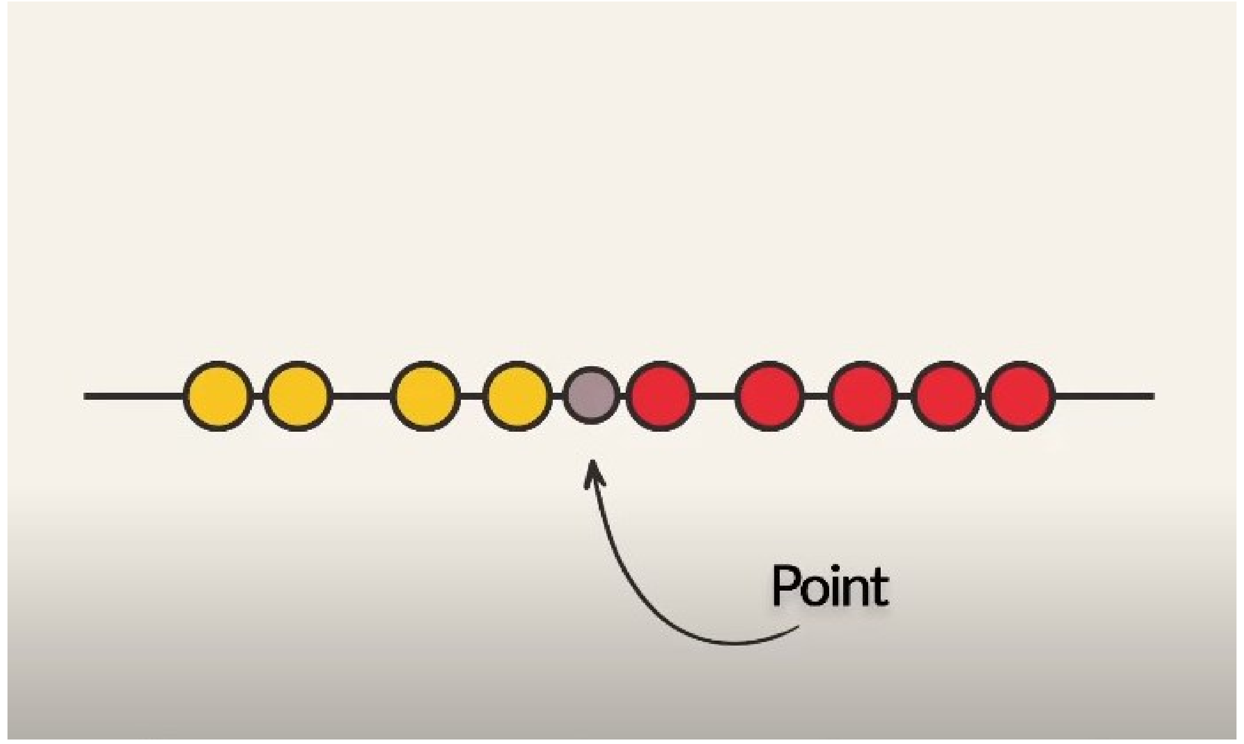
Support Vector Machine

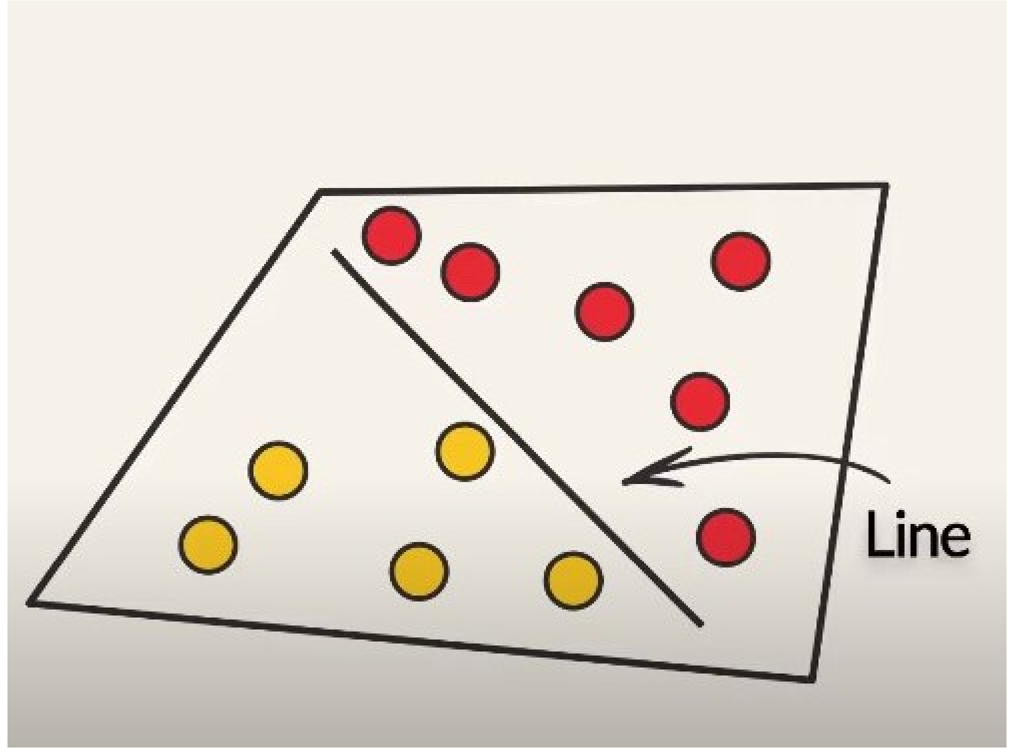
Kanokkorn Pimcharoen

Brain
Code
Camp

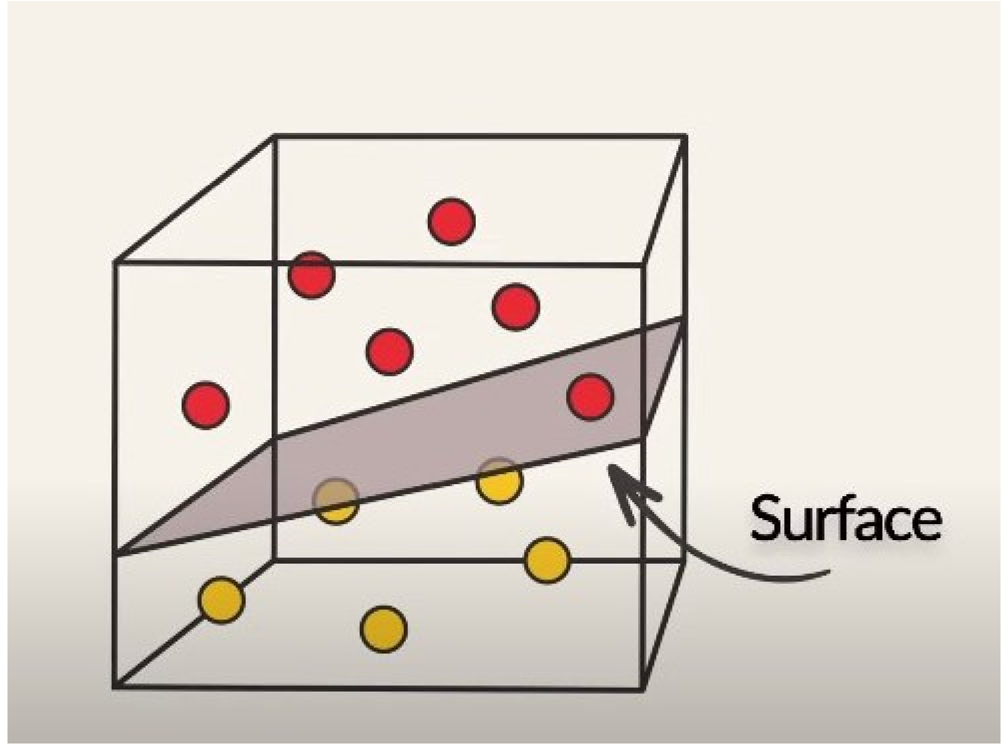
Separating Hyperplanes



1D Feature space



