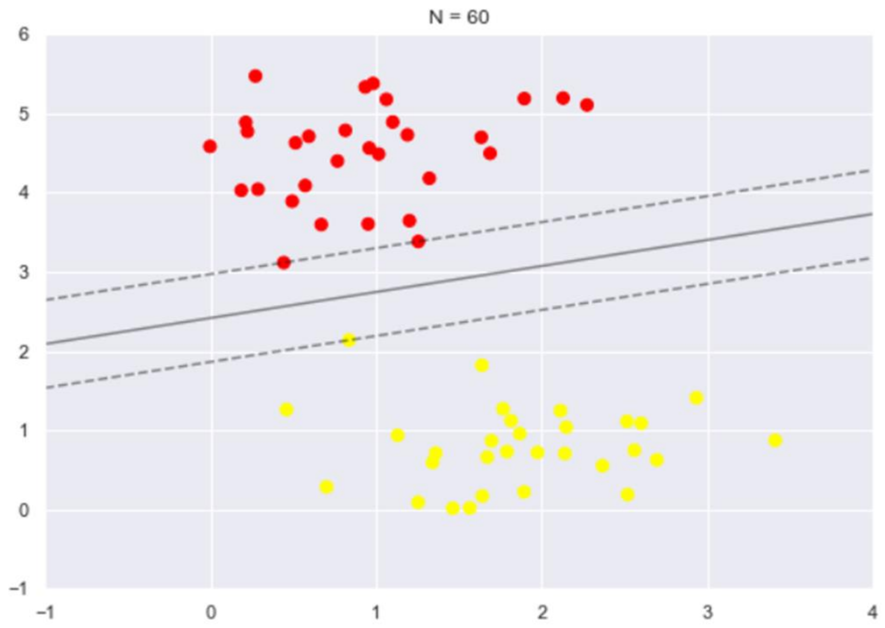
- Tìm hiểu các bước thực hiện SVM binary classification

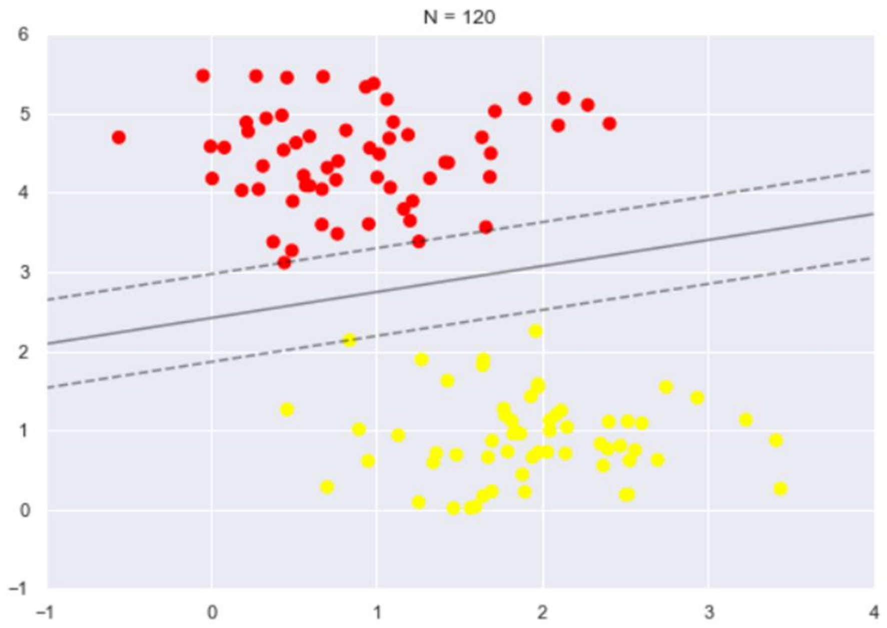


```python
def plot_svm(N=10, ax=None):
    X, y = make_blobs(n_samples=200, centers=2,
                      random_state=0, cluster_std=0.60)
    X = X[:N]
    y = y[:N]
    model = SVC(kernel='linear', C=1E10)
    model.fit(X, y)

    ax = ax or plt.gca()
    ax.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='autumn')
    ax.set_xlim(-1, 4)
    ax.set_ylim(-1, 6)
    plot_svc_decision_function(model, ax)

fig, ax = plt.subplots(1, 2, figsize=(16, 6))
fig.subplots_adjust(left=0.0625, right=0.95, wspace=0.1)
for axi, N in zip(ax, [60, 120]):
    plot_svm(N, axi)
    axi.set_title('N = {0}'.format(N))
```

