Machine Learning

Lecture 09

Min-Kuan Chang

minkuanc@nchu.edu.tw

EE, College of EECS

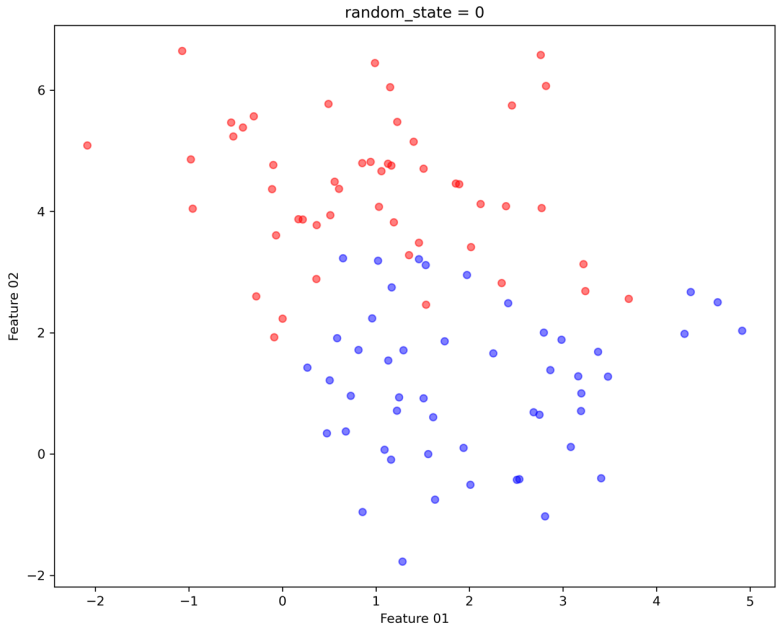
Implementation Using Scikit-Learn

Topic 01

LinearSVC

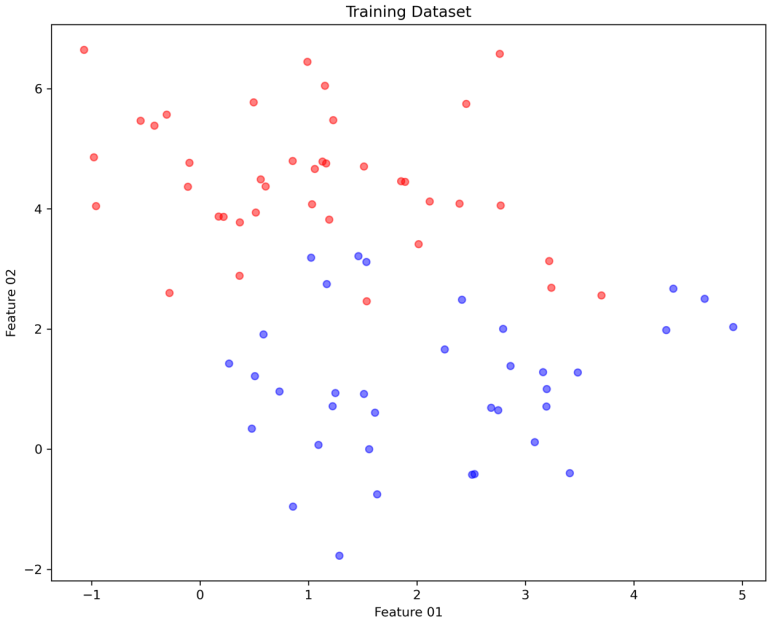
Two-class Classification
Step 01: Dataset Preparation

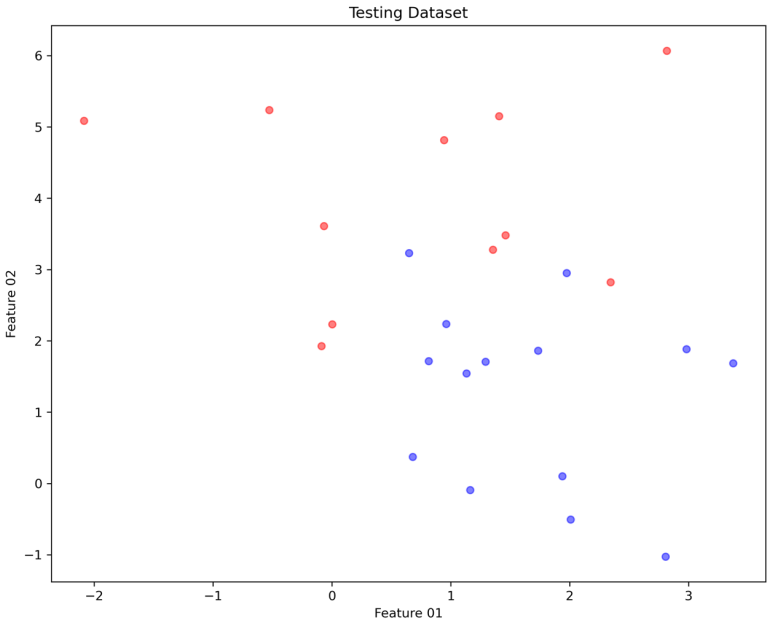
```
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples = 100, n_features = 2, centers = 2, cluster_std = 1.2, random_state = 0)
```



Two-class Classification
Step 02: Training/Testing Dataset

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 10)
```





Two-class Classification
Step 03: Build/Train ML Model

- LinearSVC in Scikit-Learn is responsible for building a SVM model when the kernel is linear

```
from sklearn.svm import LinearSVC
LinearSVC_Model = LinearSVC(penalty='l2', C=1, max_iter=10000)
```

  - 'penalty' is used to regularize the weights in SVM model
    - it can be 'l1' or 'l2'
  - 'C' is used to specify the strength of regularization
    - smaller 'C' means stronger regularization
  - 'max_iter' is the maximum number of iteration when searching for the optimal weights

Two-class Classification
Step 03: Build/Train ML Model

- The class function 'fit' is called to train the model

```
LinearSVC_Model.fit(X_train, y_train)
```

  - in 'fit' method, we simply let X_train and y_train as arguments to call this method

Two-class Classification
Step 04: Test the ML model

- We can predict the class of a new data

```
LinearSVC_Model.predict([[6,5]])
```

  - simply put this new data point in the argument of 'predict' method
  - the output looks like

```
array([0])
```

- We can also obtain the accuracy of a dataset

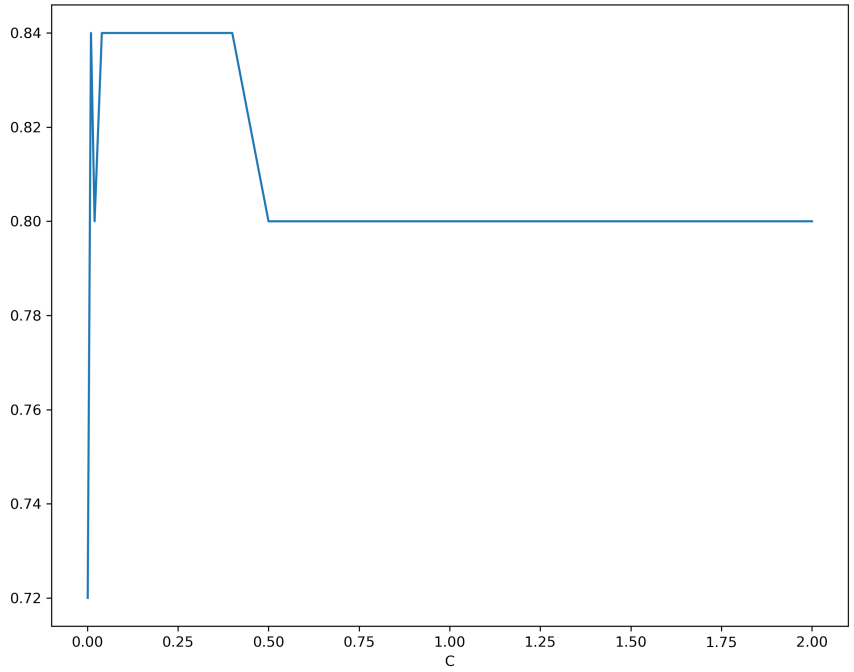
```
LinearSVC_Model.score(X_test, y_test)
```

  - the arguments in score are the data and its corresponding target
  - the output looks like

```
0.8
```

Two-class Classification Discussion

- How does ‘C’ affect the accuracy?



Two-class Classification Discussion

- What is the decision boundary?
  - the decision boundary

$$w_1 x_1 + w_2 x_2 + w_0 = 0$$

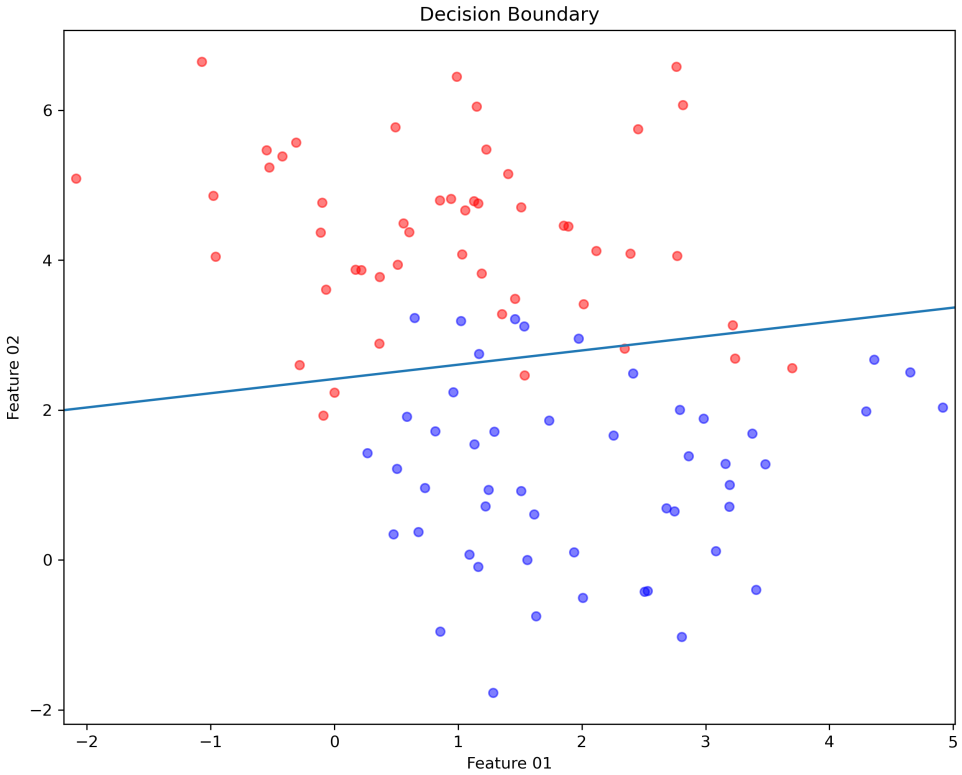
  - w_0 is recorded in 'LinearSVC_Model.intercept_'

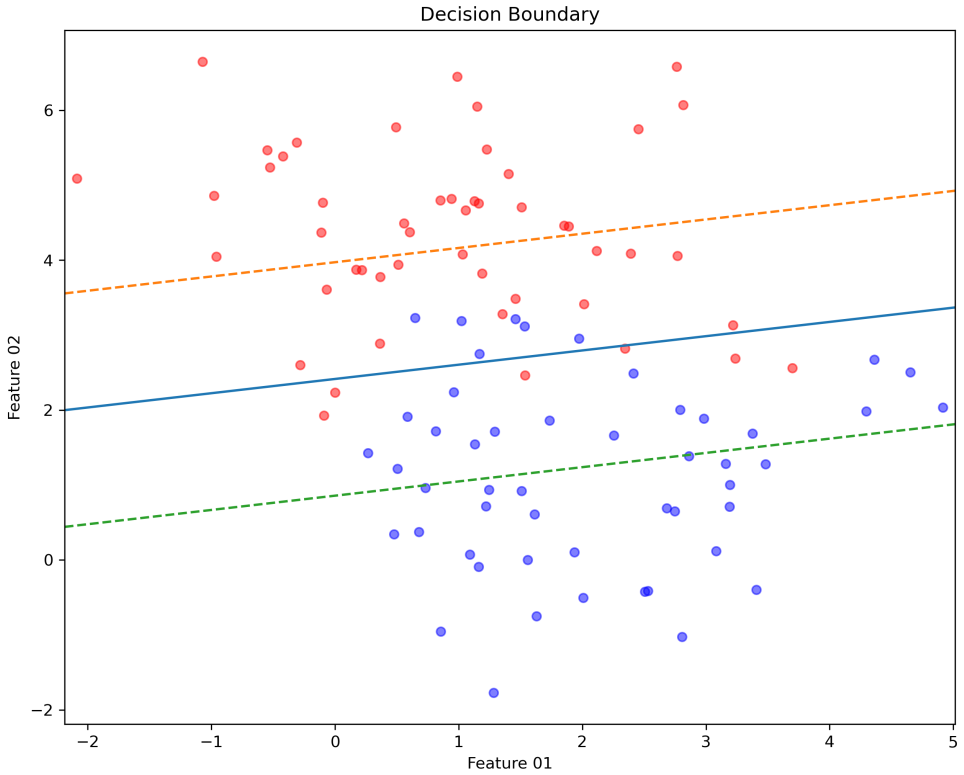
```
array([1.55000129])
```

  - w_1 and w_2 can be found in 'LinearSVC_Model.coef_'

