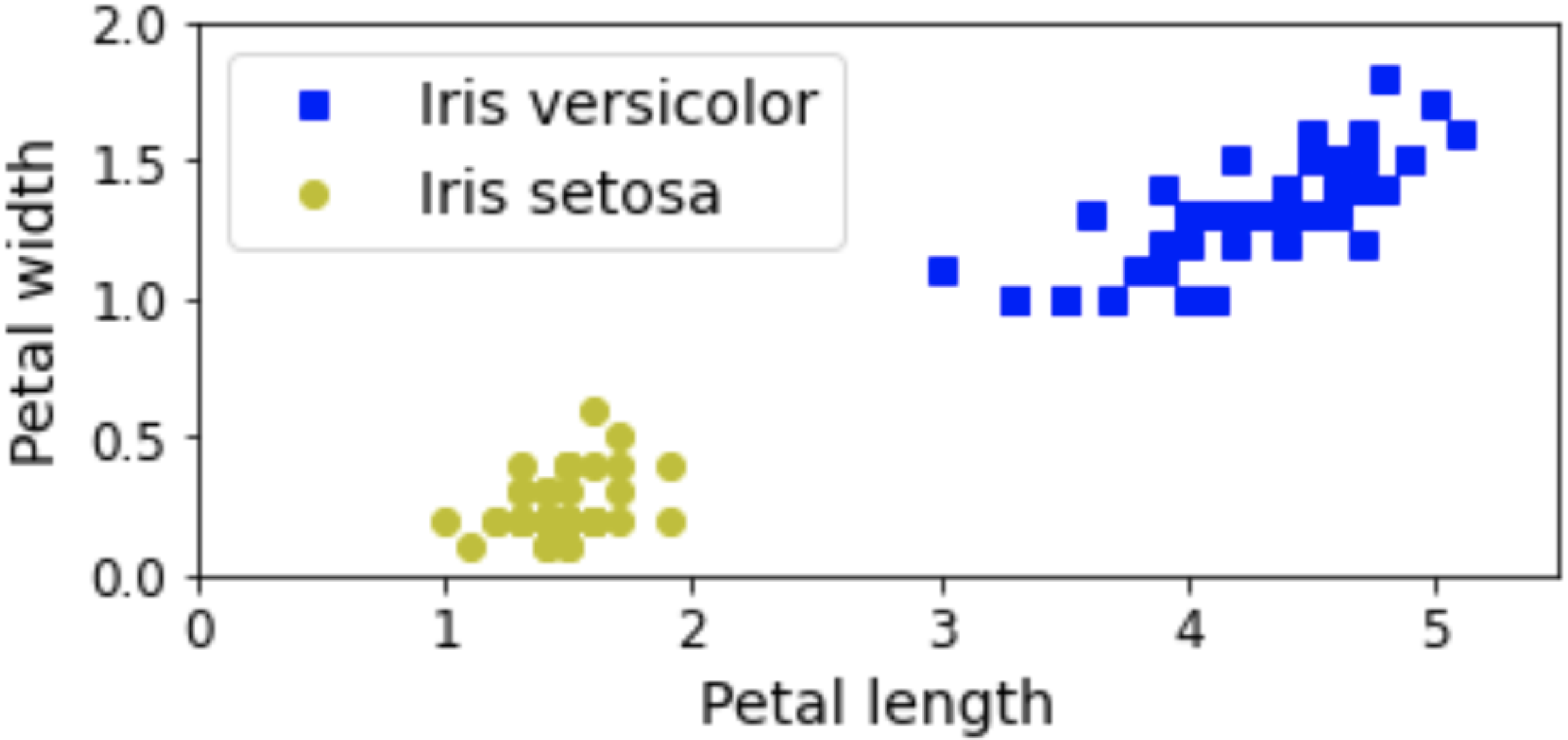
CSY2082 Introduction to Artificial Intelligence

Support Vector Machines

Support Vector Machine

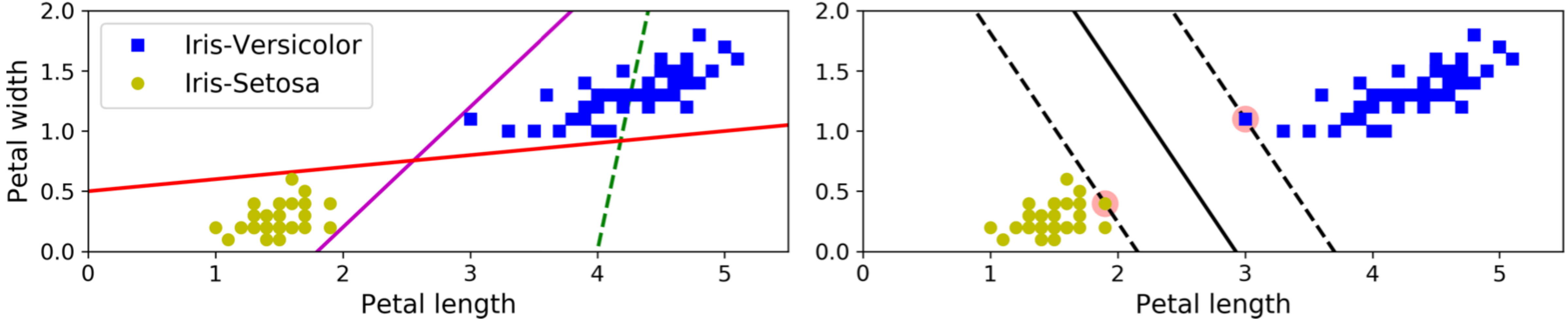
- A **Support Vector Machine (SVM)** is a very powerful and versatile Machine Learning model, capable of performing linear or nonlinear **classification**, **regression**, and even **outlier detection**.
- SVMs are particularly well suited for classification of complex but small- or medium-sized datasets.