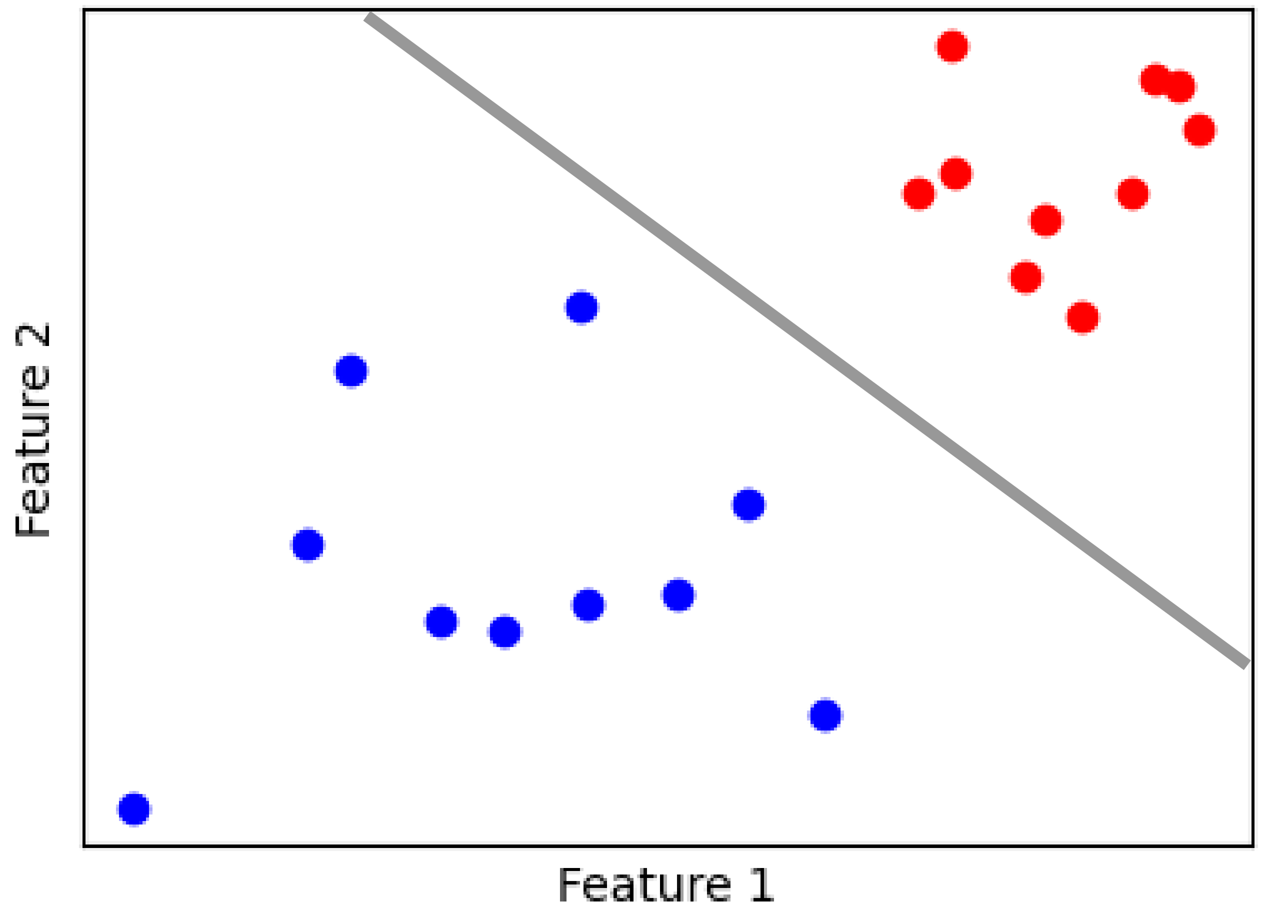
2D Feature space



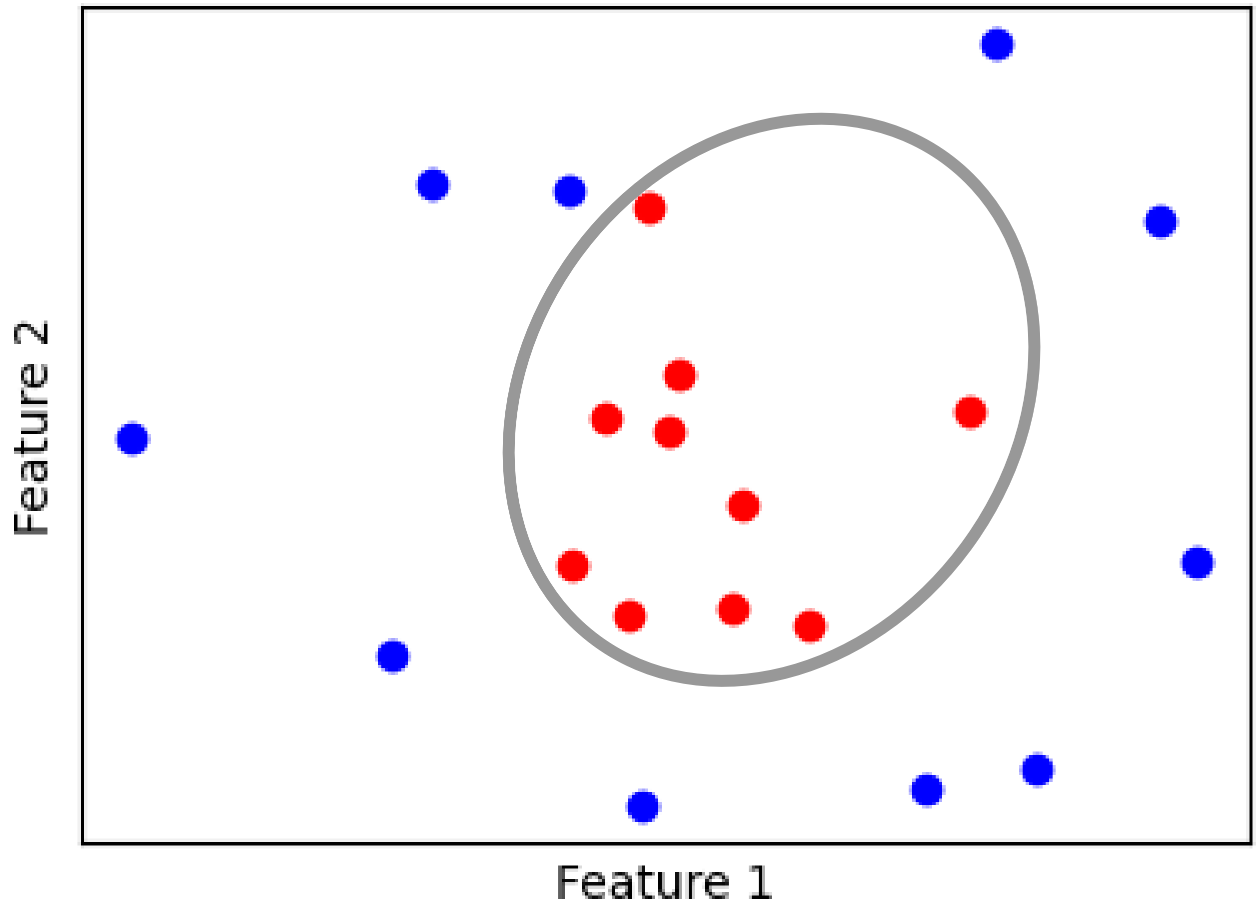
3D Feature space

Support Vector Machine

Linearly Separable

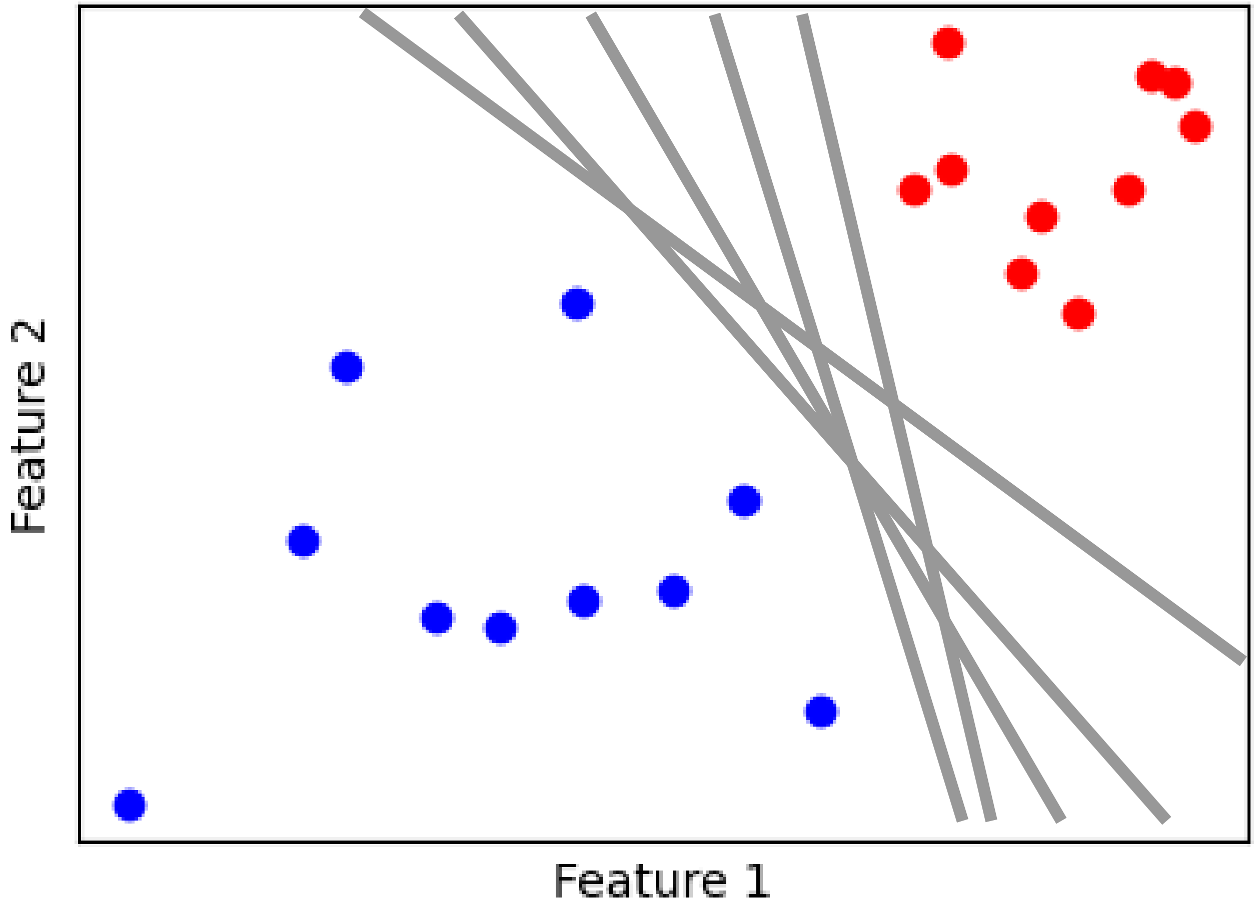


Non-Linearly Separable