```
array([[ 0.12214971, -0.64214216]])
```

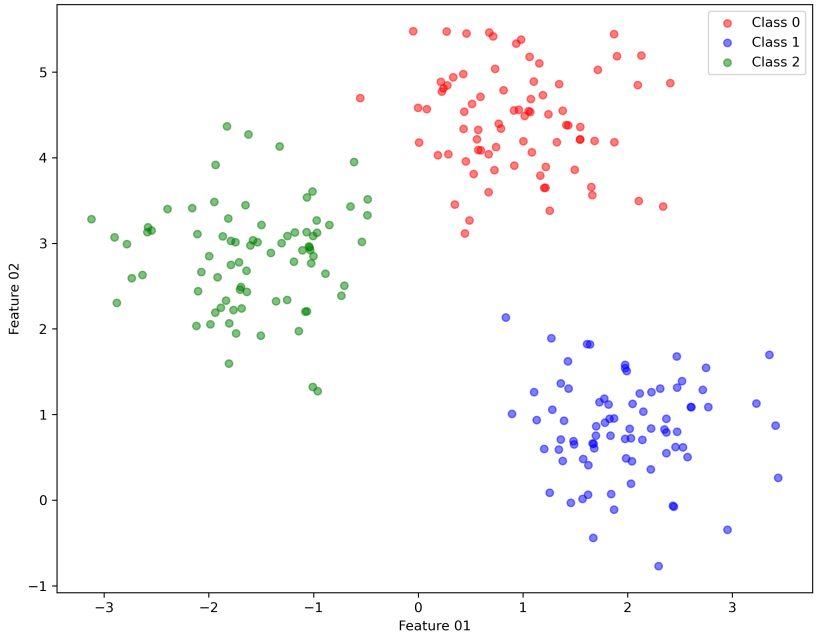
Two-class Classification Discussion





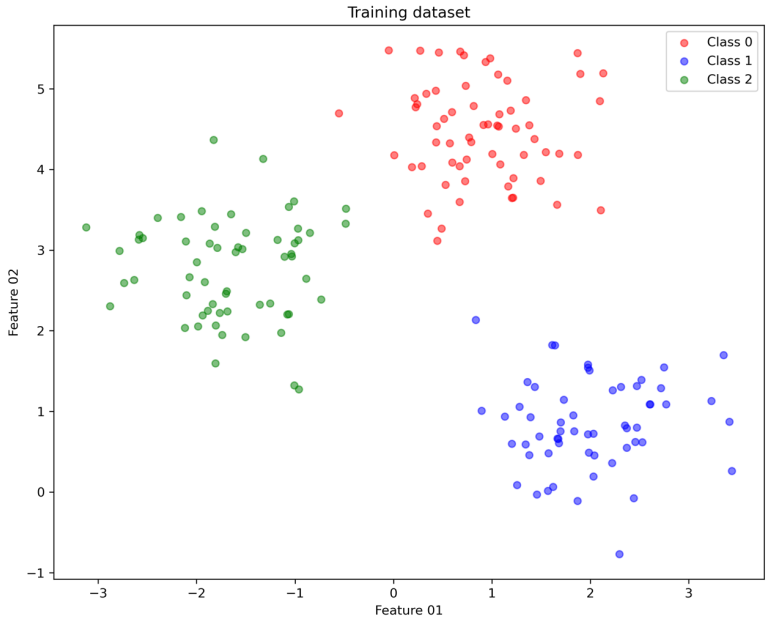
Multi-class Classification
Step 01: Dataset Preparation

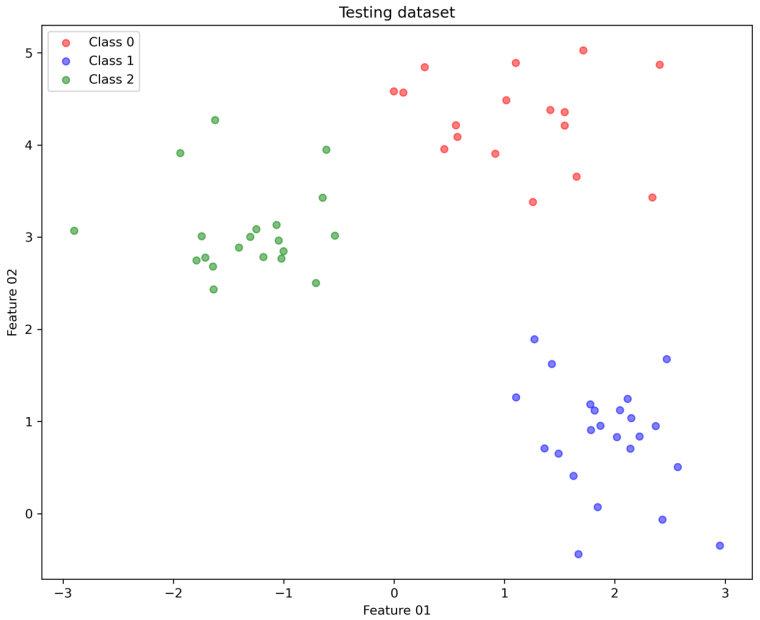
```
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples = 240, n_features = 2, centers = 3, cluster_std = 0.6, random_state = 0)
```



Multi-class Classification
Step 02: Training/Testing Dataset

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 10)
```





Multi-class Classification
Step 03: Build/Train ML Model

```
from sklearn.svm import LinearSVC

LinearSVC_Model = LinearSVC(penalty='l2', C=1, max_iter=10000)
LinearSVC_Model.fit(X_train, y_train)
```

Multi-class Classification
Step 04: Test the ML model

- We can predict the class of a new data

```
LinearSVC_Model.predict([[6,5]])
```

  - simply put this new data point in the argument of 'predict' method
  - the output looks like

```
array([0])
```

- We can also obtain the accuracy of a dataset

```
LinearSVC_Model.score(X_test, y_test)
```

  - the arguments in score are the data and its corresponding target
  - the output looks like

```
1.0
```

Multi-class Classification
Step 04: Test the ML model

- In the class method, we have a method called 'decision_function'

```
LinearSVC_Model.decision_function([[6,5]])
```

  - its output is

```
array([[ 4.38645532, 1.42885203, -9.33367477]])
```

  - these three represents the values of discriminant functions of Class 0, 1, and 2, respectively
  - since Class 0 has the largest value, [6,5] is classified into Class 0

Multi-class Classification
Step 04: Test the ML model

- In the class method, we have a method called 'decision_function'

```
LinearSVC_Model.decision_function([[6,5]])
```

  - its output is

```
array([[ 4.38645532, 1.42885203, -9.33367477]])
```

  - the information of the discriminant functions is in 'LinearSVC_Model.intercept_' and 'LinearSVC_Model.coef_'

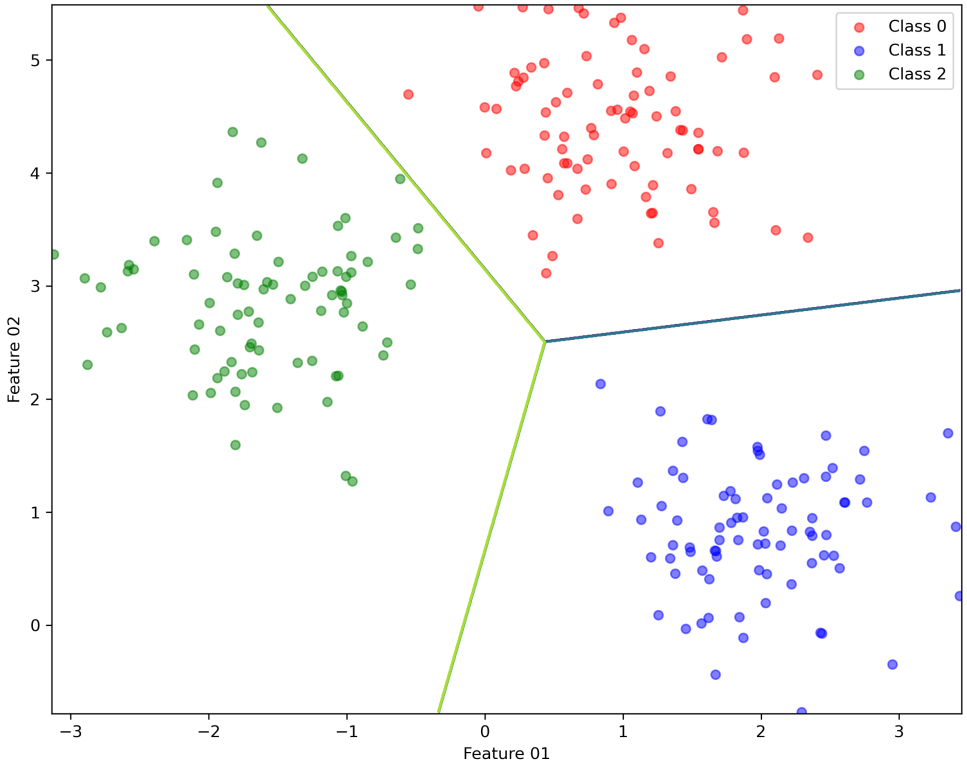
```
LinearSVC_Model.intercept_
```

```
array([-3.35114026, 1.01472471, 0.67667072])
```

```
LinearSVC_Model.coef_
```

```
array([[ 0.49474879, 0.95382057], [ 0.76418148, -0.83419231], [-1.39721975, -0.3254054 ]])
```