Support Vector Machine



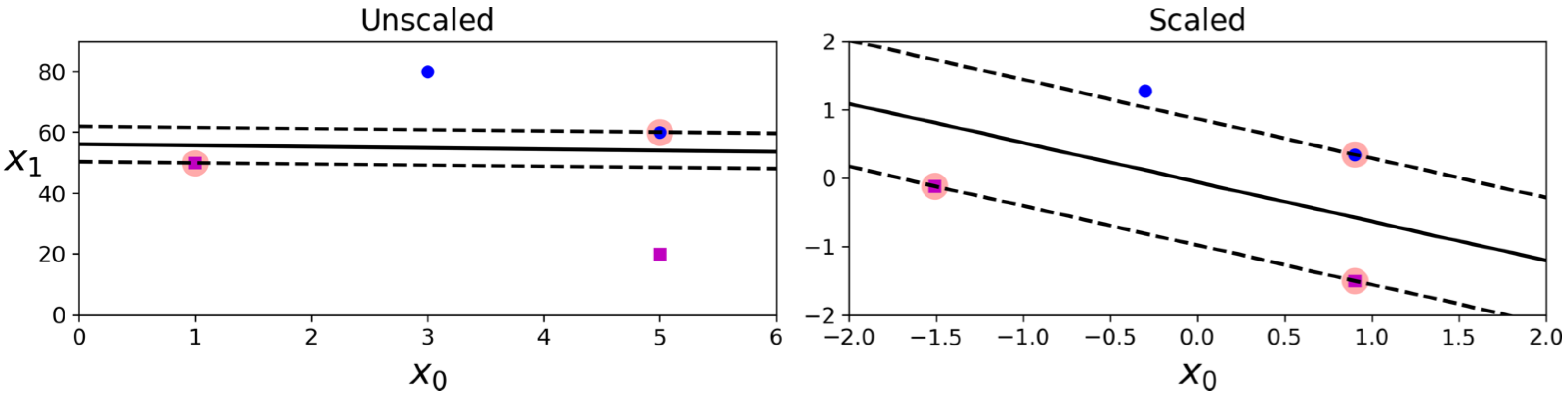
Linear SVM Classification

- Solid lines OK but decision boundaries very close to instances that these models will probably not perform as well on new instances.
- SVM classifier fitting the widest possible street (represented by the parallel dashed lines) between the classes. This is called *large margin classification*.
- Decision boundary is fully determined (or “supported”) by the instances located on the edge of the street. These instances are called the support vectors (circled)



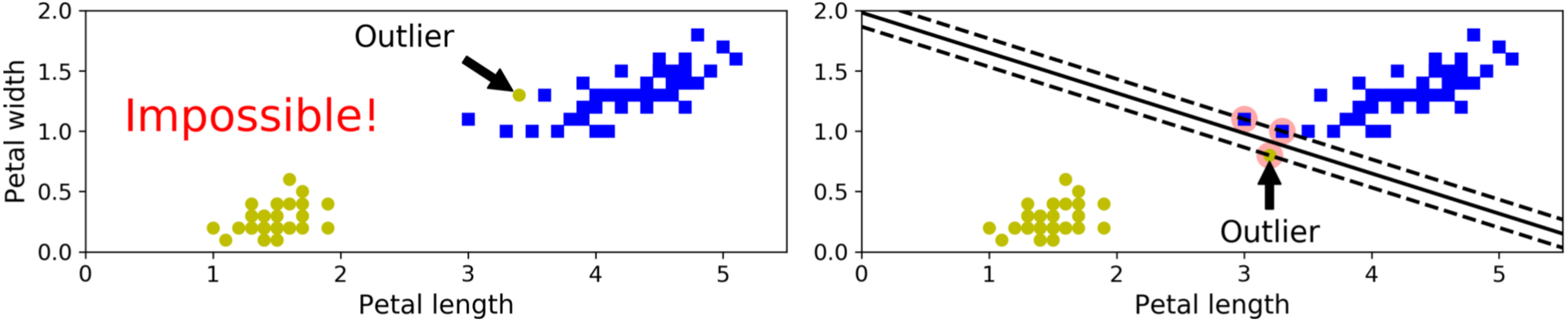
Linear SVM Classification

- SVMs are sensitive to the feature scales
- on the left plot, the vertical scale is much larger than the horizontal scale, so the widest possible street is close to horizontal.
- After feature scaling (e.g., using Scikit-Learn's *StandardScaler*), the decision boundary looks much better (on the right plot).



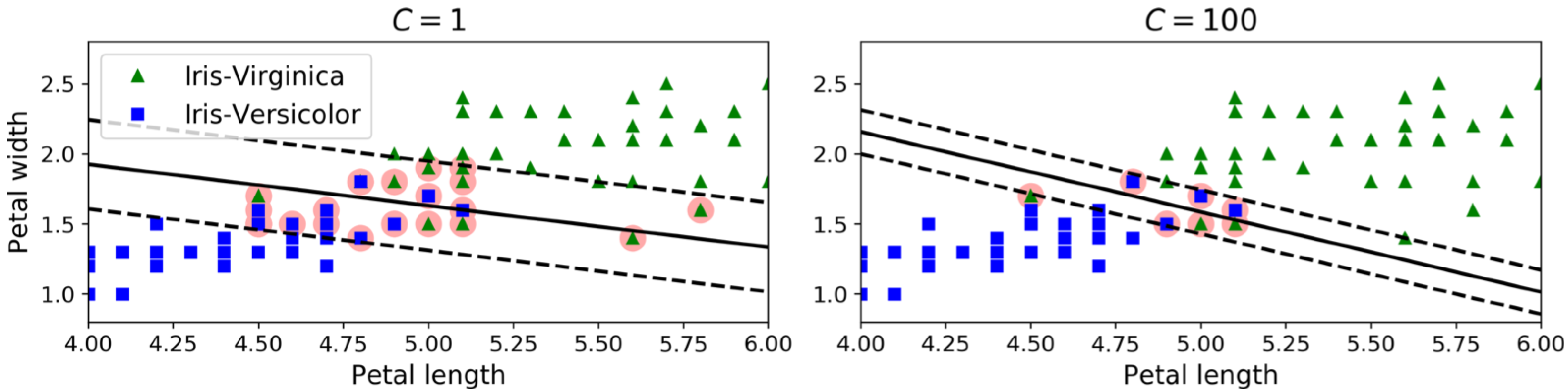
Hard Margin Classification

- If we strictly impose that all instances be off the street and on the right side, this is called hard margin classification.
- Two main issues :
  - it only works if the data is linearly separable
  - it is quite sensitive to outliers.