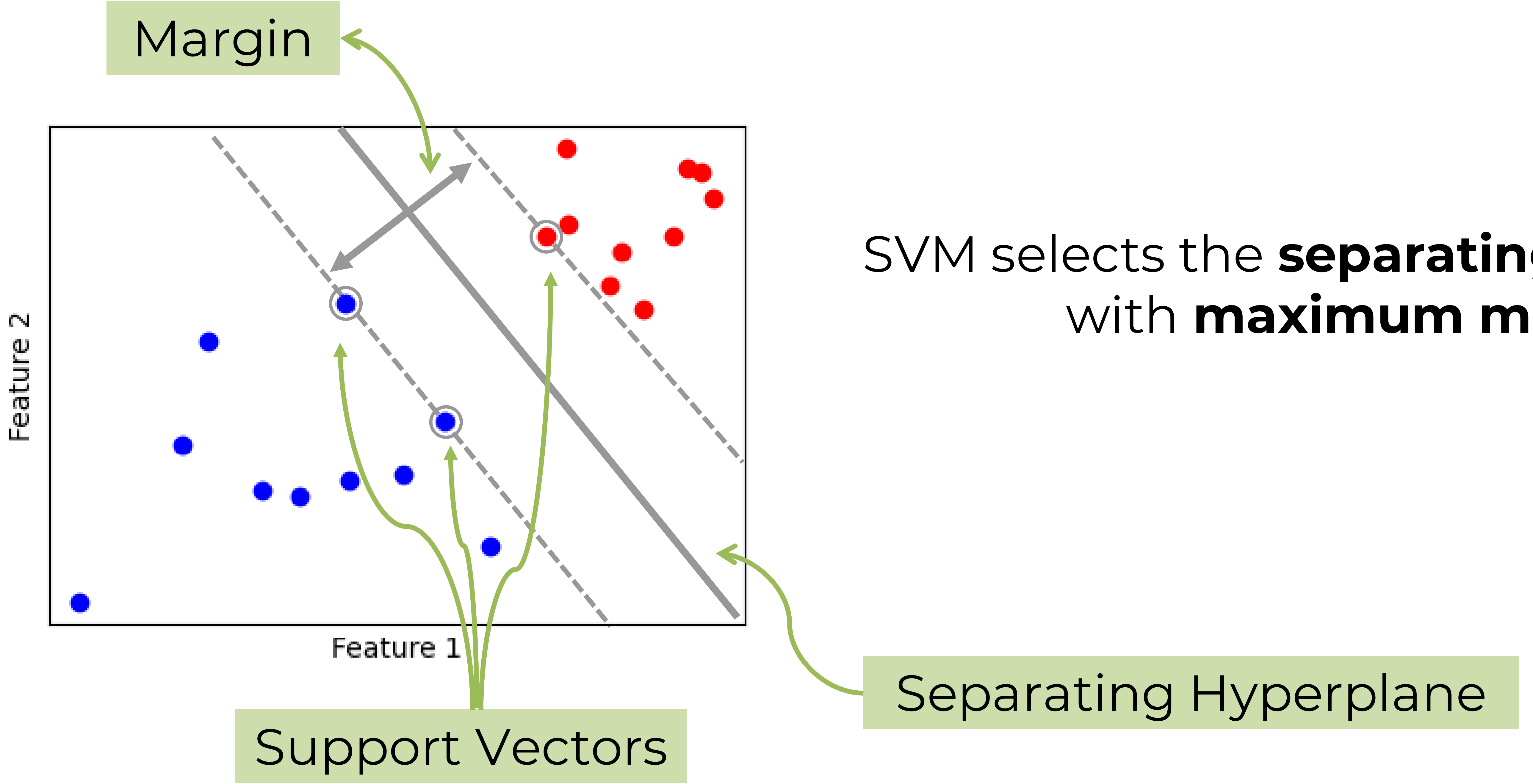
Support Vector Machine

Linearly Separable



Brain Code Camp

Support Vector Machine

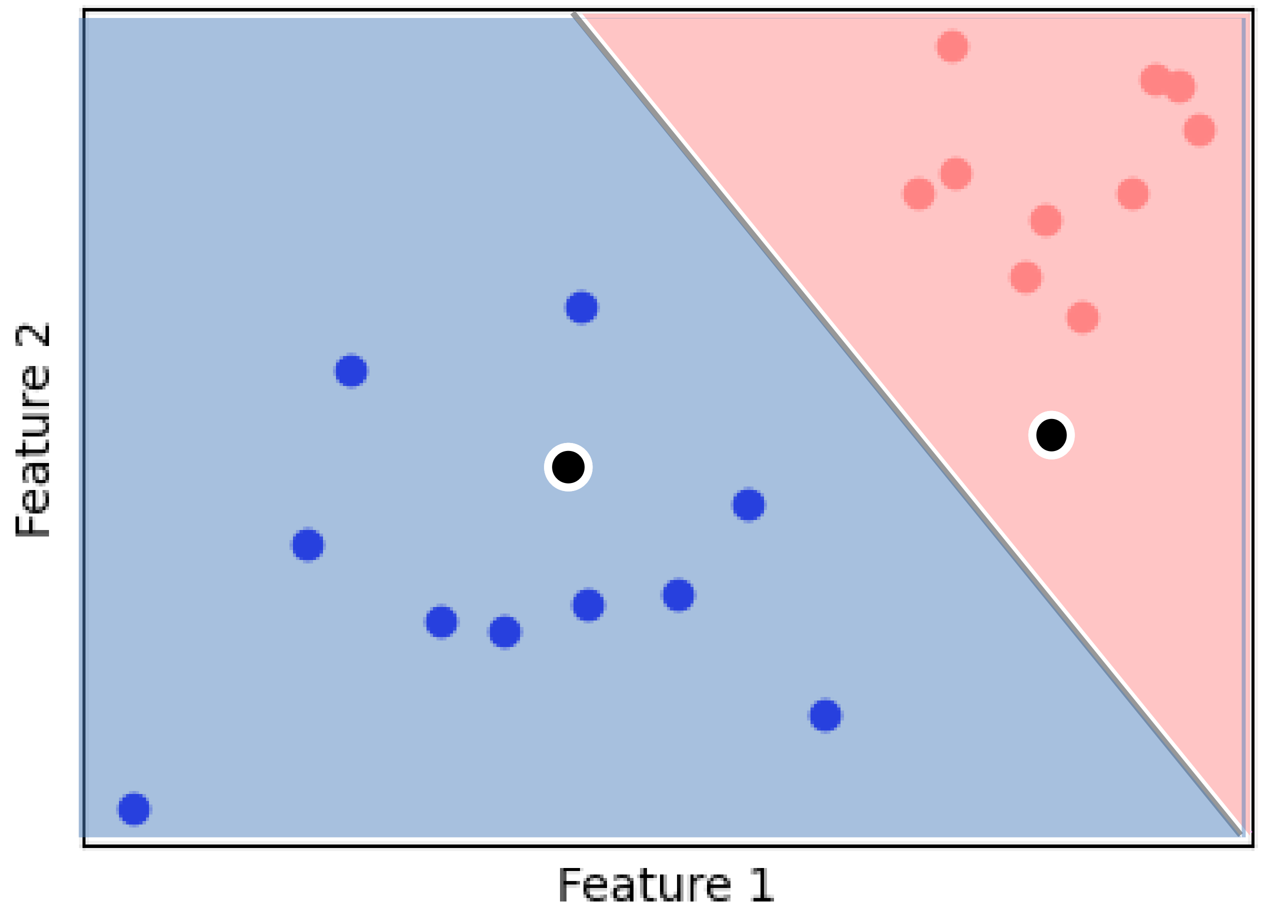


SVM selects the **separating hyperplane** with **maximum margin**.