Multi-class Classification
Step 04: Test the ML model



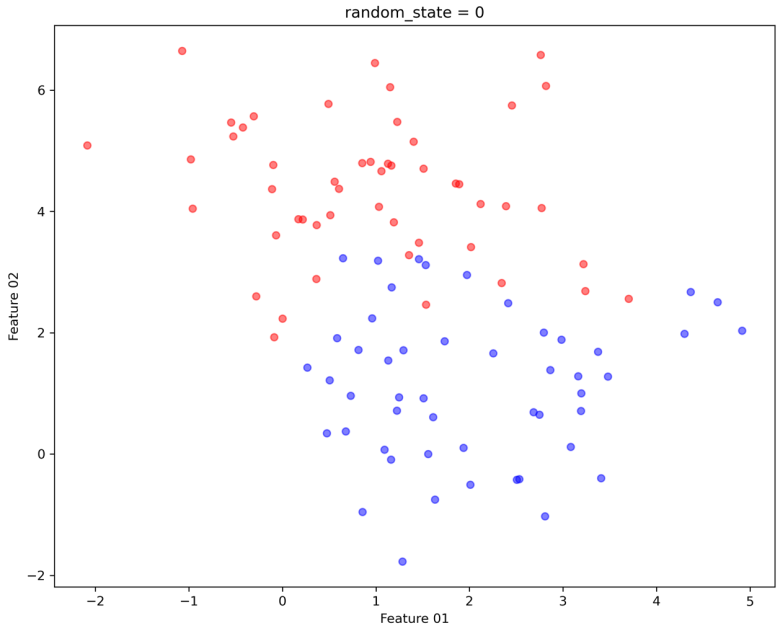
Implementation Using Scikit-Learn

Topic 02

LogisticRegression

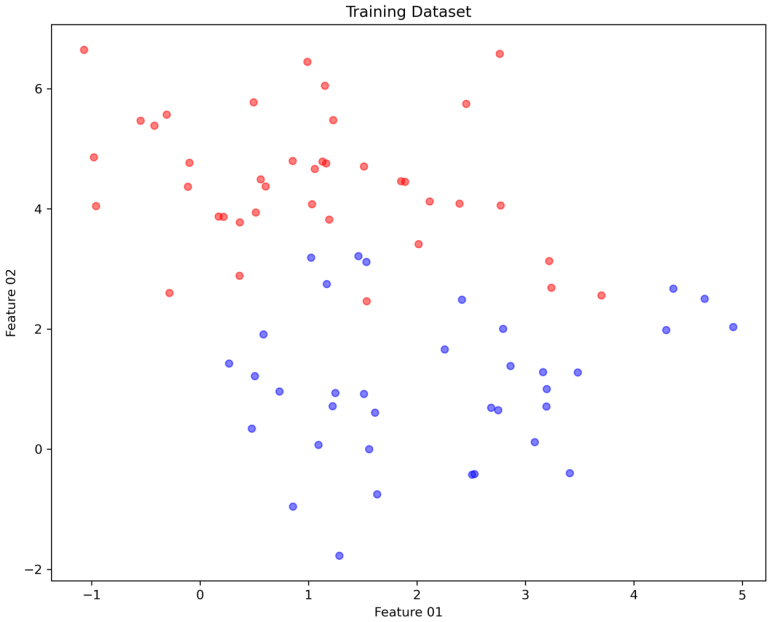
Two-class Classification
Step 01: Dataset Preparation

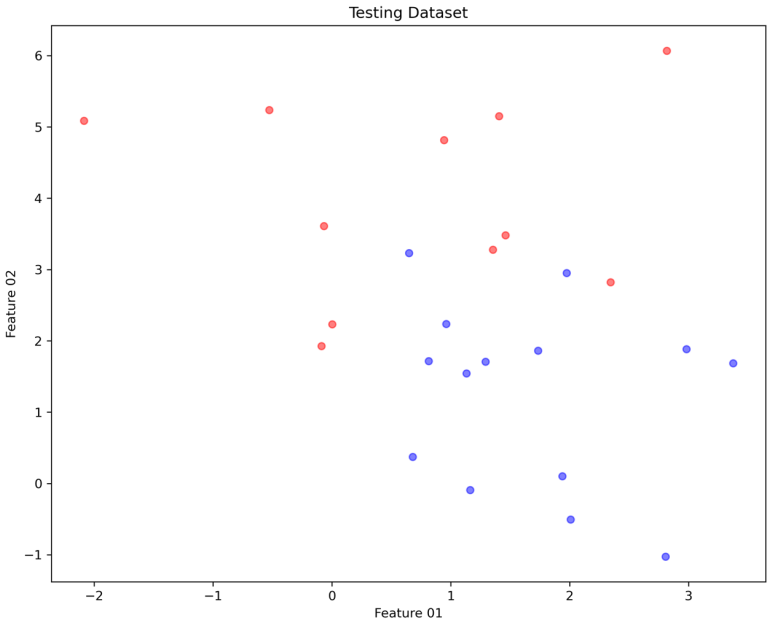
```
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples = 100, n_features = 2, centers = 2, cluster_std = 1.2, random_state = 0)
```



Two-class Classification
Step 02: Training/Testing Dataset

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 10)
```





Two-class Classification
Step 03: Build/Train ML Model

- 'LogisticRegression' in Scikit-Learn is responsible for building a logistic regression model

```
from sklearn.linear_model import LogisticRegression
LogisticRegression_Model = LogisticRegression (penalty='l2', C=1, max_iter=10000)
```

  - 'penalty' is used to regularize the weights in SVM model
    - it can be 'l1' or 'l2'
  - 'C' is used to specify the strength of regularization
    - smaller 'C' means stronger regularization
  - 'max_iter' is the maximum number of iteration when searching for the optimal weights

Two-class Classification
Step 03: Build/Train ML Model

- The class function ‘fit’ is called to train the model

```
LogisticRegression_Model.fit(X_train, y_train)
```

  - in ‘fit’ method, we simply let X_train and y_train as arguments to call this method

Two-class Classification
Step 04: Test the ML model

- We can predict the class of a new data

```
LogisticRegression_Model.predict([[3,-2]])
```

  - simply put this new data point in the argument of 'predict' method
  - the output looks like

```
array([1])
```

- We can also obtain the accuracy of a dataset

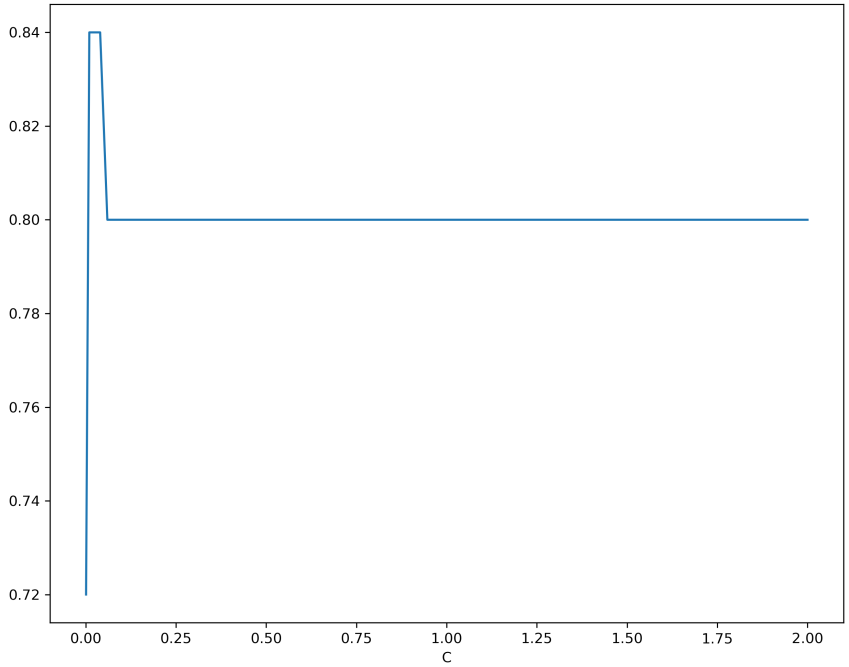
```
LogisticRegression_Model.score(X_test, y_test)
```

  - the arguments in score are the data and its corresponding target
  - the output looks like

```
0.8
```

Two-class Classification Discussion

- How does ‘C’ affect the accuracy?



Two-class Classification Discussion

- What is the decision boundary?
  - the decision boundary

$$w_1 x_1 + w_2 x_2 + w_0 = 0$$

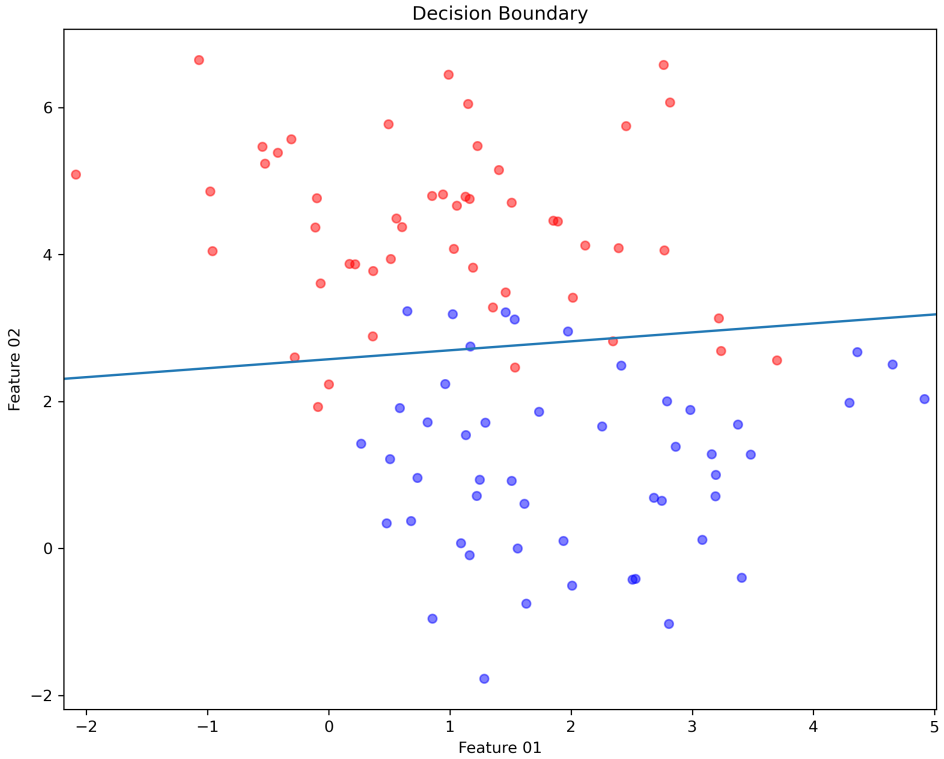
  - w_0 is recorded in 'LogisticRegression_Model.intercept_'