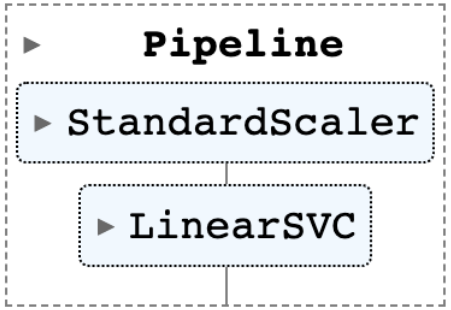
Soft Margin Classification

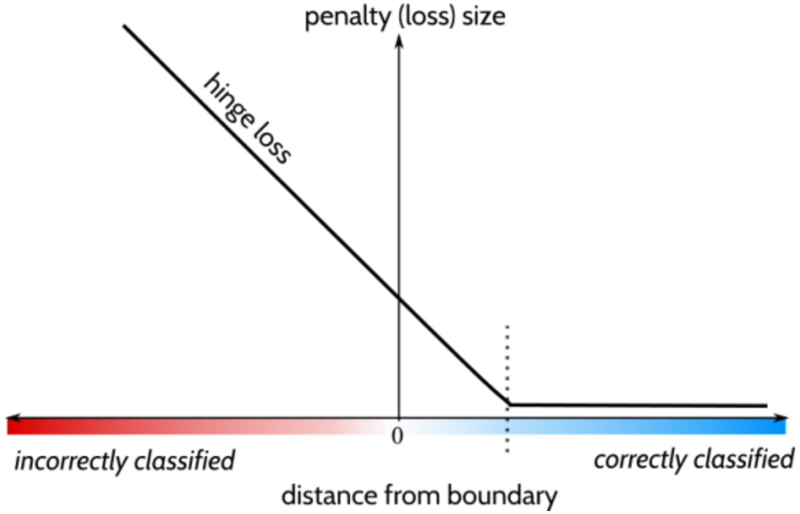
- To avoid these issues it is preferable to use a more flexible model.
  - The objective is to find a good balance between keeping the street as large as possible and limiting the margin violations. This is called soft margin classification.
- In Scikit-Learn's SVM classes, you can control this balance using the C hyperparameter: a smaller C value leads to a wider street but more margin violations.



LinearSVC

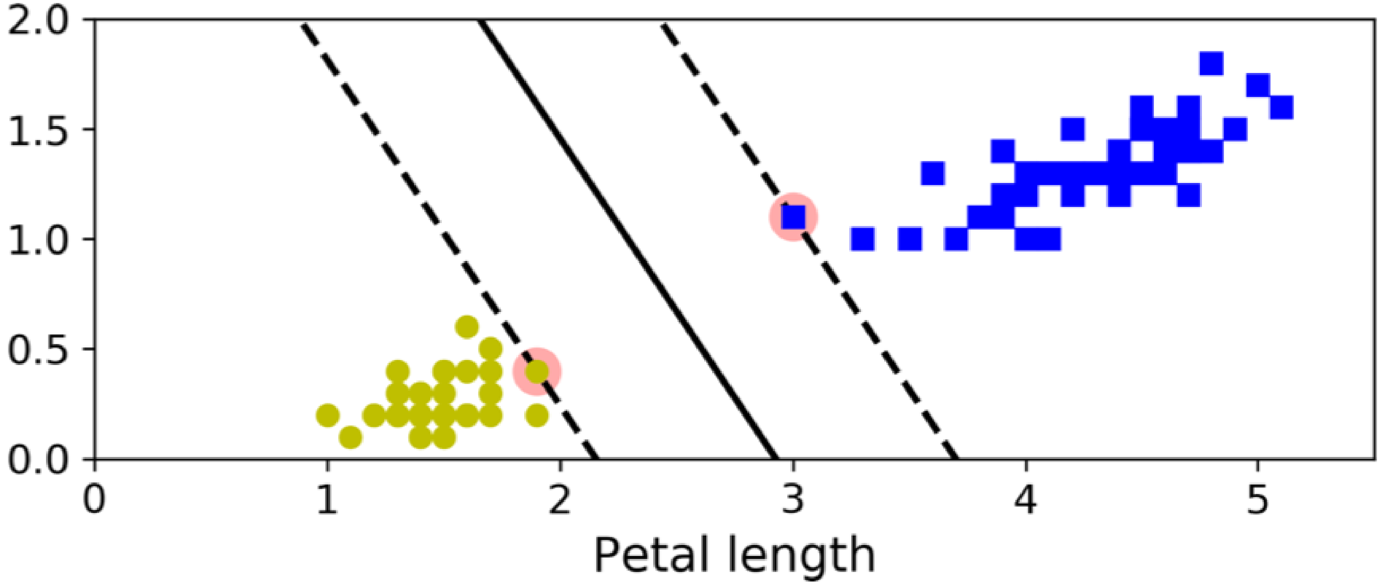
```python
import numpy as np
from sklearn.datasets import load_iris
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import LinearSVC

iris = load_iris(as_frame=True)
X = iris.data[["petal length (cm)", "petal width (cm)"]].values
y = (iris.target == 2)  # Iris virginica

svm_clf = make_pipeline(StandardScaler(),
                        LinearSVC(C=1))
svm_clf.fit(X, y)
```

