

Capstone

- Please contact the instructor for the corresponding projects (**first come first serve**).
- Your instructor will help you to define what should add into the **first Phase report** which is due this Friday.

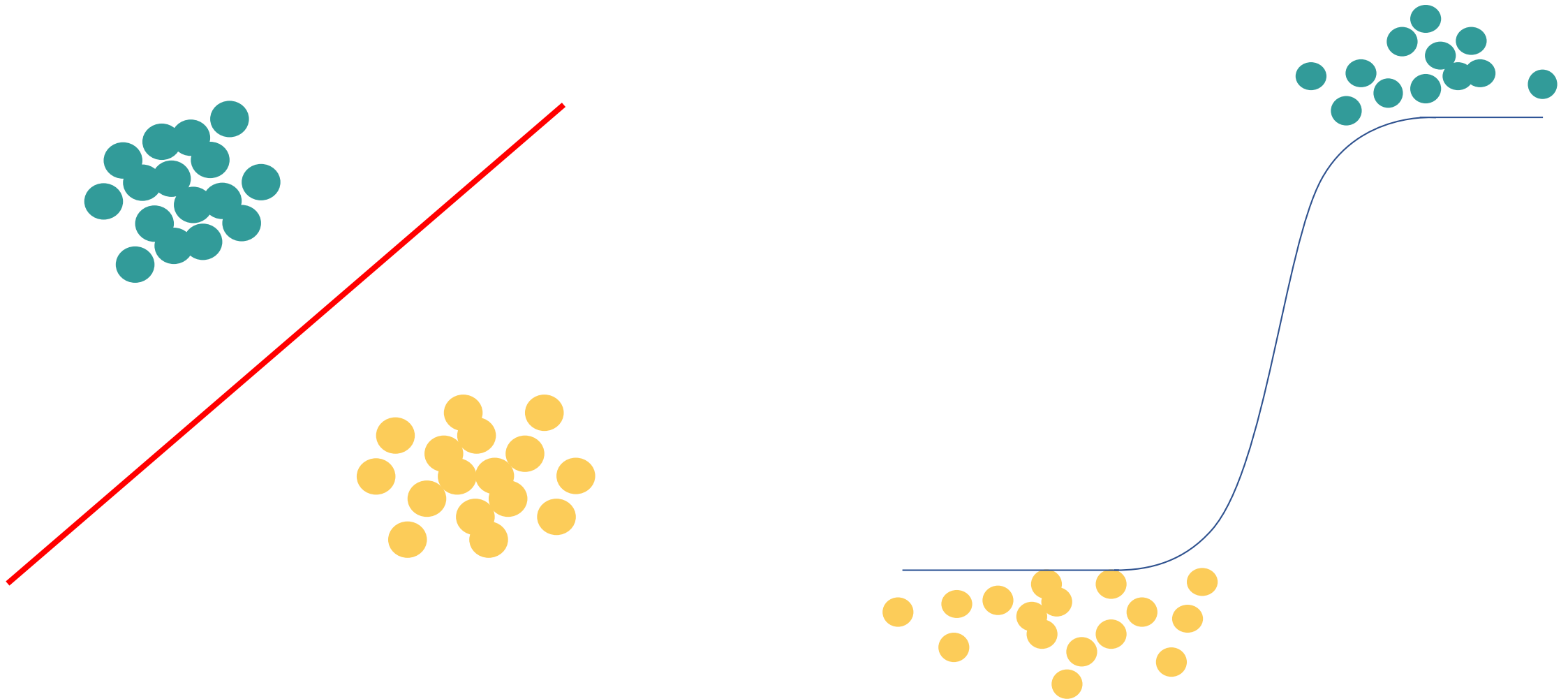
- Dr. Chase Rainwater: cer@uark.edu
- Dr. ShengfanZhang: shengfan@uark.edu
- Dr. Khoa Luu: khoaluu@uark.edu
- Dr. Ngan Le: thile@uark.edu

Applied Machine Learning

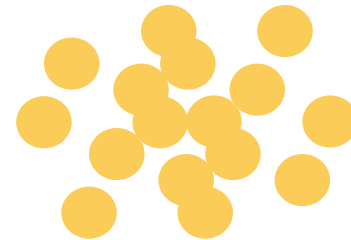
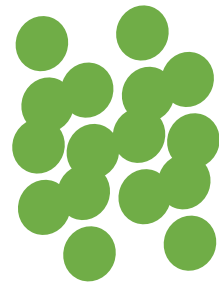
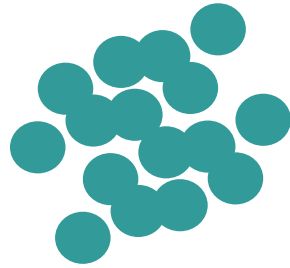
Classification: Multiclass Classification