```
array([5.07667728])
```

  - w_1 and w_2 can be found in 'LogisticRegression_Model.coef_'

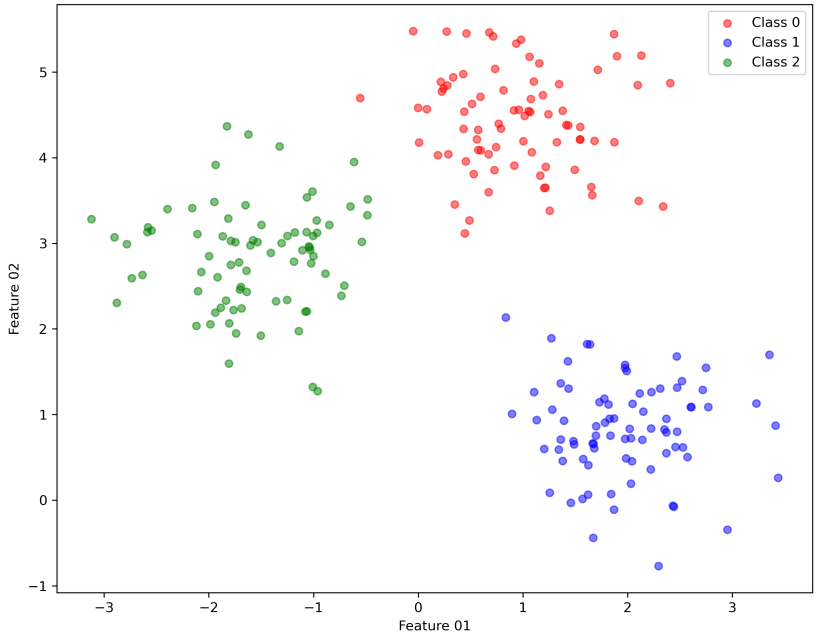
```
array([[ 0.24045067, -1.97272652]])
```

Two-class Classification Discussion



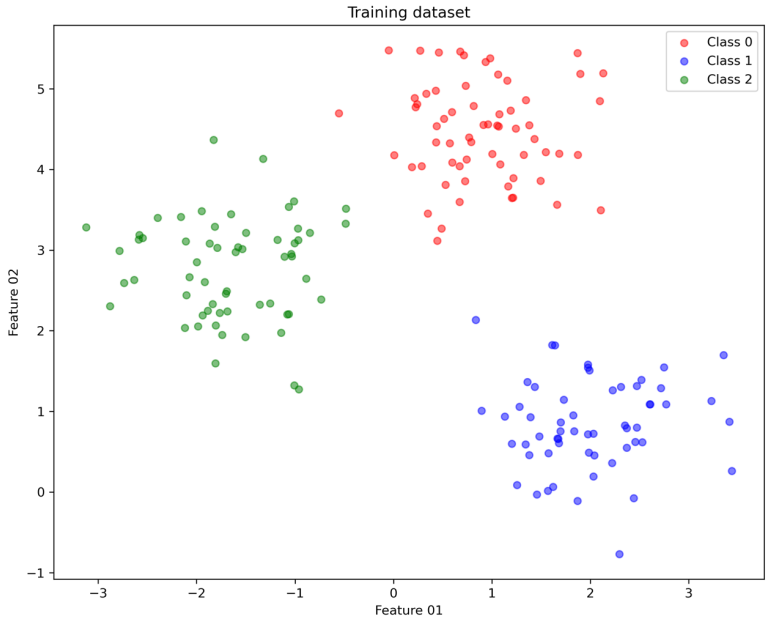
Multi-class Classification
Step 01: Dataset Preparation

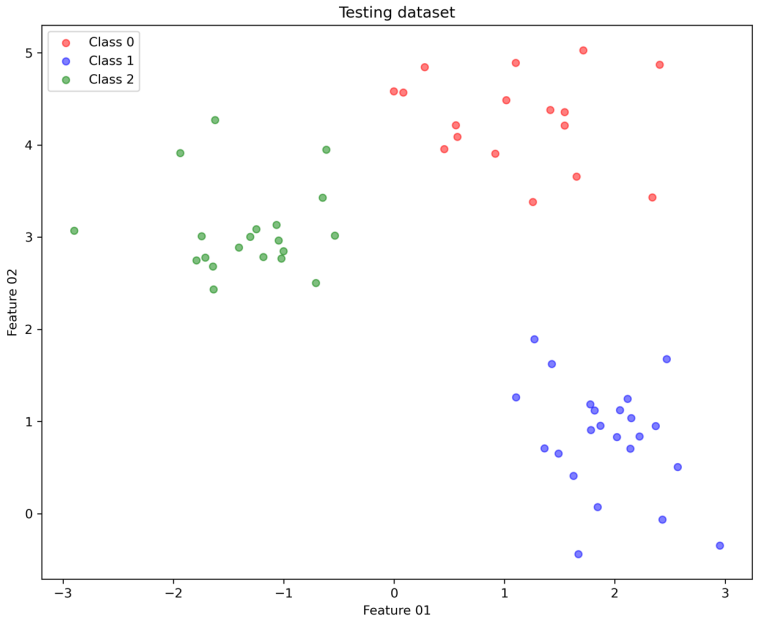
```
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples = 240, n_features = 2, centers = 3, cluster_std = 0.6, random_state = 0)
```



Multi-class Classification
Step 02: Training/Testing Dataset

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 10)
```





Multi-class Classification
Step 03: Build/Train ML Model

```
from sklearn.linear_model import LogisticRegression
LogisticRegression_Model = LogisticRegression (penalty='l2', C=1, max_iter=10000)
LogisticRegression_Model.fit(X_train, y_train)
```

Multi-class Classification
Step 04: Test the ML model

- We can predict the class of a new data

```
LogisticRegression_Model.predict([[6,5]])
```

  - simply put this new data point in the argument of 'predict' method
  - the output looks like

```
array([0])
```

- We can also obtain the accuracy of a dataset

```
LogisticRegression_Model.score(X_test, y_test)
```

  - the arguments in score are the data and its corresponding target
  - the output looks like

```
1.0
```

Multi-class Classification
Step 04: Test the ML model

- In the class method, we have a method called 'decision_function'

```
LogisticRegression_Model.decision_function([[6,5]])
```

  - its output is

```
array([[ 9.97861555, 2.67181898, -12.65043453]])
```

  - these three represents the values of discriminant functions of Class 0, 1, and 2, respectively
  - since Class 0 has the largest value, [6,5] is classified into Class 0

Multi-class Classification
Step 04: Test the ML model

- In the class method, we have a method called 'decision_function'

```
LogisticRegression_Model.decision_function([[6,5]])
```

- its output is

```
array([[ 9.97861555, 2.67181898, -12.65043453]])
```

- the information of the discriminant functions is in 'LinearSVC_Model.intercept_' and 'LinearSVC_Model.coef_'

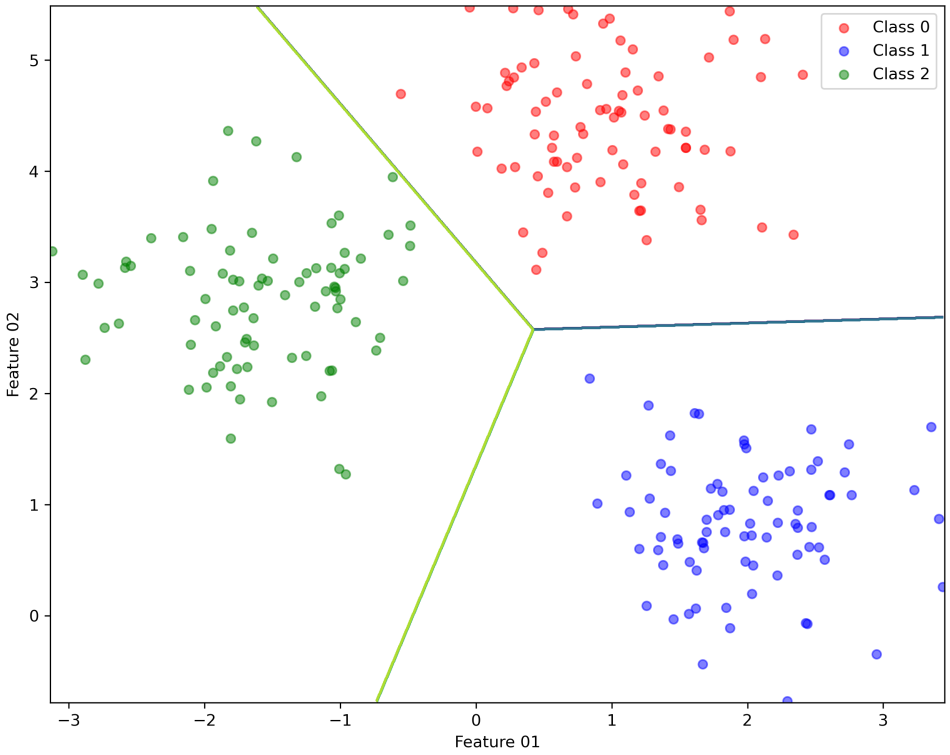
```
LogisticRegression_Model.intercept_
```

```
array([-5.11517997, 3.31237307, 1.8028069 ])
```

```
LogisticRegression_Model.coef_
```

```
array([[ 0.99653519, 1.82291687], [ 1.11680451, -1.46827622], [-2.1133397 , -0.35464065]])
```

Multi-class Classification
Step 04: Test the ML model



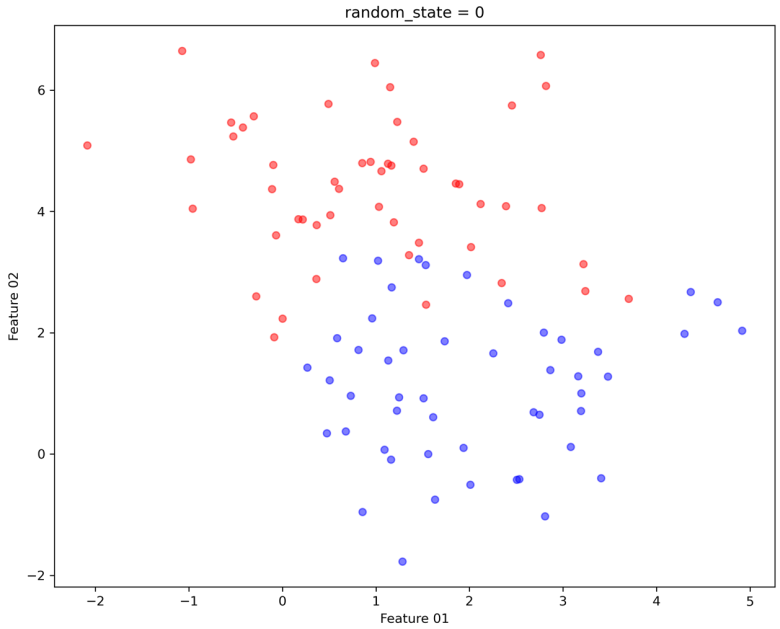
Implementation Using Scikit-Learn

Topic 03

GaussianNB

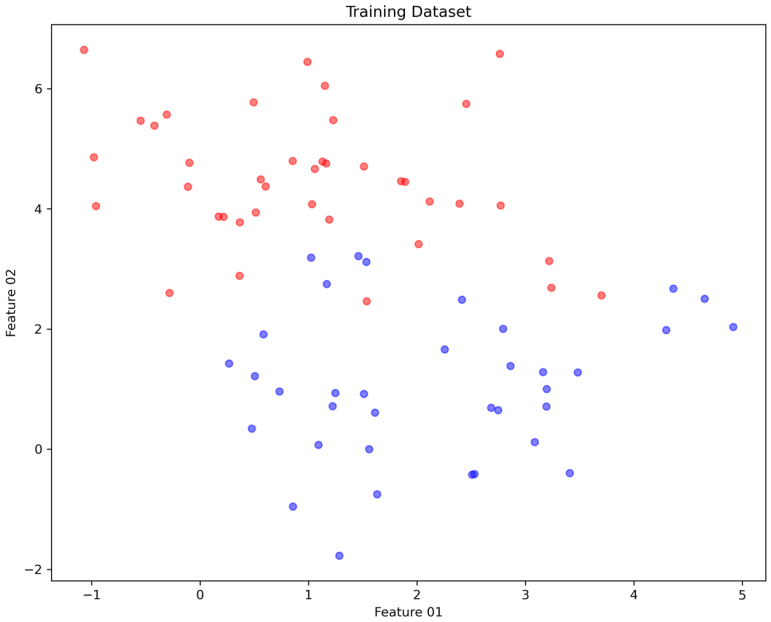
Two-class Classification
Step 01: Dataset Preparation