Thực hành SVM

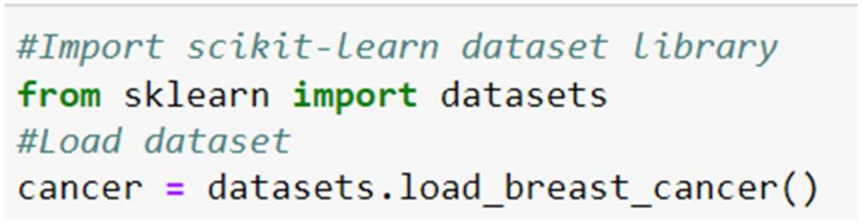
- Tìm hiểu các bước thực hiện SVM binary classification



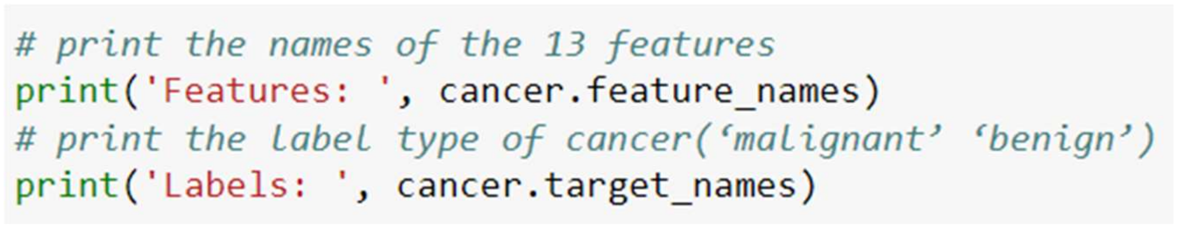


Thực hành SVM

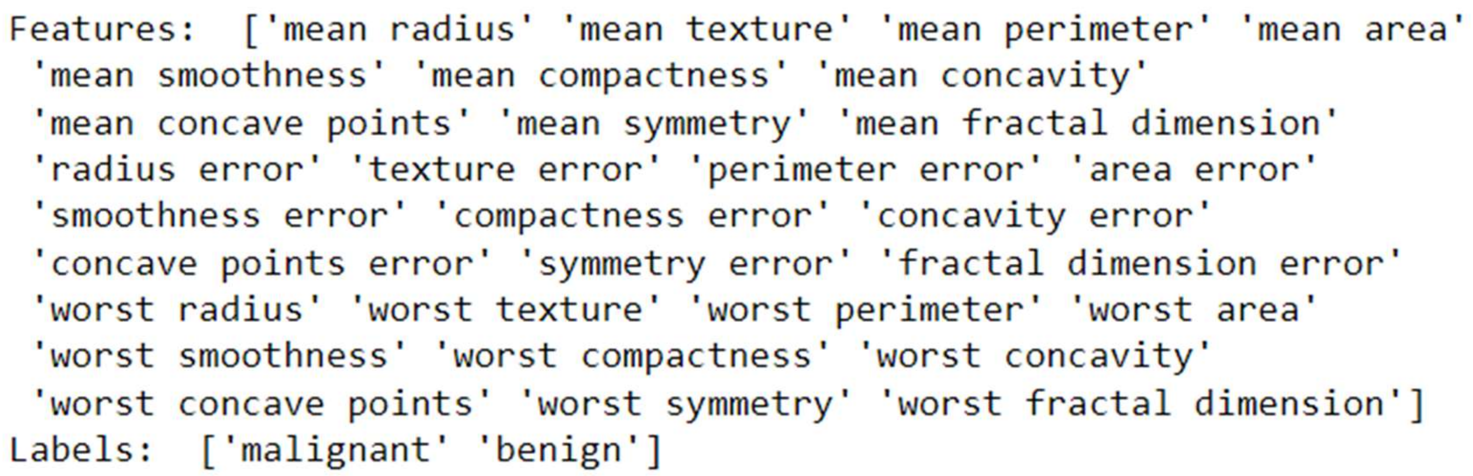
- Ứng dụng thực tế của SVM binary classification



```
#Import scikit-learn dataset library
from sklearn import datasets
#Load dataset
cancer = datasets.load_breast_cancer()
```