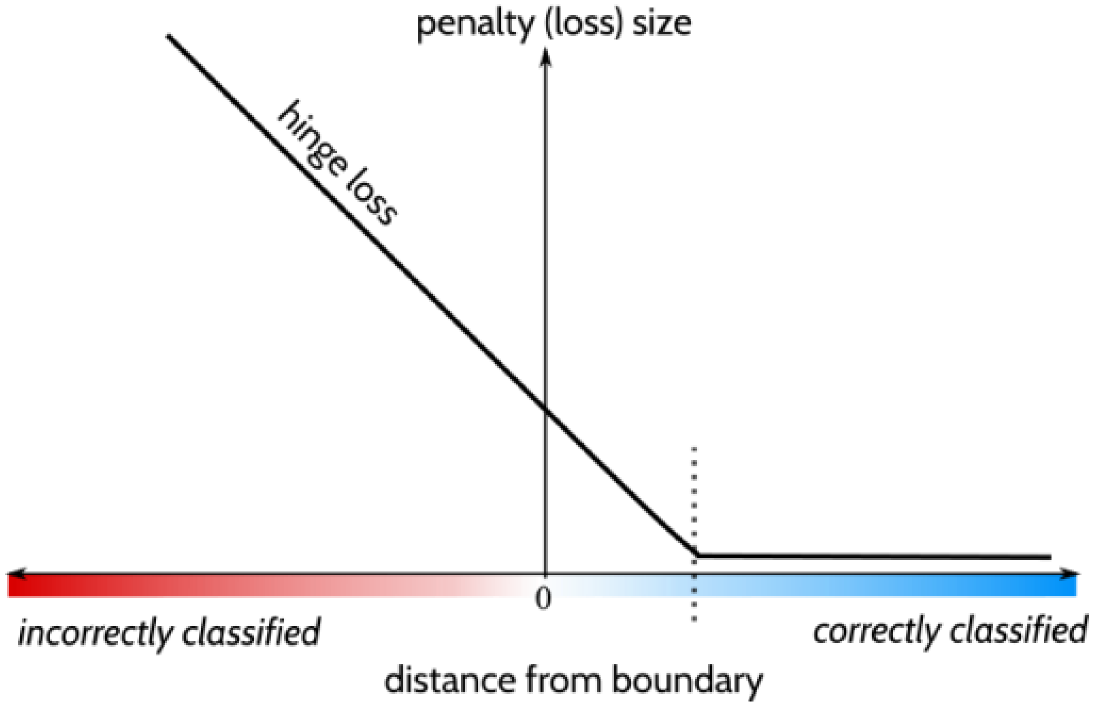




Hinge loss

- “Find me a line that produces the widest street between the 2 classes”.





LinearSVC

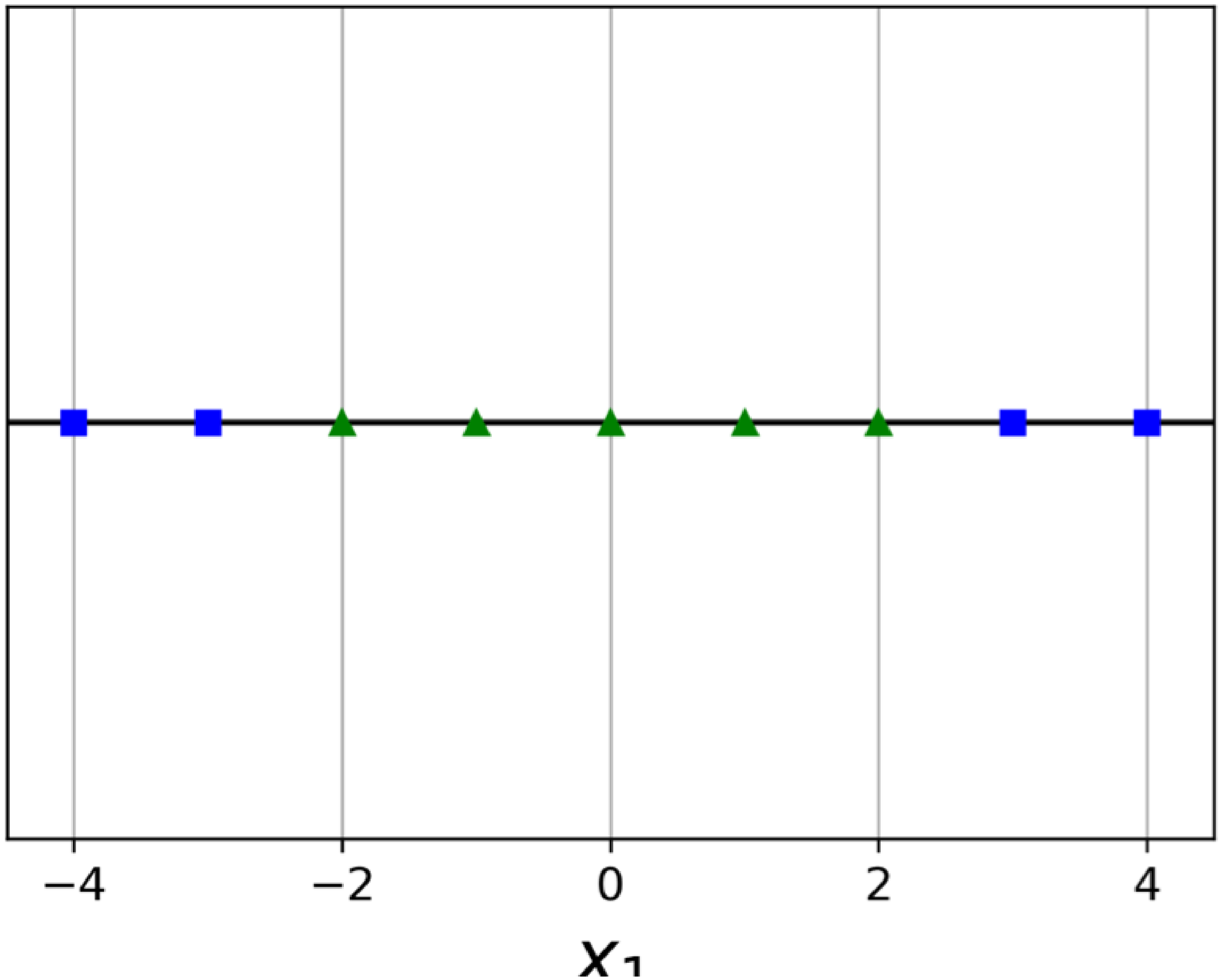
```
X_new = [[5.5, 1.7], [5.0, 1.5]]
svm_clf.predict(X_new)
```

```
array([ True, False])
```

```
svm_clf.decision_function(X_new)
```