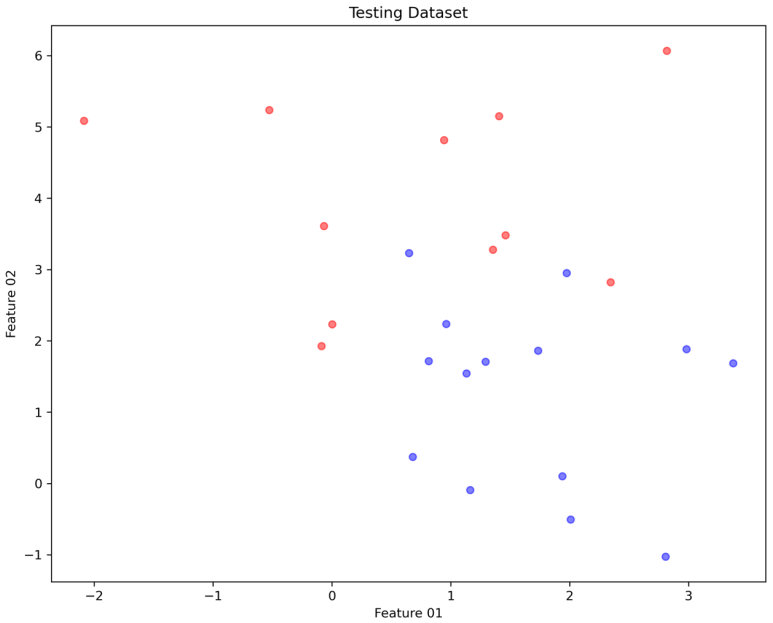
```
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples = 100, n_features = 2, centers = 2, cluster_std = 1.2, random_state = 0)
```



Two-class Classification
Step 02: Training/Testing Dataset

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 10)
```





Two-class Classification
Step 03: Build/Train ML Model

- LinearSVC in Scikit-Learn is responsible for building a SVM model when the kernel is linear

```
from sklearn.naive_bayes import GaussianNB
GaussianNB_Model = GaussianNB()
```

Two-class Classification
Step 03: Build/Train ML Model

- The class function ‘fit’ is called to train the model

```
GaussianNB_Model.fit(X_train, y_train)
```

  - in ‘fit’ method, we simply let X_train and y_train as arguments to call this method

Two-class Classification
Step 04: Test the ML model

- We can predict the class of a new data

```
LogisticRegression_Model.predict([[3,-2]])
```

  - simply put this new data point in the argument of 'predict' method
  - the output looks like

```
array([1])
```

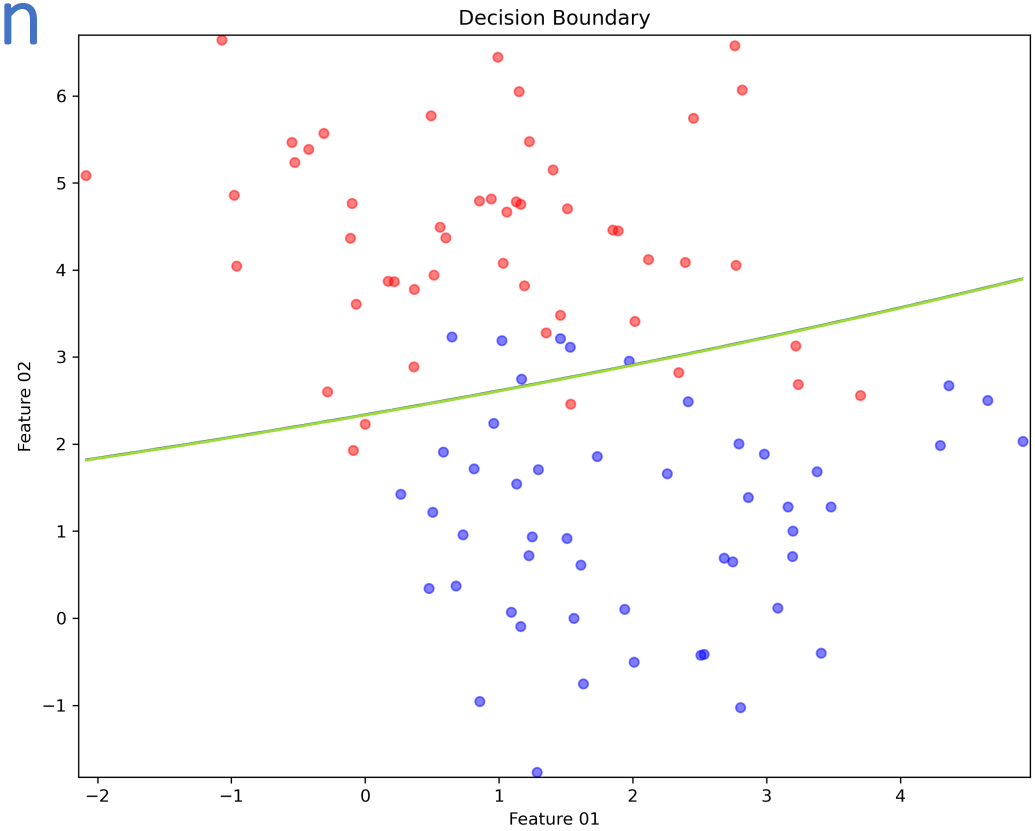
- We can also obtain the accuracy of a dataset

```
LogisticRegression_Model.score(X_test, y_test)
```

  - the arguments in score are the data and its corresponding target
  - the output looks like

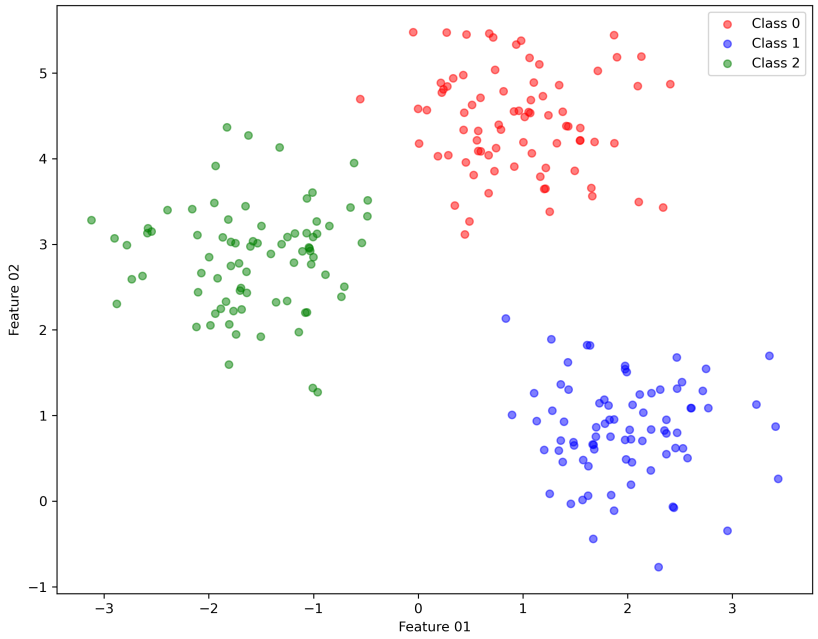
```
0.8
```

Two-class Classification Discussion



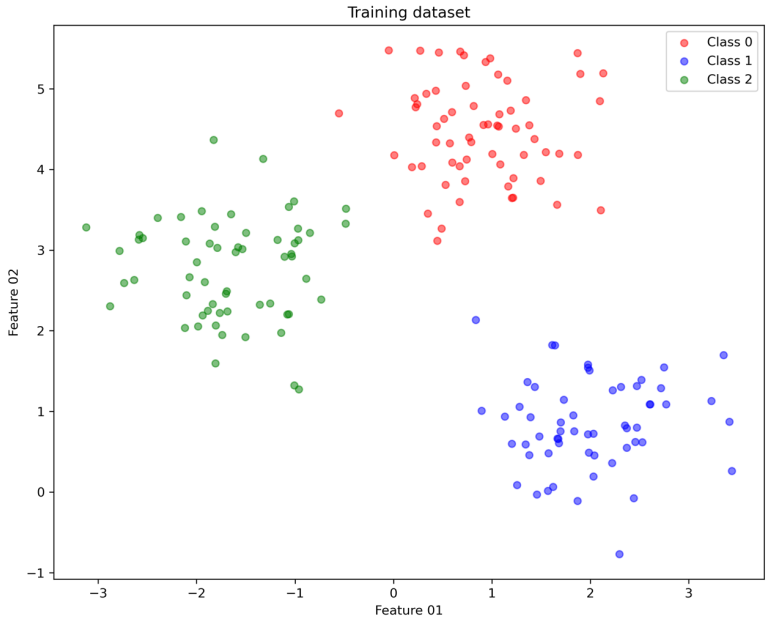
Multi-class Classification
Step 01: Dataset Preparation

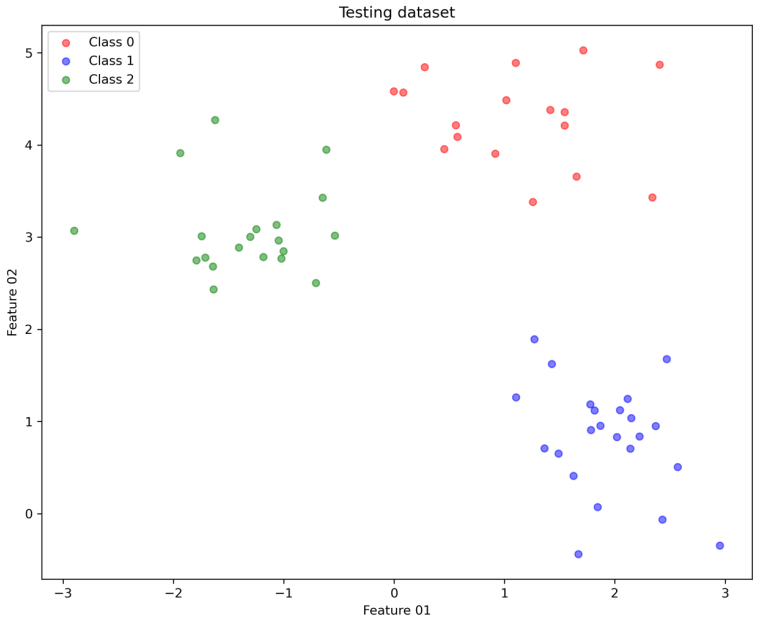
```
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples = 240, n_features = 2, centers = 3, cluster_std = 0.6, random_state = 0)
```



Multi-class Classification
Step 02: Training/Testing Dataset

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 10)
```





Multi-class Classification
Step 03: Build/Train ML Model

```
from sklearn.naive_bayes import GaussianNB
GaussianNB_Model = GaussianNB()
GaussianNB_Model.fit(X_train, y_train)
```

Multi-class Classification
Step 04: Test the ML model

- We can predict the class of a new data

```
GaussianNB_Model.predict([[6,5]])
```