Brain
Code
Camp

Support Vector Machine

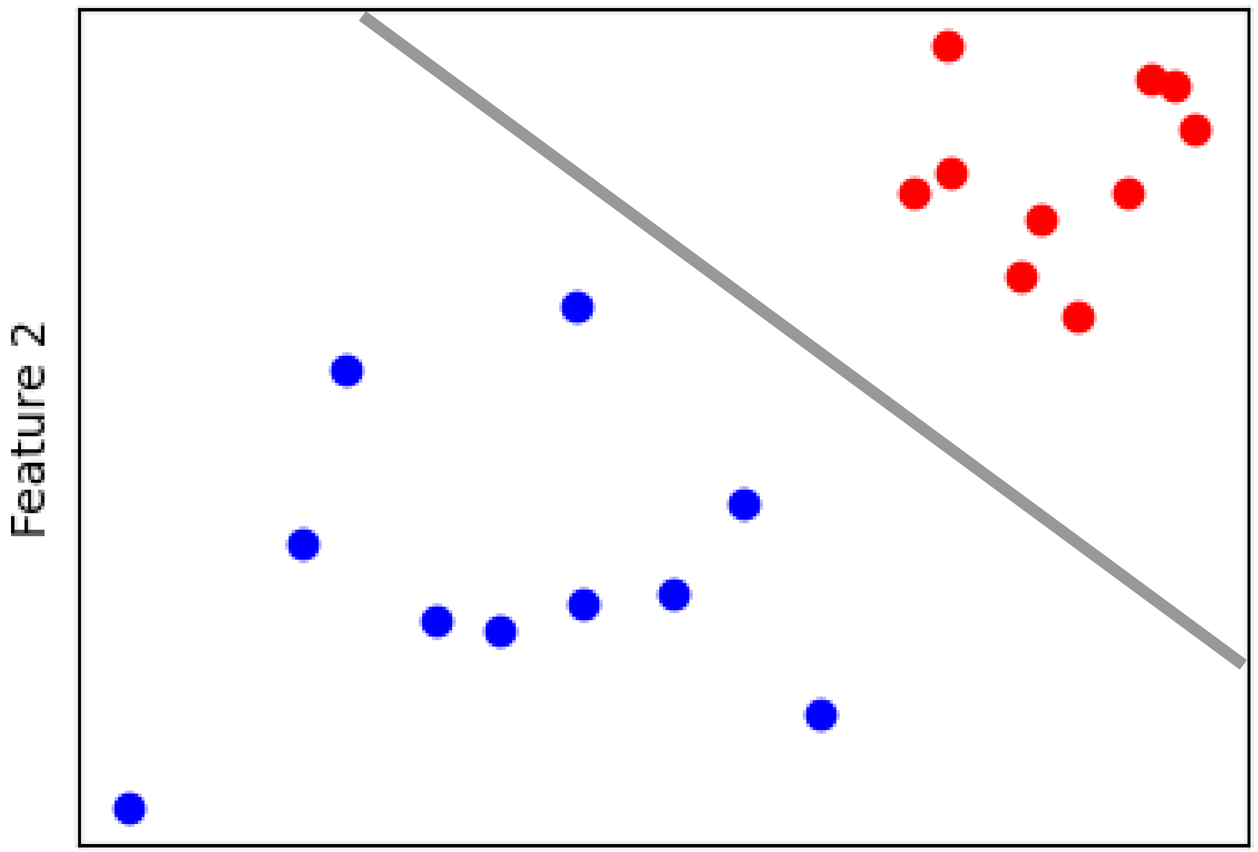
Decision Boundary



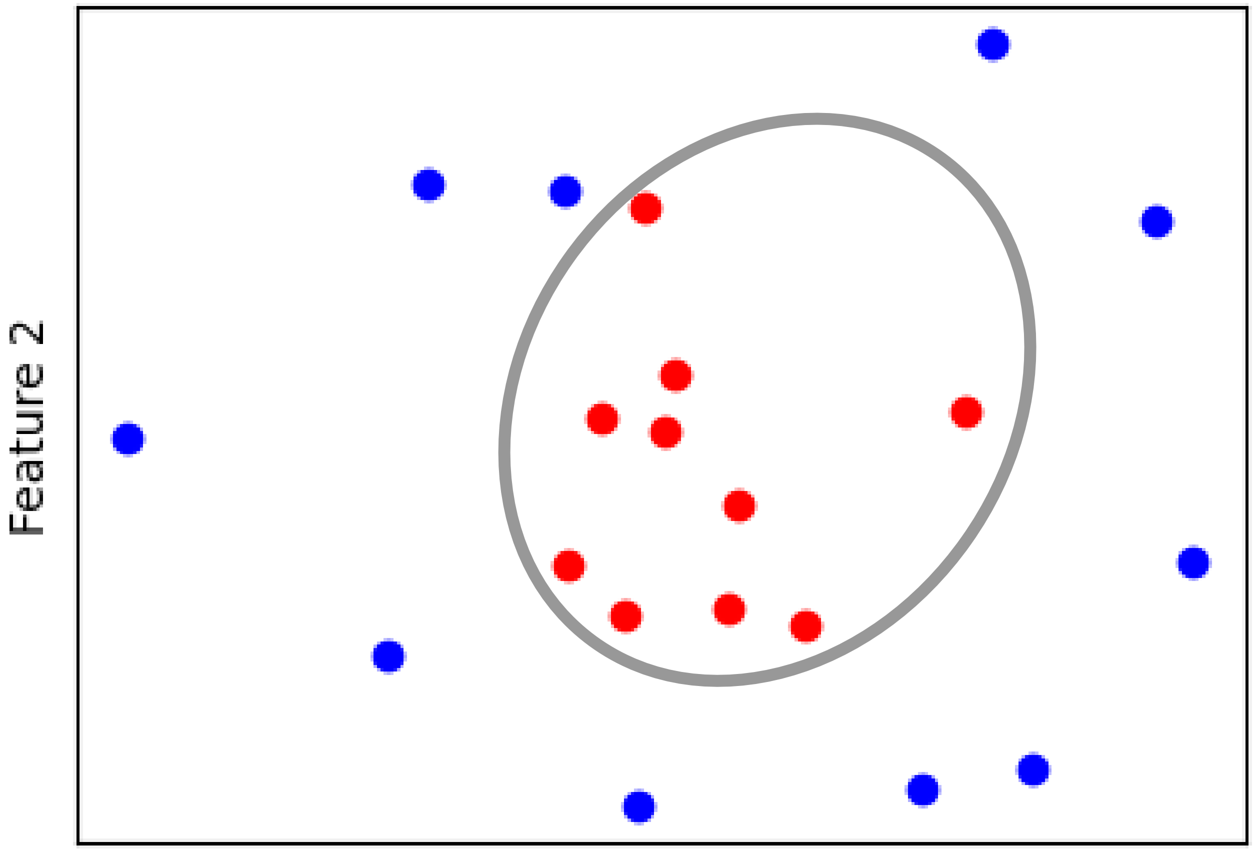
SVM selects the **separating hyperplane** with **maximum margin**.