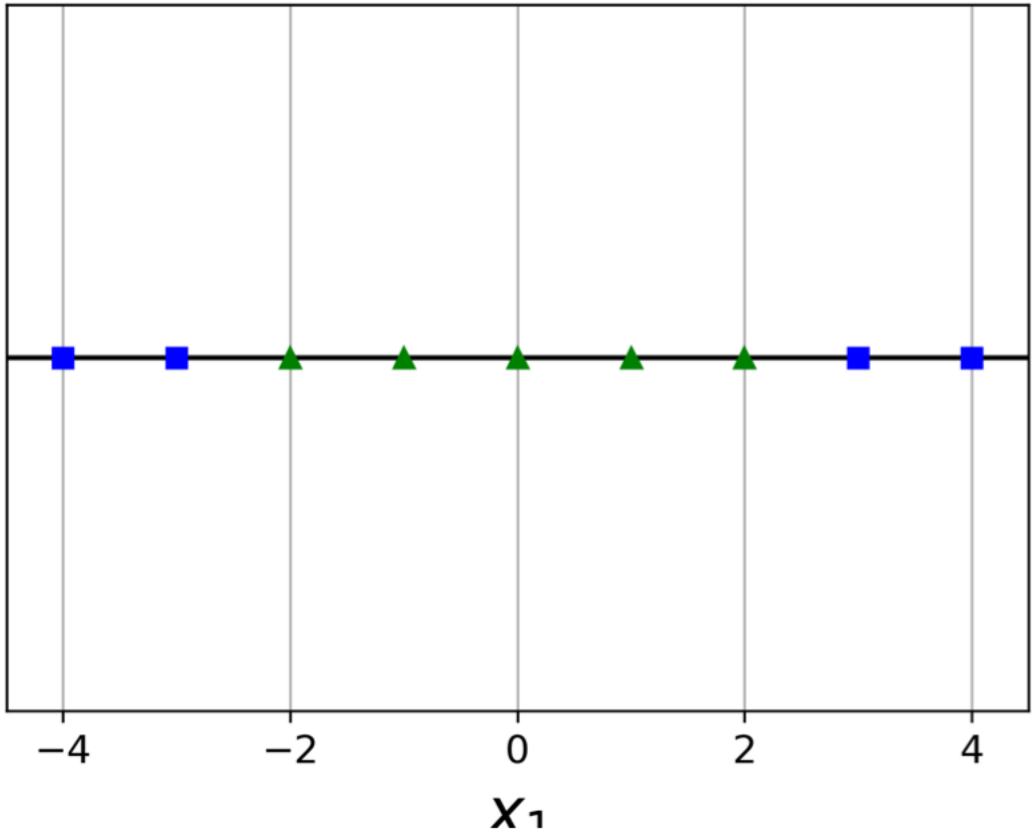
```
array([ 0.66162545, -0.22036792])
```