  - simply put this new data point in the argument of 'predict' method
  - the output looks like

```
array([0])
```

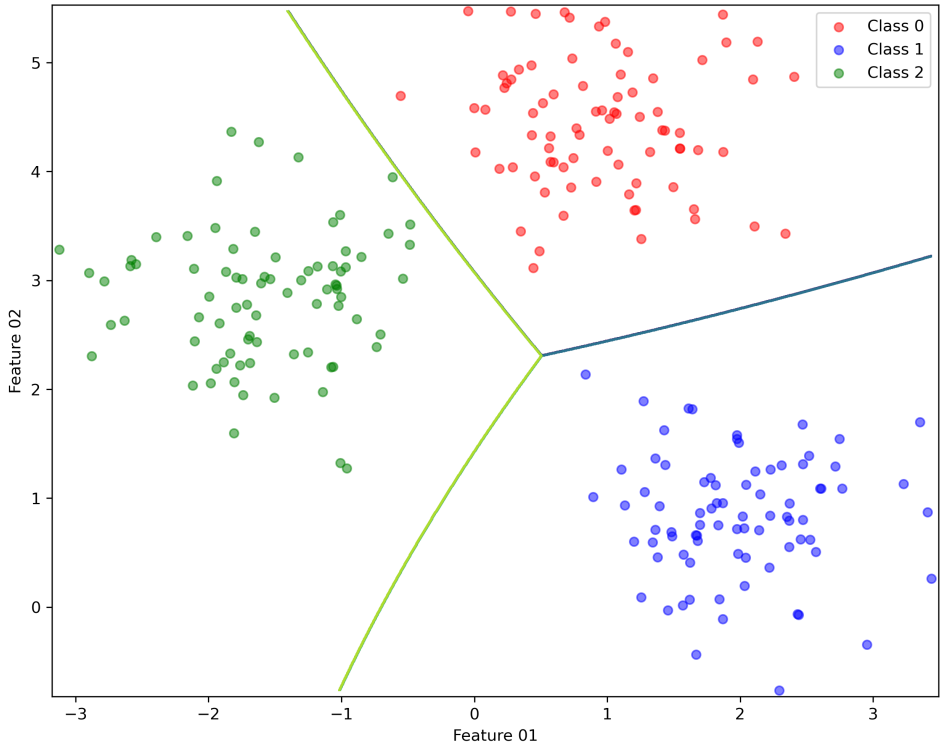
- We can also obtain the accuracy of a dataset

```
GaussianNB_Model.score(X_test, y_test)
```

  - the arguments in score are the data and its corresponding target
  - the output looks like

```
1.0
```

Multi-class Classification
Step 04: Test the ML model



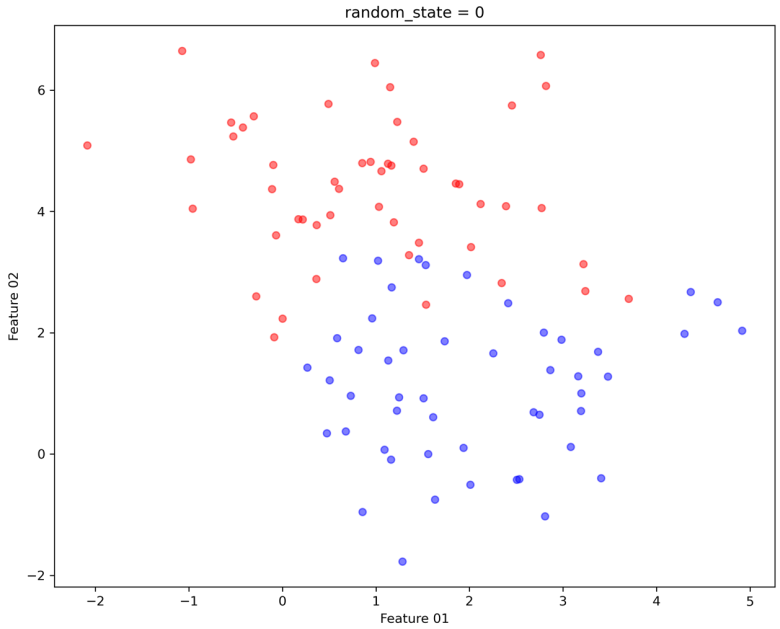
Implementation Using Scikit-Learn

Topic 04

SVC

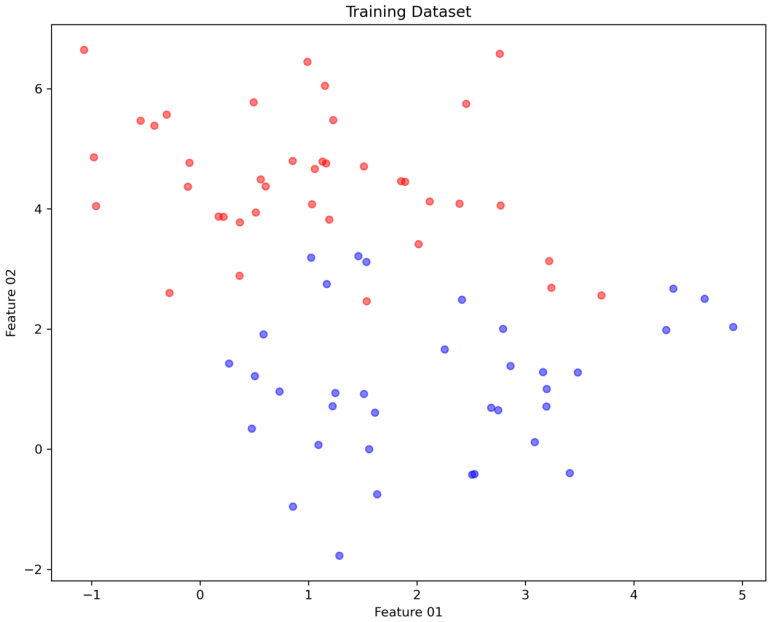
Two-class Classification
Step 01: Dataset Preparation

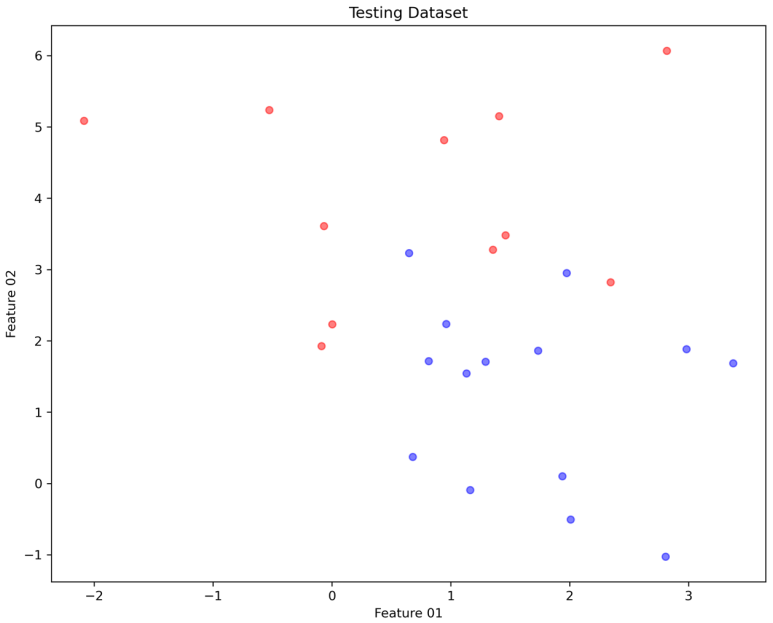
```
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples = 100, n_features = 2, centers = 2, cluster_std = 1.2, random_state = 0)
```



Two-class Classification
Step 02: Training/Testing Dataset

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 10)
```





Two-class Classification
Step 03: Build/Train ML Model

- SVC in Scikit-Learn is responsible for building a general SVM model

```
from sklearn.svm import SVC
SVC_Model = SVC(kernel='rbf', C=1, gamma='scale', max_iter=10000)
```

- 'kernel' is used to specifies the kernel type
  - it can be 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'
- 'C' is used to specify the strength of regularization
  - smaller 'C' means stronger regularization
- 'gamma' is the kernel coefficient for 'rbf', 'poly' and 'sigmoid'
- 'max_iter' is the maximum number of iteration when searching for the optimal weights

Two-class Classification
Step 03: Build/Train ML Model

- The class function ‘fit’ is called to train the model

```
SVC_Model.fit(X_train, y_train)
```

  - in ‘fit’ method, we simply let X_train and y_train as arguments to call this method

Two-class Classification
Step 04: Test the ML model

- We can predict the class of a new data

```
LinearSVC_Model.predict([[6,5]])
```

  - simply put this new data point in the argument of 'predict' method
  - the output looks like

```
array([1])
```

- We can also obtain the accuracy of a dataset

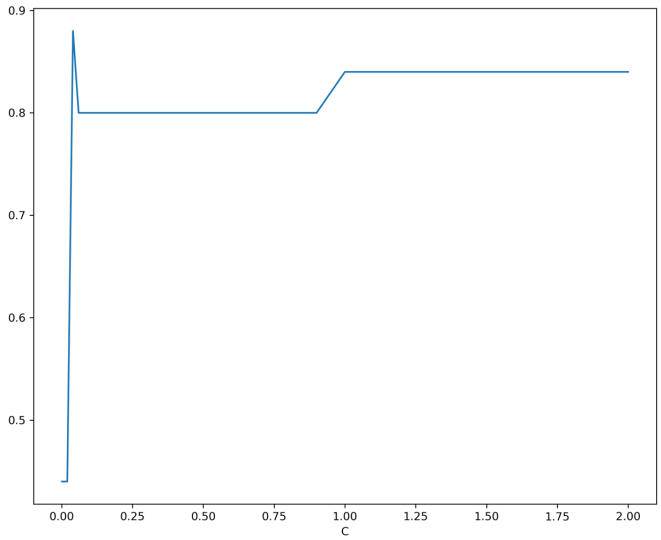
```
LinearSVC_Model.score(X_test, y_test)
```

  - the arguments in score are the data and its corresponding target
  - the output looks like