Support Vector Machine

Linearly Separable

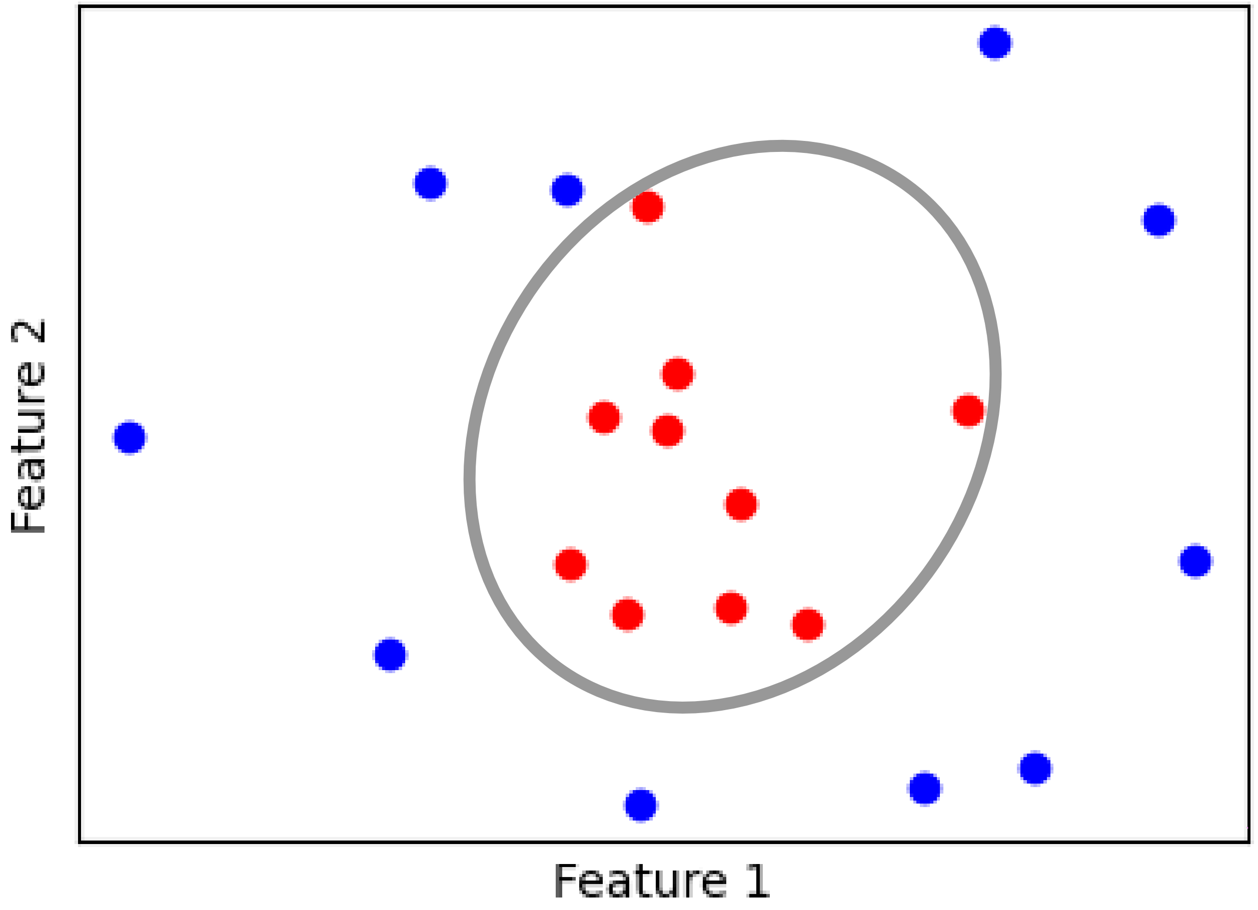


Non-Linearly Separable



Support Vector Machine

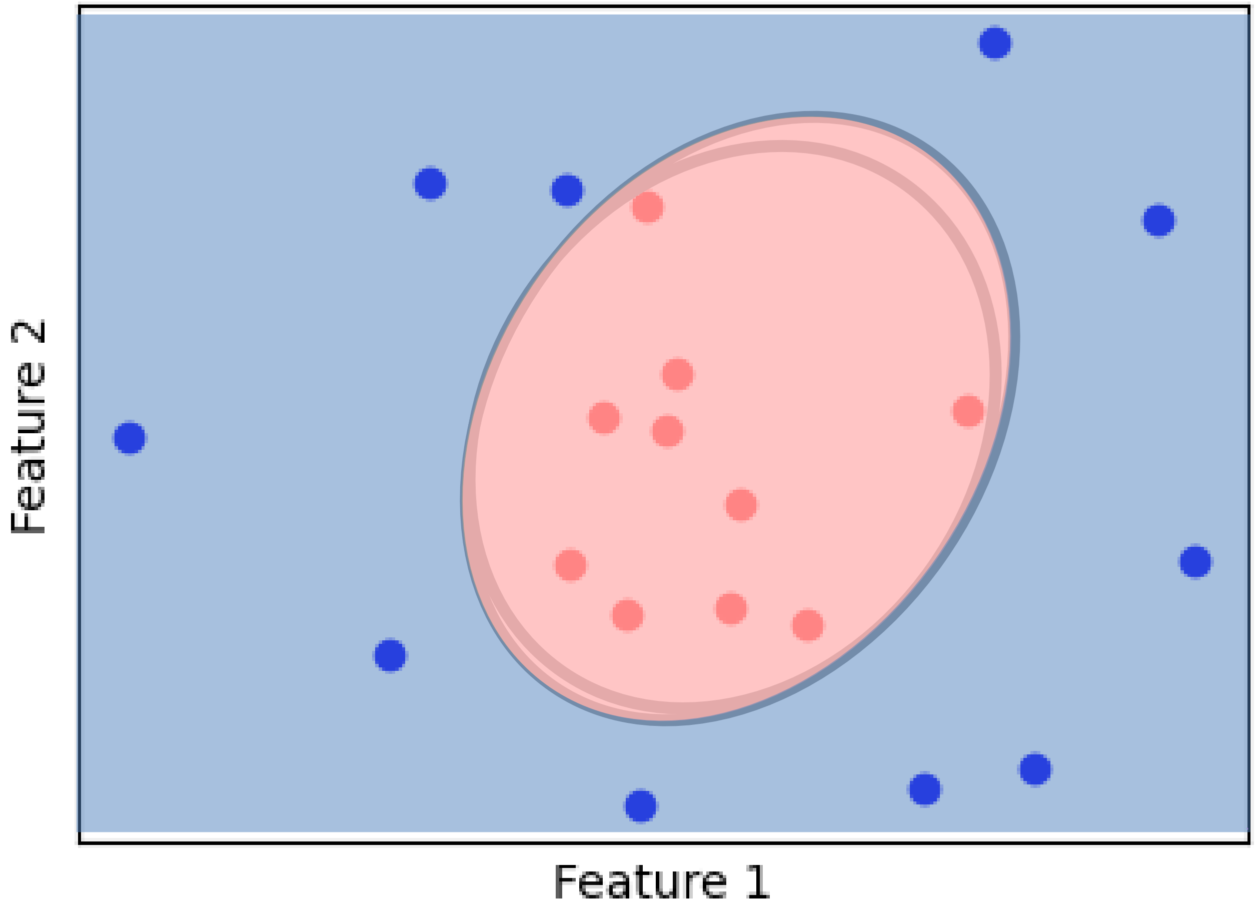
Non-Linearly Separable



SVM selects the **separating hyperplane** with **maximum margin**.

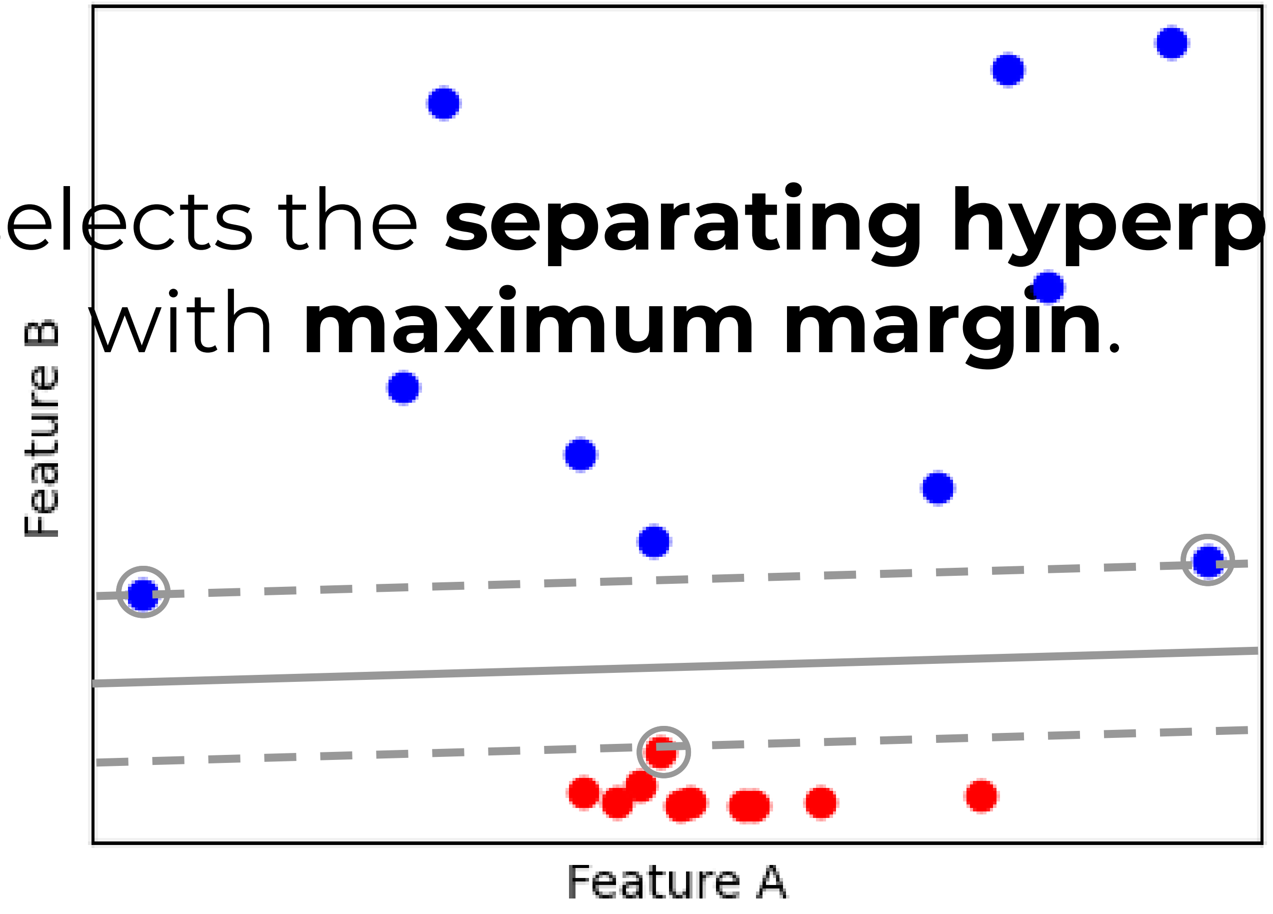
Support Vector Machine

Non-Linearly Separable



Kernel

Linearly Separable



SVM selects the **separating hyperplane** with **maximum margin**.

Non-Linear Decision Boundary

Brain
Code
Camp

sklearn.svm.SVC

class `sklearn.svm.SVC`(**, C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', break_ties=False, random_state=None*) ¶ [source]

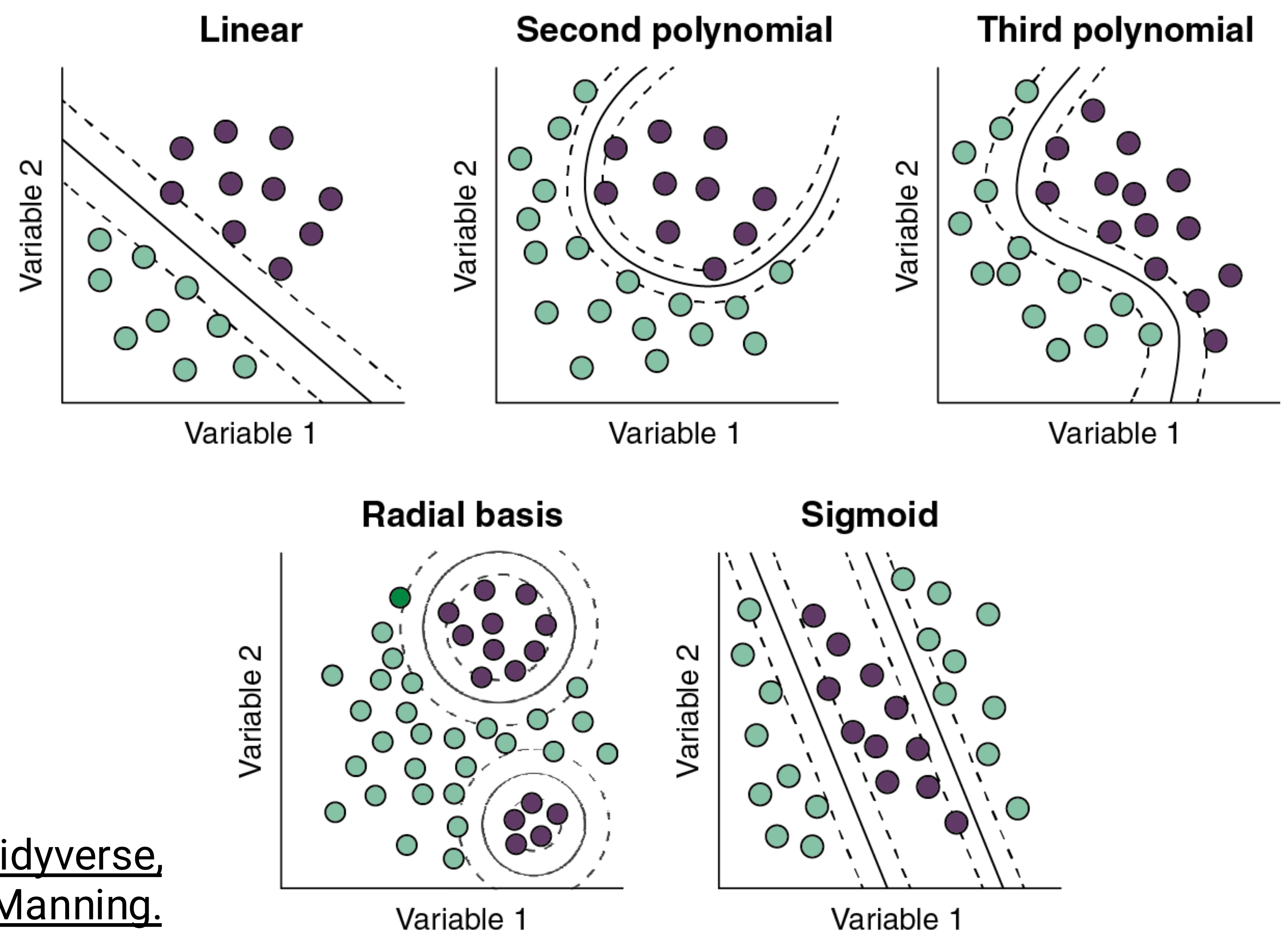
Parameters:

kernel : *{'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'} or callable, default='rbf'*

Specifies the kernel type to be used in the algorithm. If none is given, 'rbf' will be used.

degree : *int, default=3*

Degree of the polynomial kernel function ('poly').



Rhys, H. (2020). Machine Learning with R, the Tidyverse, and Mlr. United States: Manning.

sklearn.svm.SVC

class sklearn.svm.SVC(*, *C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', break_ties=False, random_state=None*) [source]

Parameters:

C : *float, default=1.0*

Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty.

$$\mathcal{L}(x,y) + \lambda\, R(w) \longrightarrow C\ \mathcal{L}(x,y) + R(w)$$

Setting C to high value will reduce the effect of regularization

sklearn.svm.SVC

class `sklearn.svm.SVC`(**, C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', break_ties=False, random_state=None*) [source]

Attributes:

`coef_` : ***ndarray of shape (n_classes * (n_classes - 1) / 2, n_features)***
Weights assigned to the features when `kernel="linear"`.

dual_coef_ : ***ndarray of shape (n_classes -1, n_SV)***
Dual coefficients of the support vector in the decision function

intercept_ : ***ndarray of shape (n_classes * (n_classes - 1) / 2,)***
Constants in decision function.

support_ : ***ndarray of shape (n_SV)***
Indices of support vectors.

support_vectors_ : ***ndarray of shape (n_SV, n_features)***
Support vectors.

`n_support_` : ***ndarray of shape (n_classes,), dtype=int32***
Number of support vectors for each class.

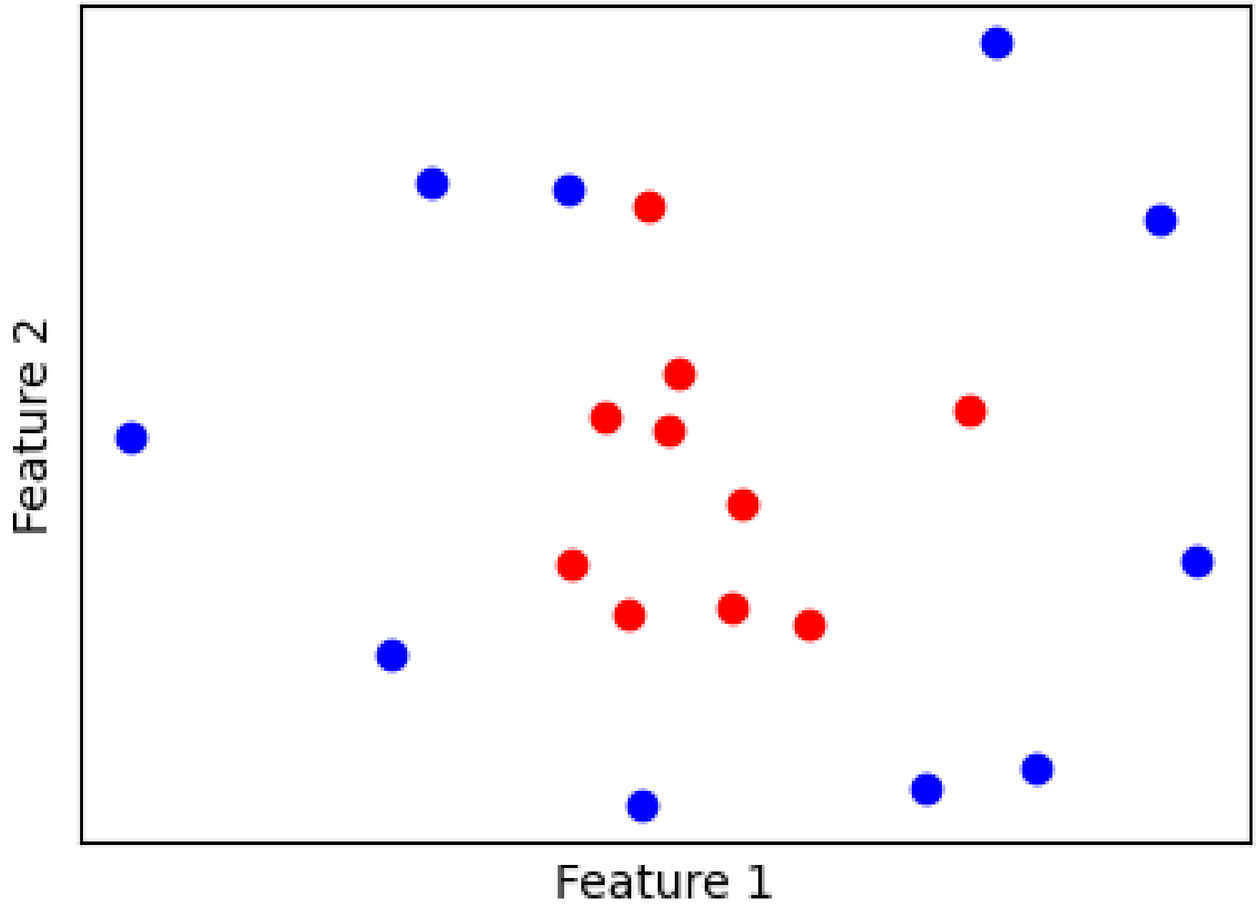
Decision Boundary

Support Vectors

Support Vector Machine

Dataset

$$(x_{1,1}, x_{1,2}, y_1), (x_{2,1}, x_{2,2}, y_2), (x_{3,1}, x_{3,2}, y_3), .., (x_{n,1}, x_{n,2}, y_n)$$



X

$$\begin{bmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \\ x_{3,1} & x_{3,2} \\ \vdots & \vdots \\ x_{n,1} & x_{n,2} \end{bmatrix}$$