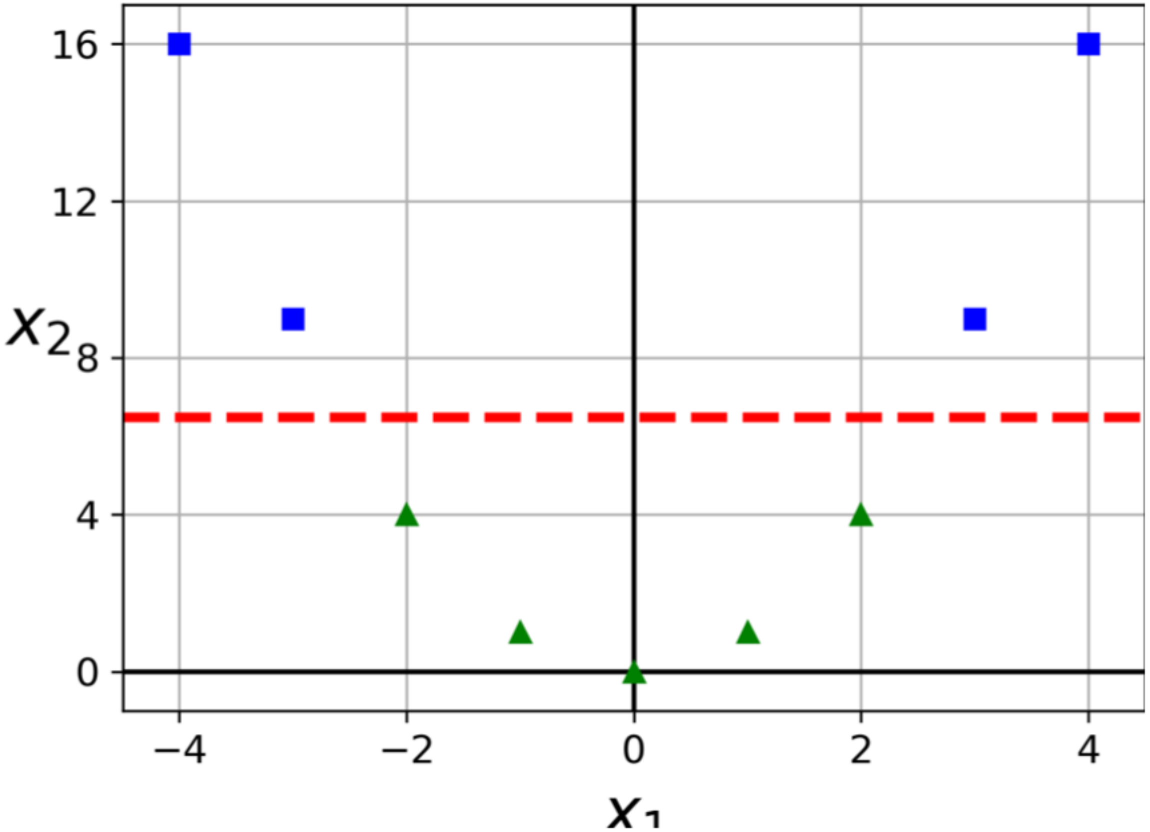
Nonlinear SVM Classification



Nonlinear SVM Classification

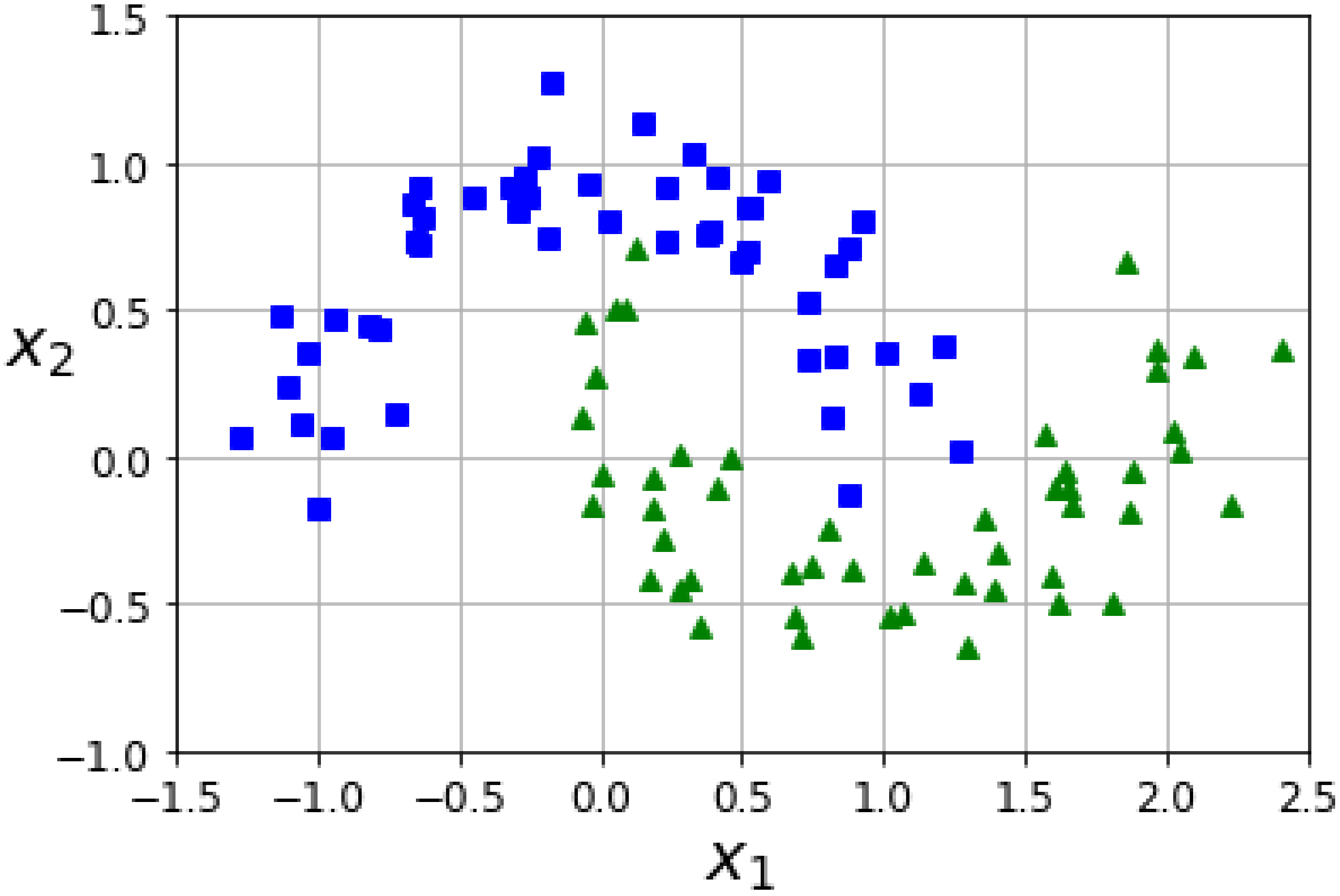
- One approach to handling nonlinear datasets is to add more features, such as polynomial features; in some cases this can result in a linearly separable dataset.