```
0.84
```

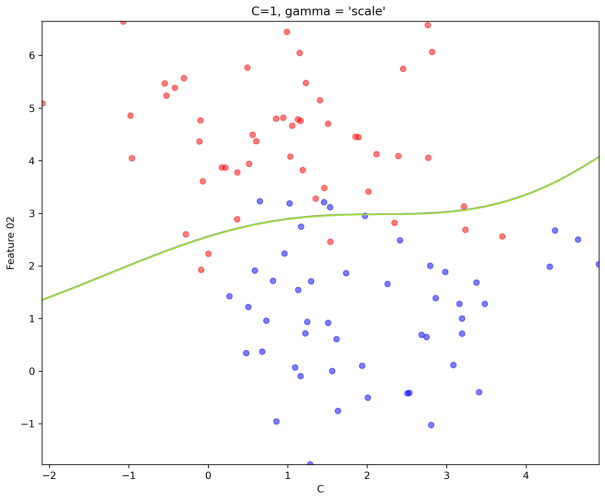
Two-class Classification Discussion

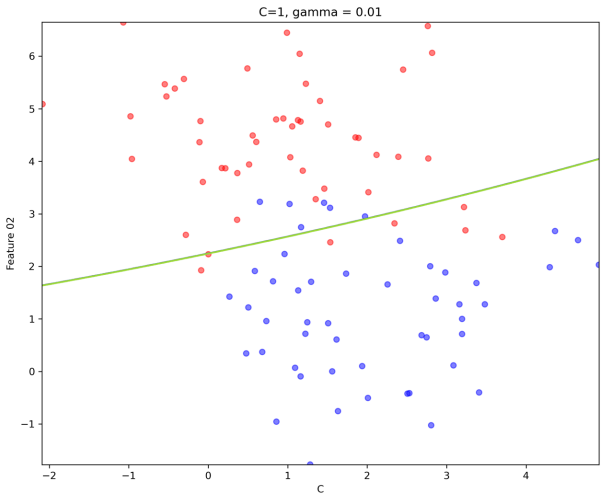
- How does ‘C’ affect the accuracy?

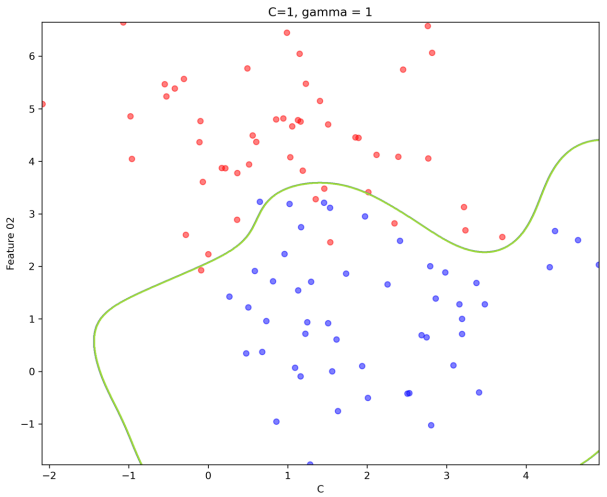


Two-class Classification Discussion

- How does ‘gamma’ affect the model?

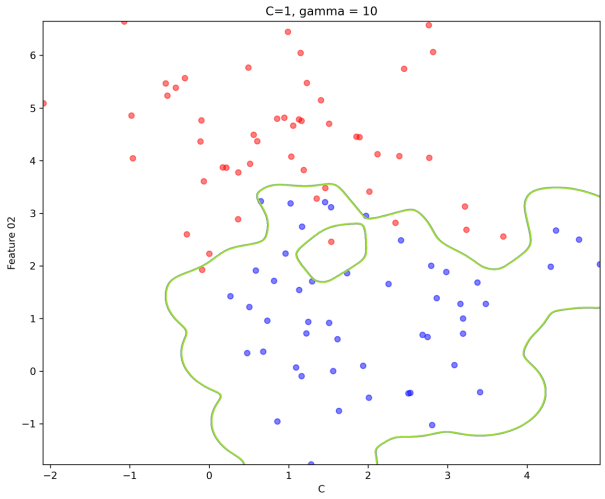


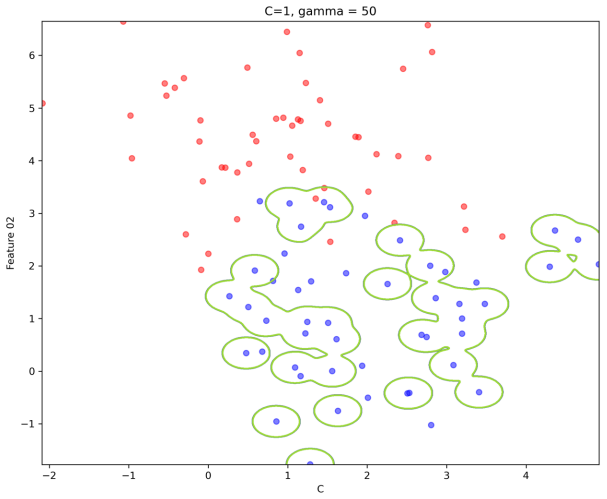




Two-class Classification Discussion

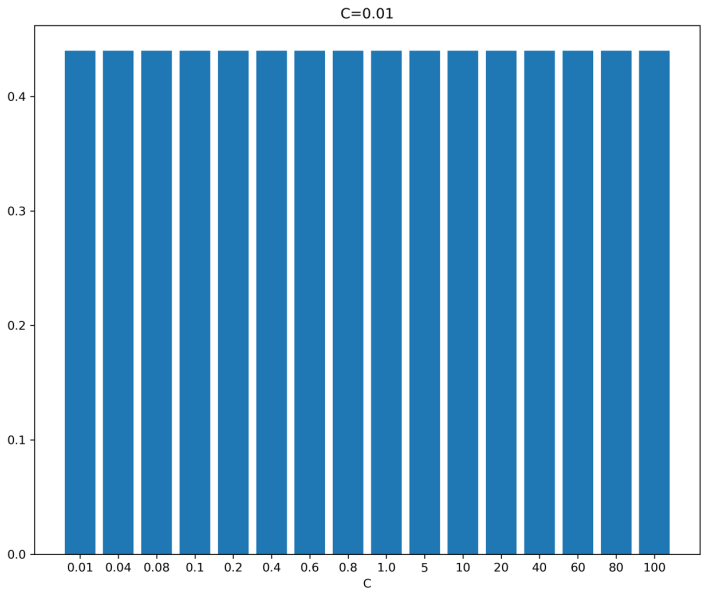
- How does ‘gamma’ affect the model?

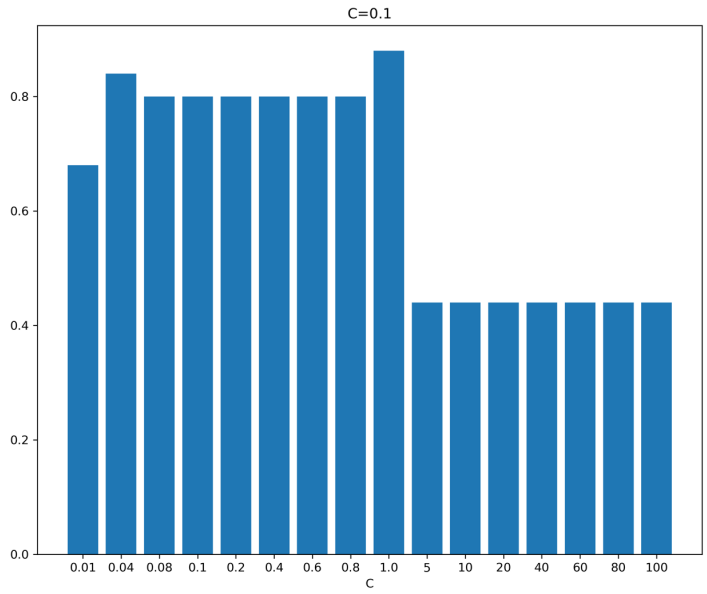




Two-class Classification Discussion

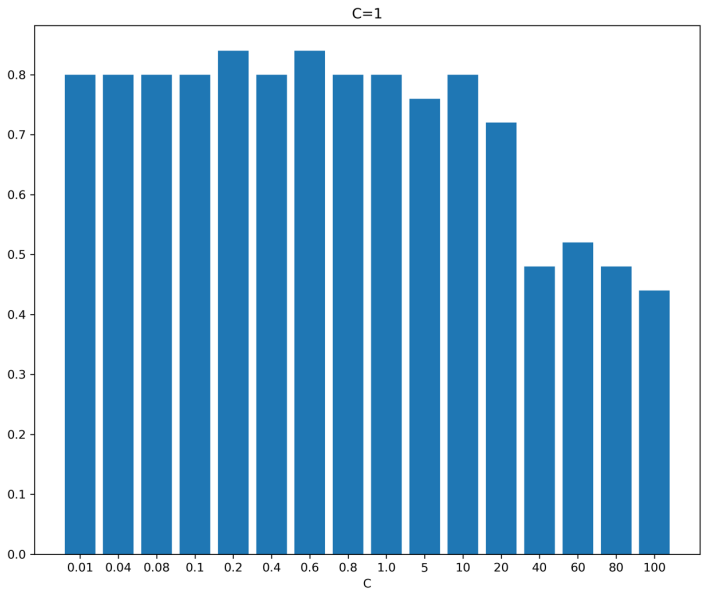
- How does ‘gamma’ affect the model?

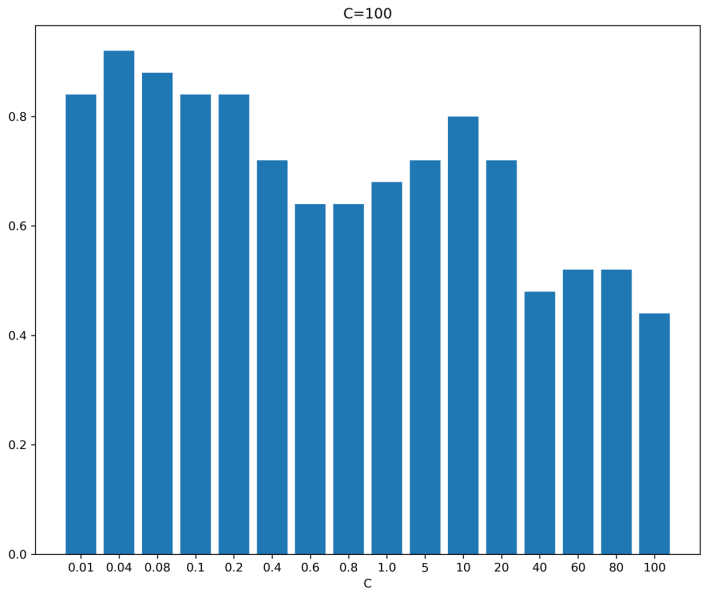




Two-class Classification Discussion

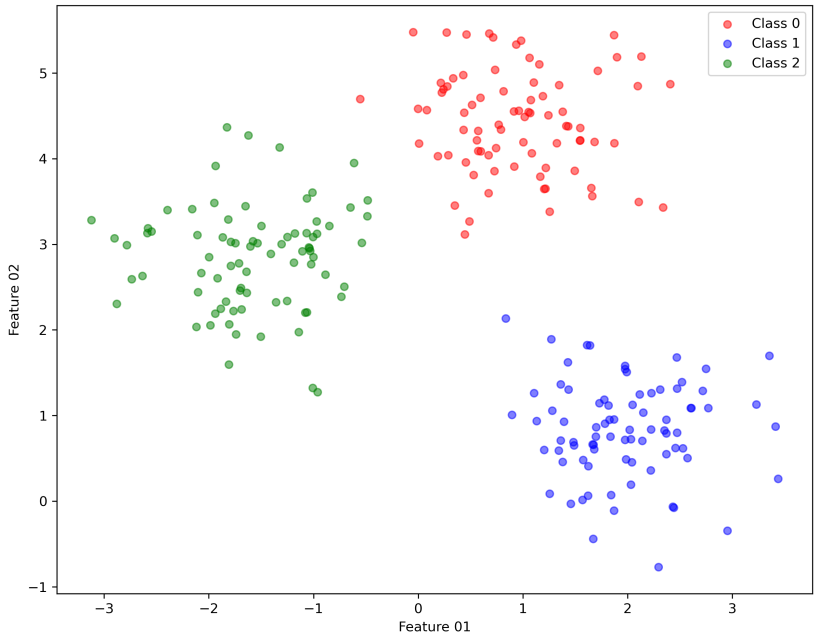
- How does ‘gamma’ affect the model?





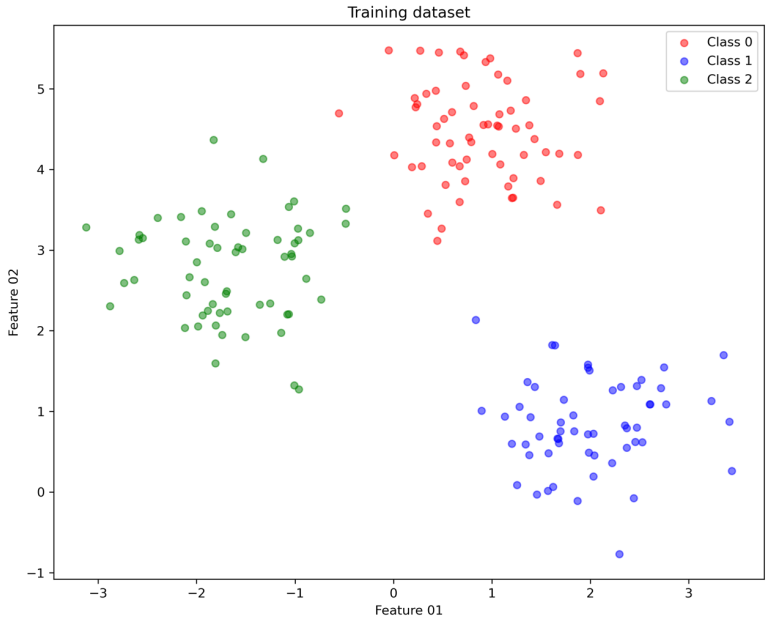
Multi-class Classification
Step 01: Dataset Preparation

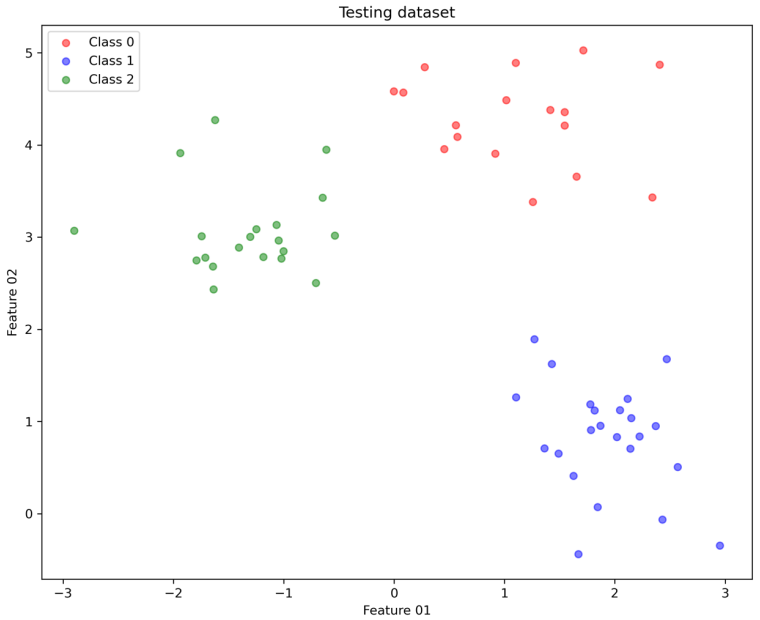
```
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples = 240, n_features = 2, centers = 3, cluster_std = 0.6, random_state = 0)
```



Multi-class Classification
Step 02: Training/Testing Dataset

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 10)
```