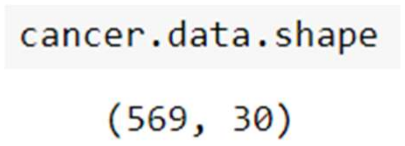
```
# print the names of the 13 features
print('Features: ', cancer.feature_names)
# print the label type of cancer(‘malignant’ ‘benign’)
print('Labels: ', cancer.target_names)
```



```
Features:  ['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
Labels:  ['malignant' 'benign']
```

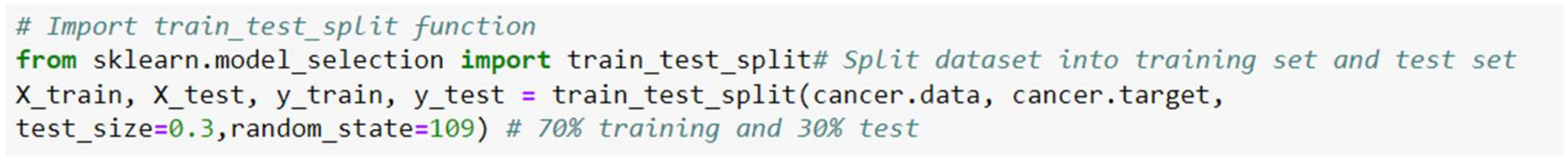
Thực hành SVM

- Ứng dụng thực tế của SVM binary classification

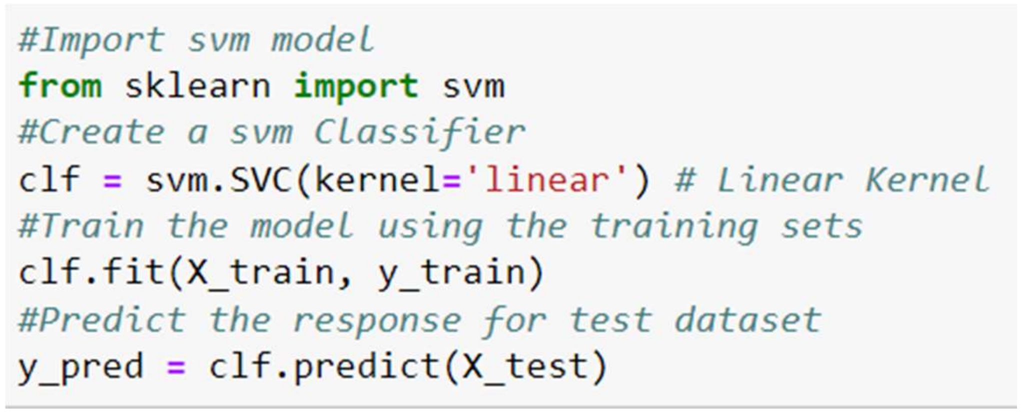


```
cancer.data.shape
```

```
    (569, 30)
```