Nonlinear SVM Classification

```
X, y = make_moons(n_samples=100, noise=0.15, random_state=42)
```

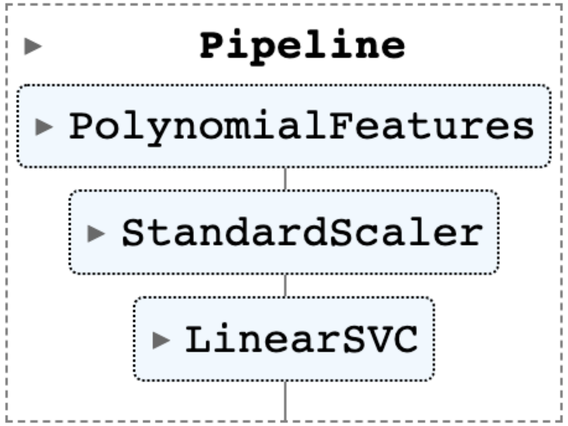


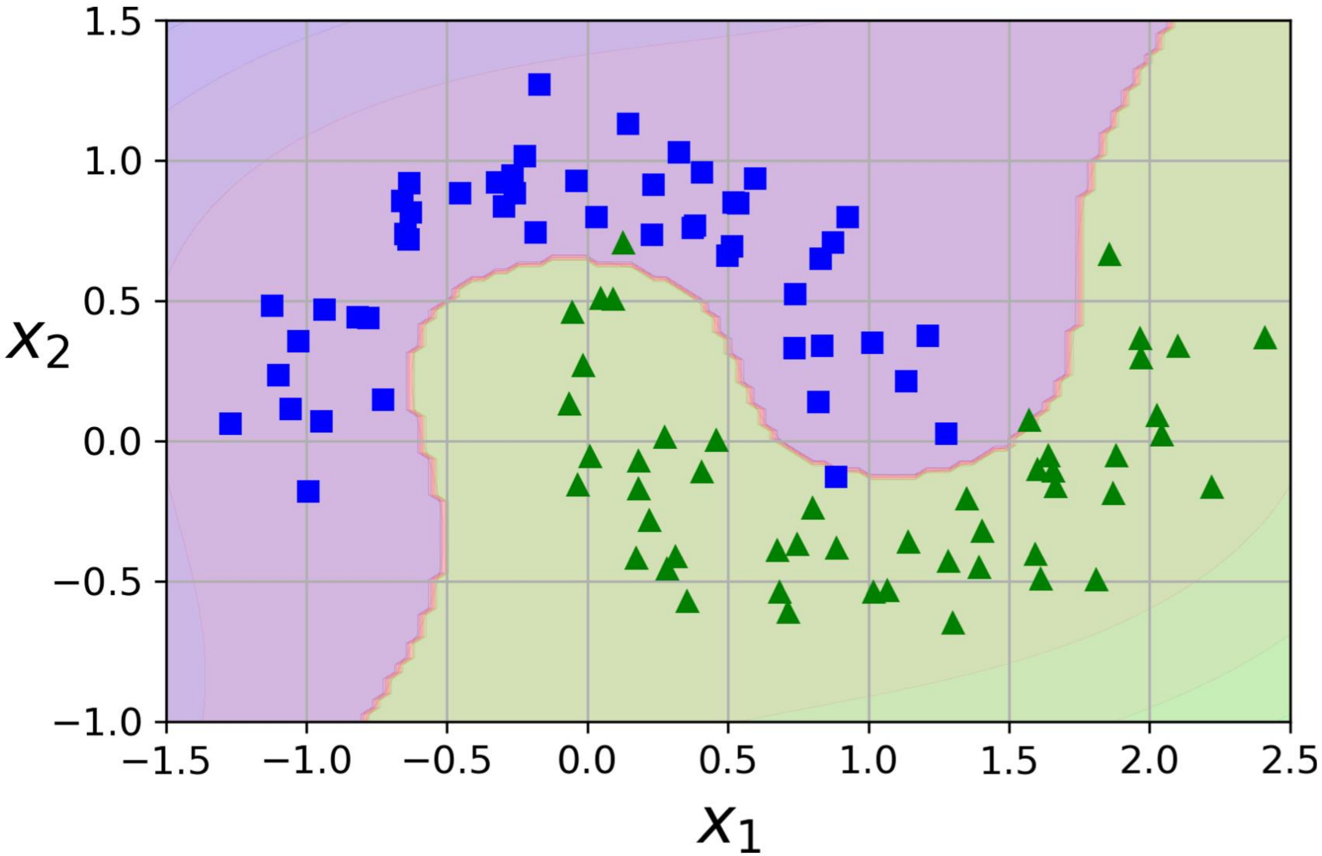
sklearn.datasets.make_moons. Make two interleaving half circles. A simple toy dataset to visualize clustering and classification algorithms.

Nonlinear SVM Classification

- Create a Pipeline containing a PolynomialFeatures transformer.

```
polynomial_svm_clf = make_pipeline(
    PolynomialFeatures(degree=3),
    StandardScaler(),
    LinearSVC(C=10, max_iter=10_000)
)
polynomial_svm_clf.fit(X, y)
```