shape = $(n, 2)$

y

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix}$$

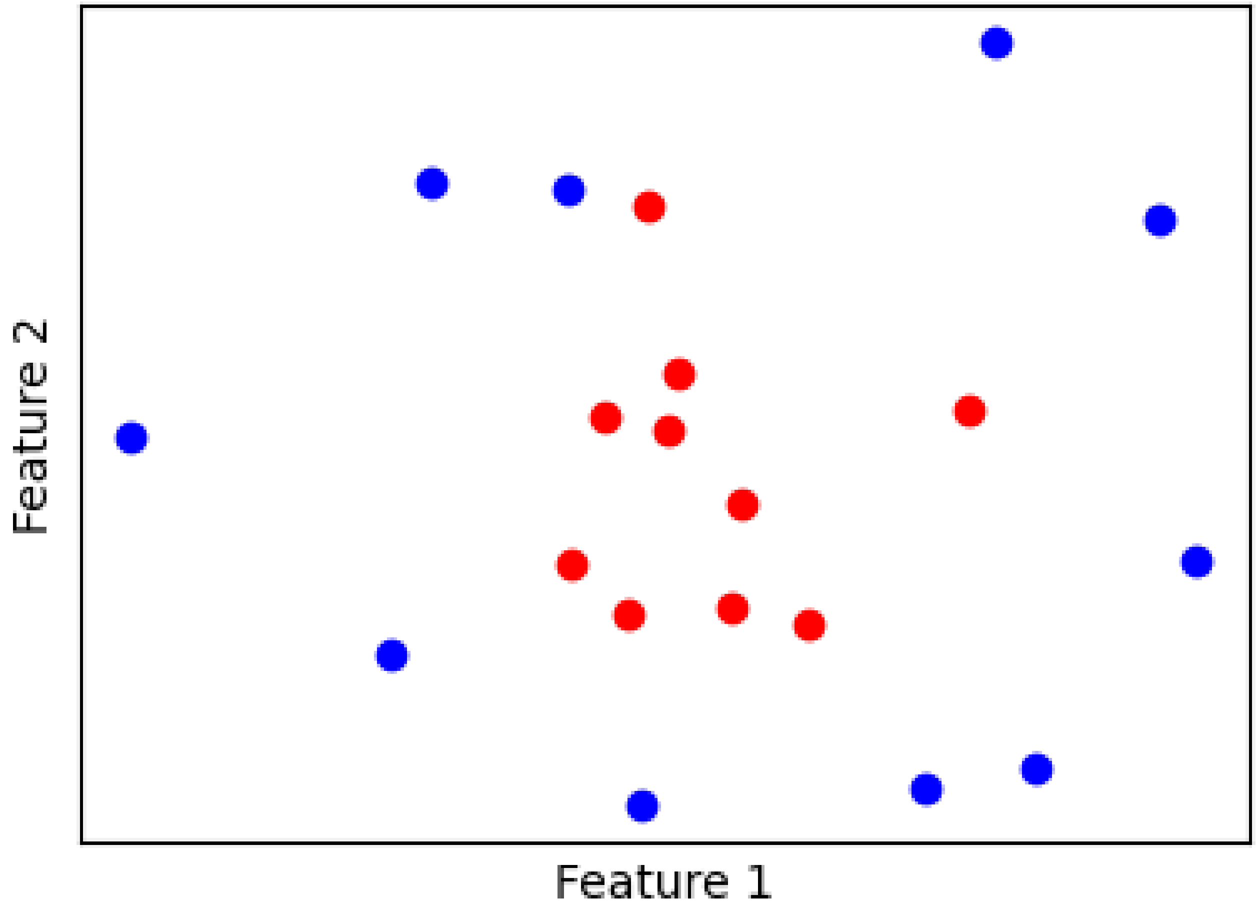
shape = $(n, 1)$

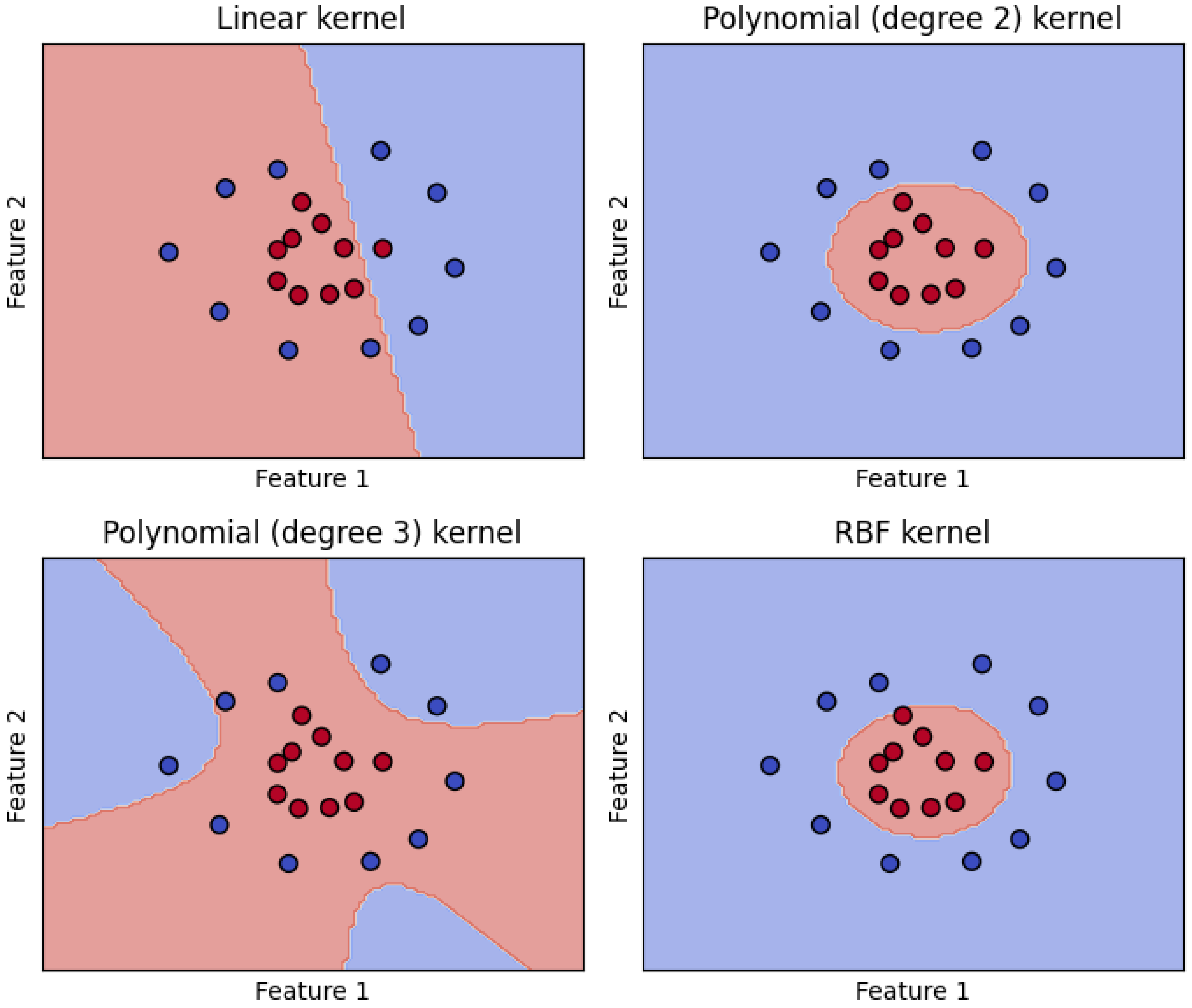
```
# Import a necessary modules
from sklearn.svm import SVC

# Create the model with linear kernel
clf = SVC(kernel='linear')

# Train the model
clf.fit(X,y)

# Make prediction
y_pred = clf.predict(X_test)
```

```
# Create the model with non-linear kernel
clf = SVC(kernel='poly', degree=2)
clf = SVC(kernel='rbf')
```