Multi-class Classification
Step 03: Build/Train ML Model

```
from sklearn.svm import SVC

SVC_Model = SVC(kernel='rbf', C=1, gamma='scale', max_iter=10000)
SVC_Model.fit(X_train, y_train)
```

Multi-class Classification
Step 04: Test the ML model

- We can predict the class of a new data

```
SVC_Model.predict([[6,5]])
```

  - simply put this new data point in the argument of 'predict' method
  - the output looks like

```
array([0])
```

- We can also obtain the accuracy of a dataset

```
SVC_Model.score(X_test, y_test)
```

  - the arguments in score are the data and its corresponding target
  - the output looks like

```
1.0
```

Multi-class Classification
Step 04: Test the ML model

- In the class method, we have a method called 'decision_function'

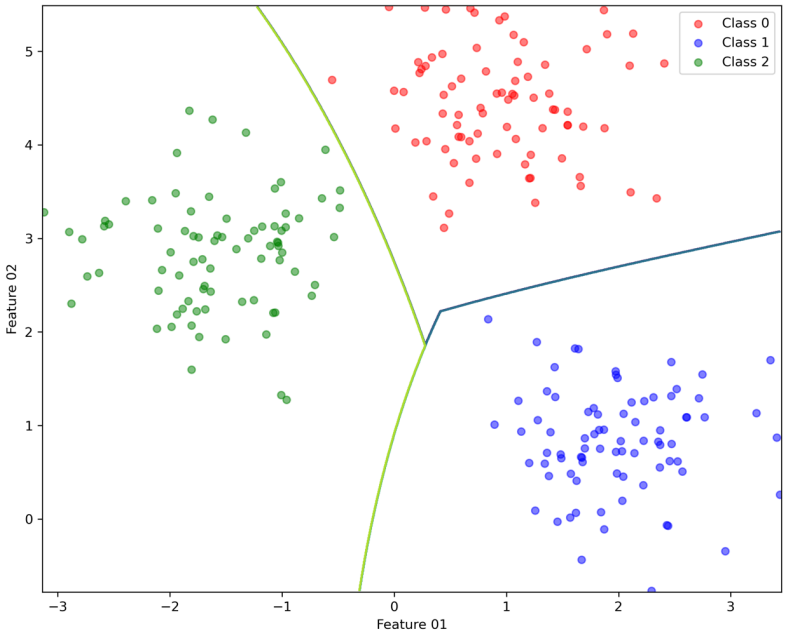
```
SVC_Model.decision_function([[6,5]])
```

  - its output is

```
array([ 0.04932277, 0.09157022, -0.10604332])
```

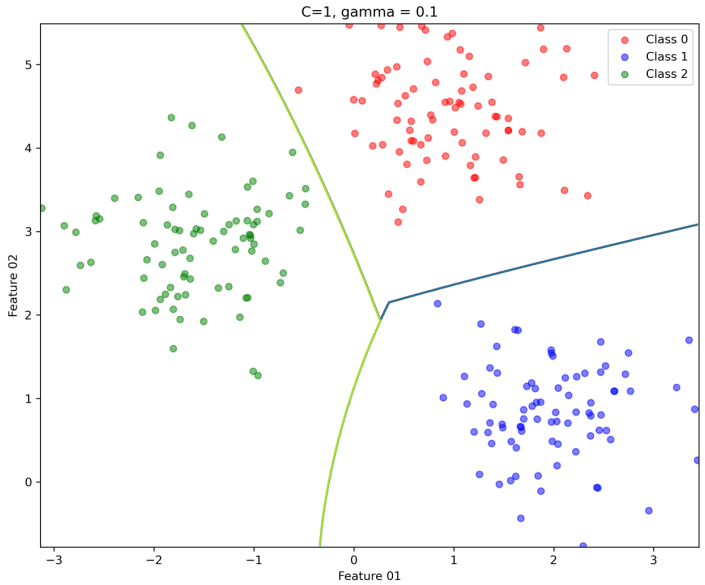
  - these three represents the values of discriminant functions of Class 0, 1, and 2, respectively
  - since Class 0 has the largest value, [6,5] is classified into Class 0

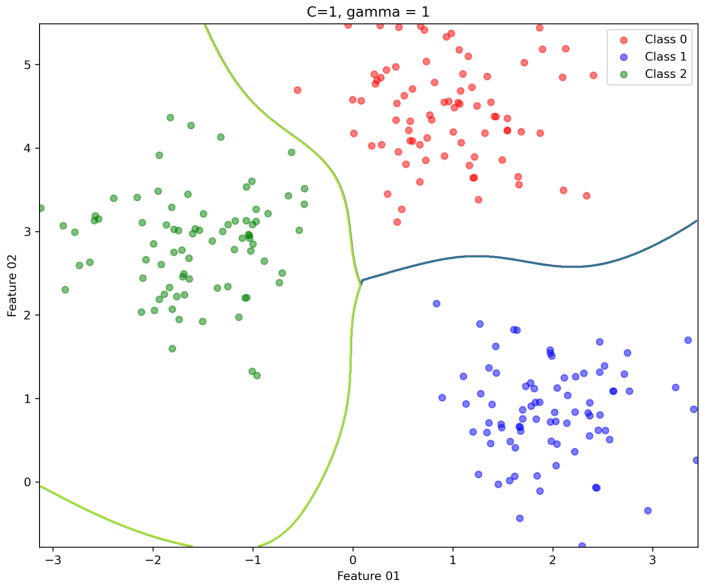
Multi-class Classification
Step 04: Test the ML model



Multi-class Classification Discussion

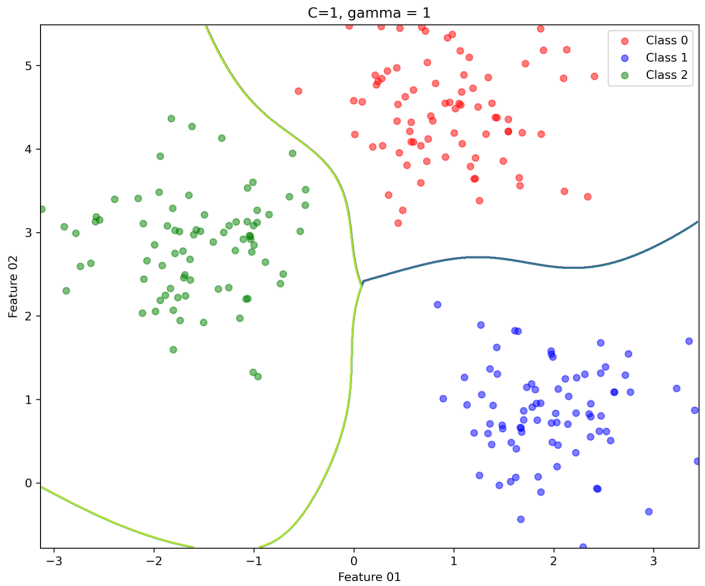
- How does ‘gamma’ affect the model?

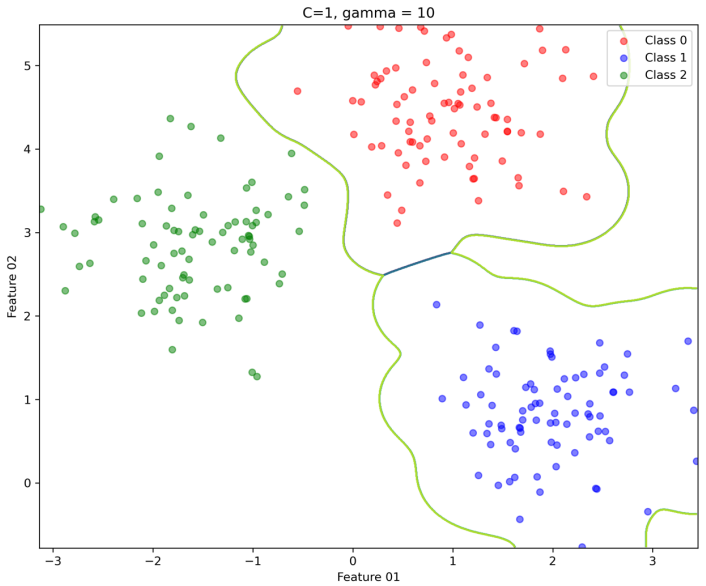




Multi-class Classification Discussion

- How does ‘gamma’ affect the model?





Q&A