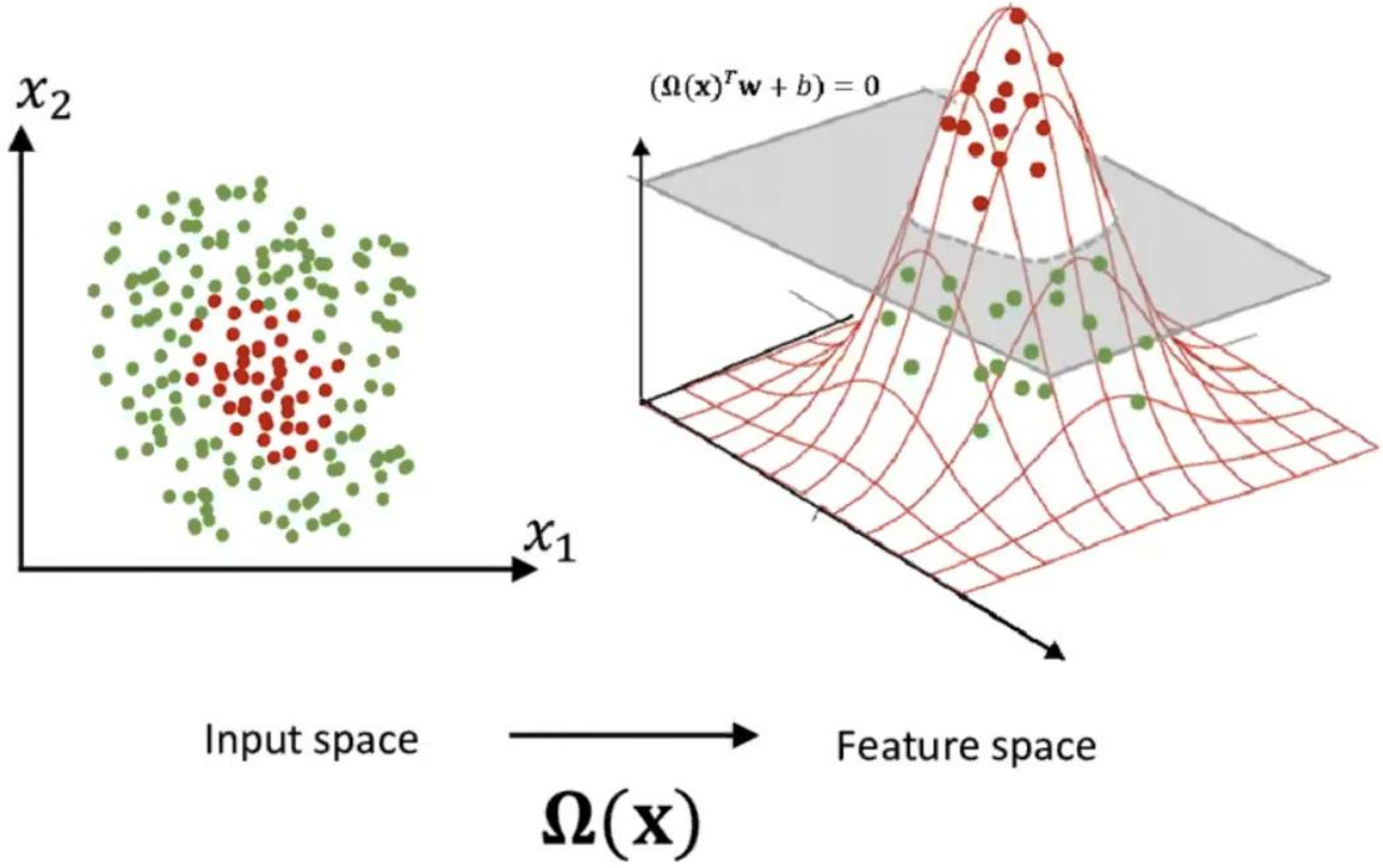




sklearn.datasets.make_moons. Make two interleaving half circles. A simple toy dataset to visualize clustering and classification algorithms.

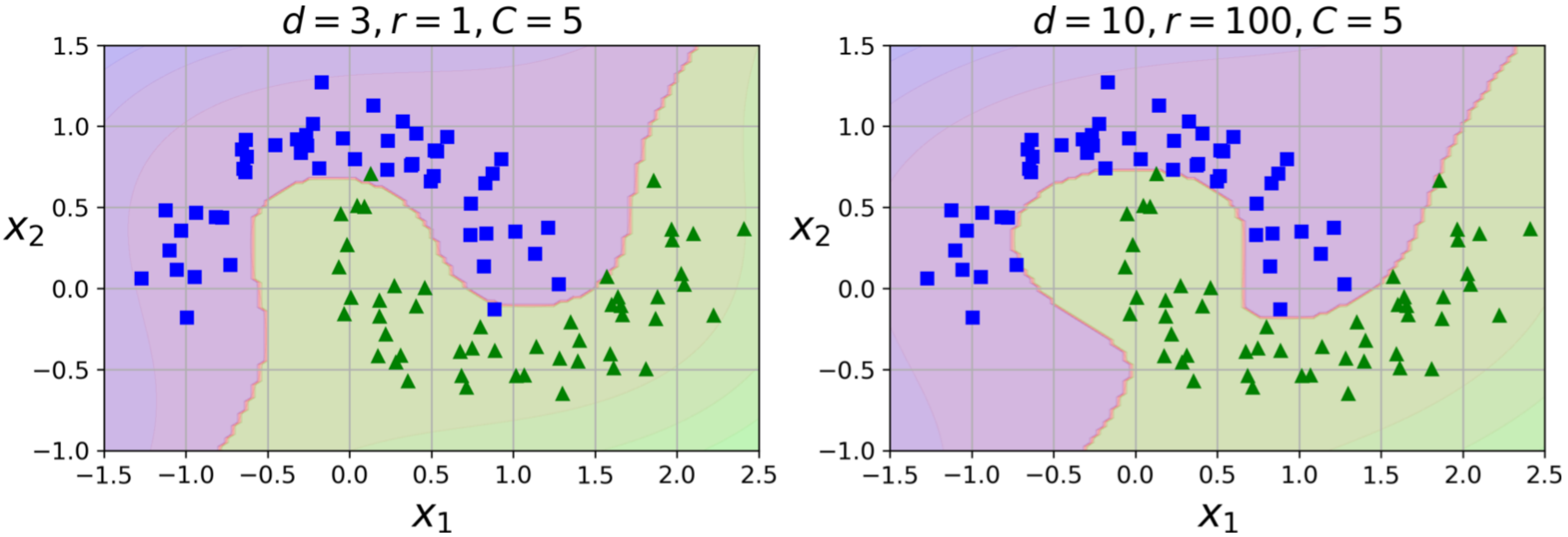
sklearn.svm.SVC

kernel{*‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’*}, *default=’rbf’*



Nonlinear SVM Classification

```python
from sklearn.svm import SVC

poly_kernel_svm_clf = make_pipeline(StandardScaler(),
                                    SVC(kernel="poly", degree=3, coef0=1, C=5))
poly_kernel_svm_clf.fit(X, y)
```



Multi-class??

https://scikit-learn.org/stable/modules/svm.html

- One-vs-Rest (OvR)
- One-vs-One (OvO)

Exercise:

- Use SVM on iris dataset
  - LinearSVC to separate virginica from non-virginica
  - Try non-linear options
  - (optional) expand to 3 class classification.