Brain Code Camp

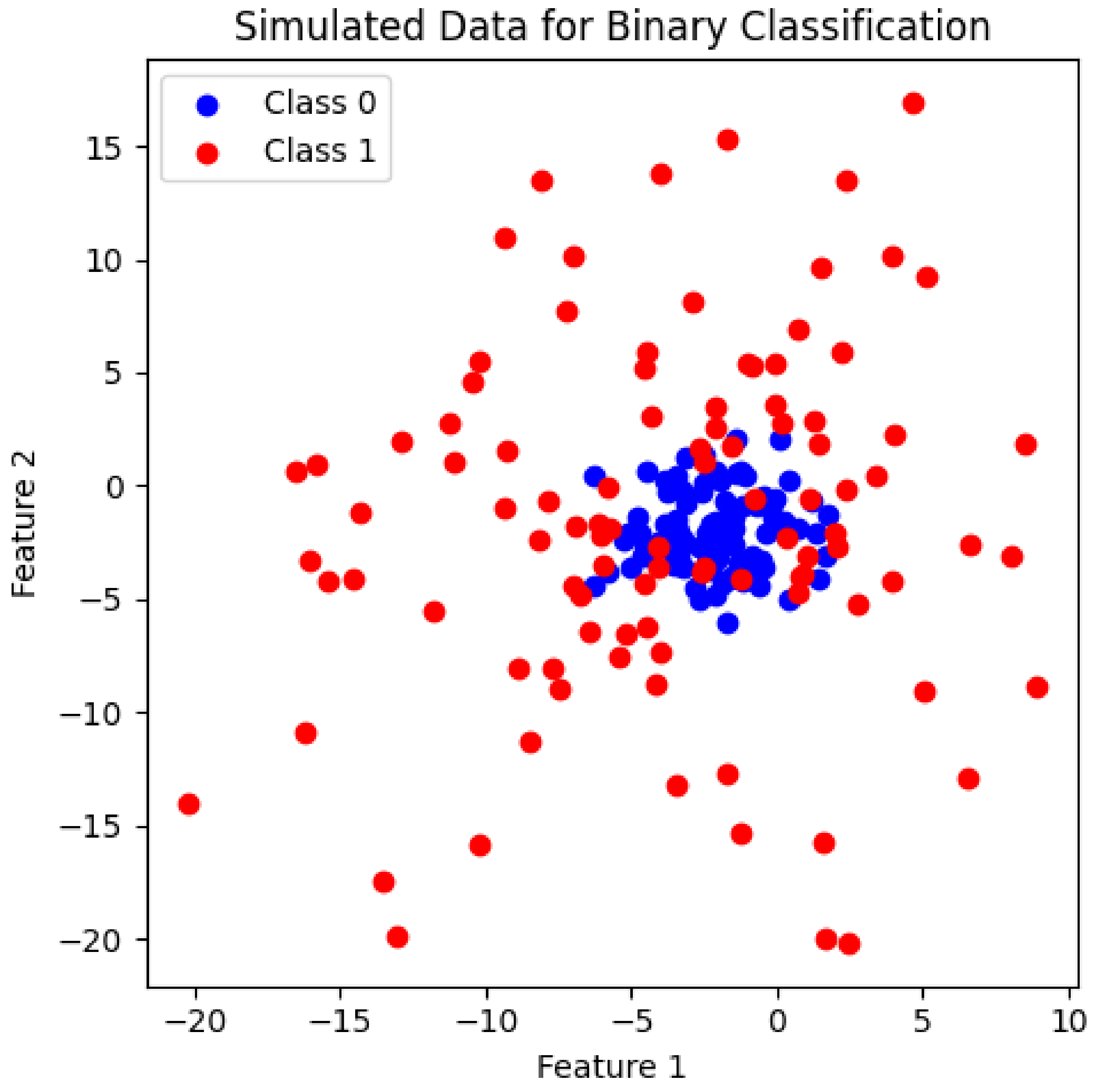
Support Vector Machine

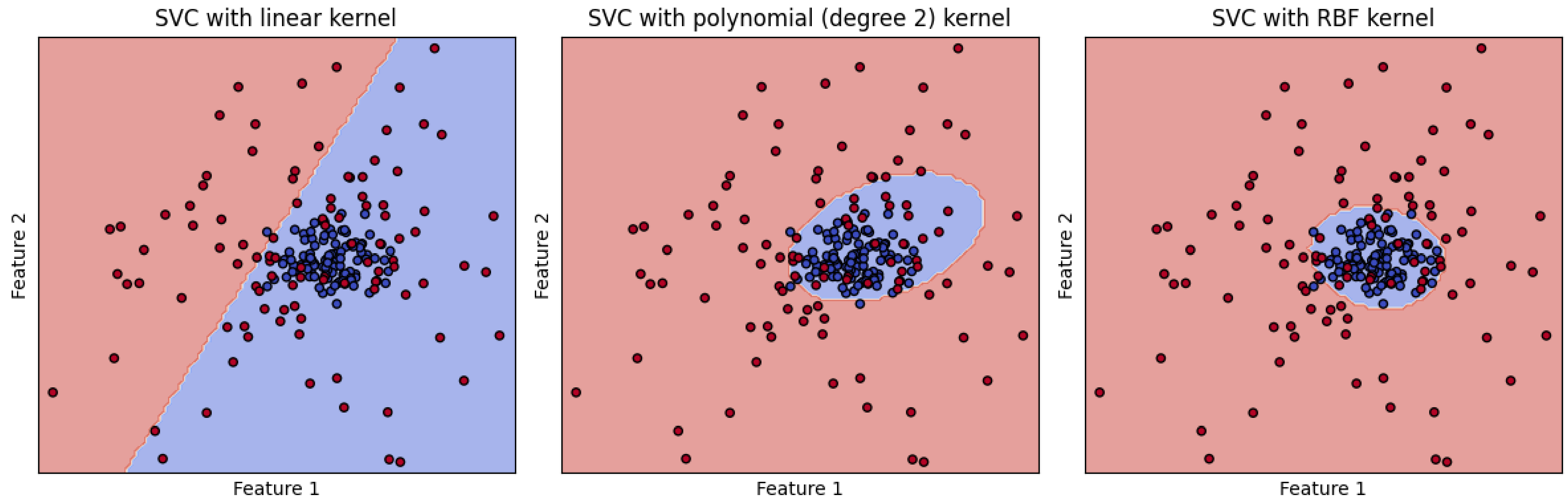




Brain Code Camp

Support Vector Machine

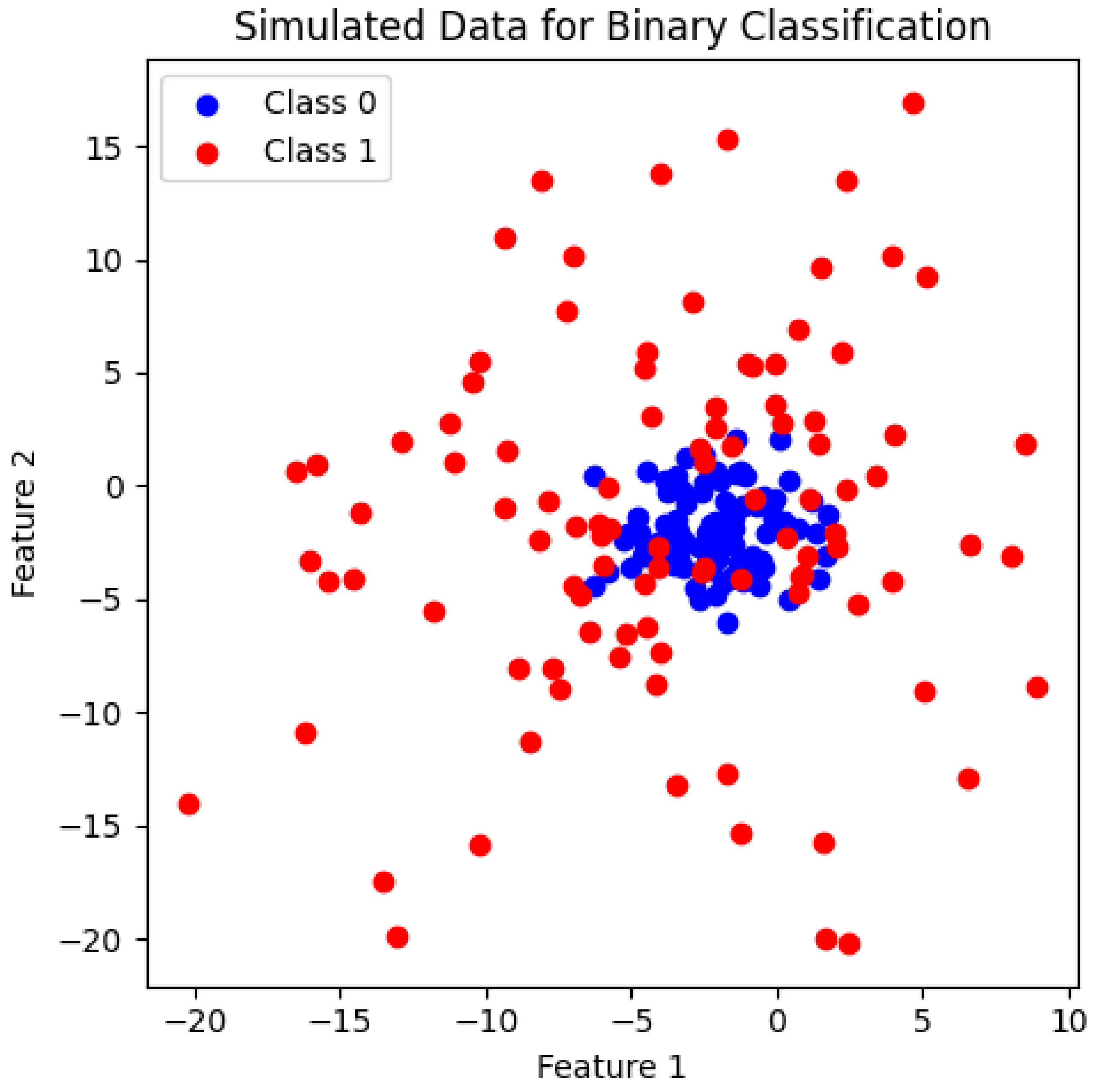


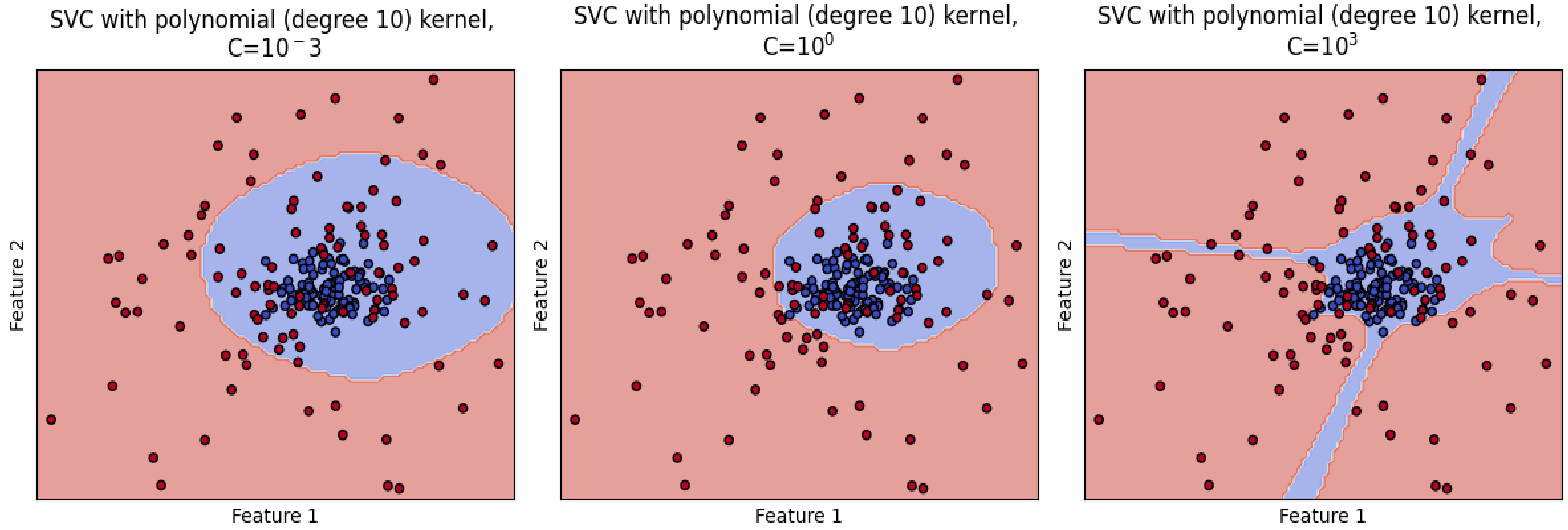


Linear kernels always yield linear decision boundary. For a non-linearly separable dataset, a non-linear kernel is preferred.

Thus, polynomial (degree 2) and RBF kernels can fit this dataset better than linear kernels.

Support Vector Machine





$$C\ \mathcal{L}(x,y) + R(w)$$

- Too low C → regulation effect is strengthened. → Underfitting
- Too high C → regulation effect is weakened. → Overfitting