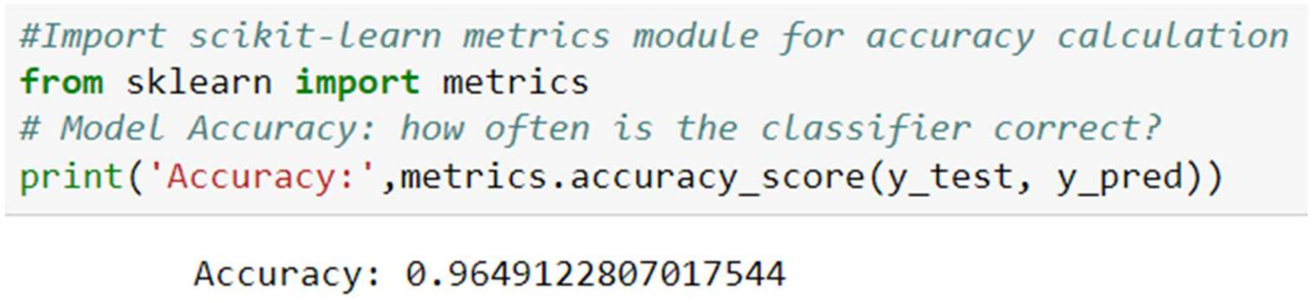
```
# Import train_test_split function
from sklearn.model_selection import train_test_split# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target,
test_size=0.3,random_state=109) # 70% training and 30% test
```



```
#Import svm model
from sklearn import svm
#Create a svm Classifier
clf = svm.SVC(kernel='linear') # Linear Kernel
#Train the model using the training sets
clf.fit(X_train, y_train)
#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

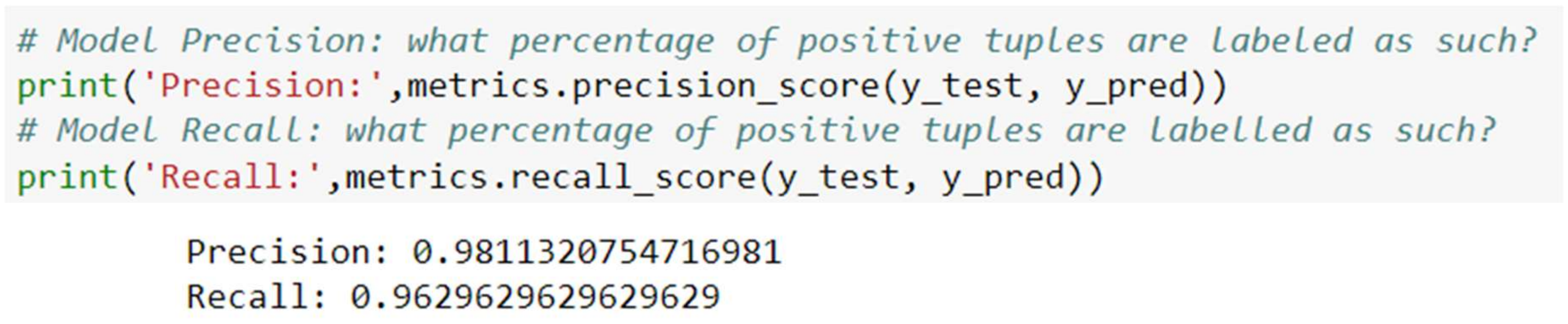
Thực hành SVM

- Ứng dụng thực tế của SVM binary classification



```python
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
# Model Accuracy: how often is the classifier correct?
print('Accuracy:',metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9649122807017544
```



```python
# Model Precision: what percentage of positive tuples are labeled as such?
print('Precision:',metrics.precision_score(y_test, y_pred))
# Model Recall: what percentage of positive tuples are labelled as such?
print('Recall:',metrics.recall_score(y_test, y_pred))
```

```
Precision: 0.9811320754716981
Recall: 0.9629629629629629
```

Thực hành SVM

Q / A