Ngan Le
thile@uark.edu

Binary- Classification



Multiclass Classification



Multiclass Classification

OvA: One vs. All (One vs. Rest)

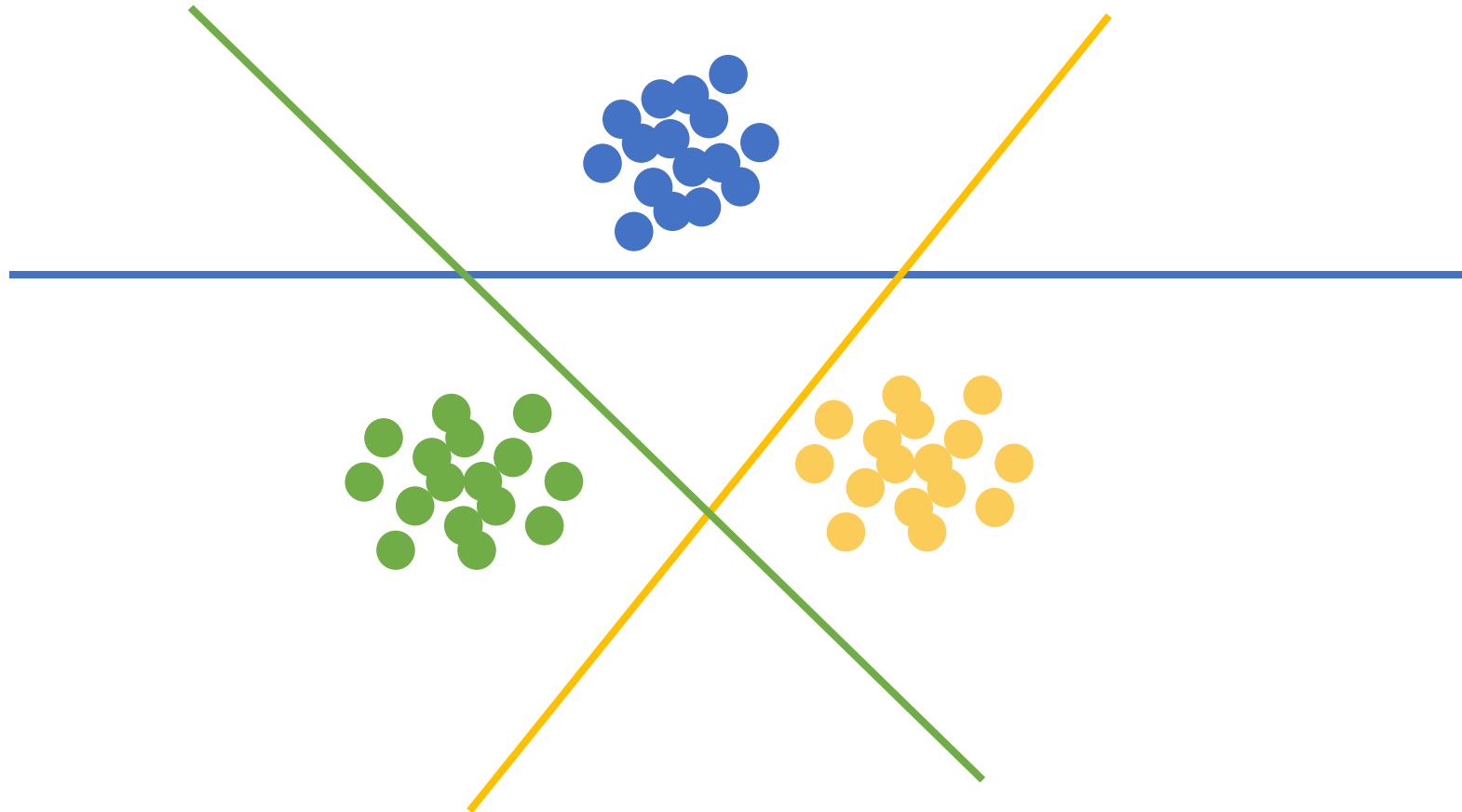
splits the dataset into one binary dataset for each class

OvO: One vs. One

splits the dataset into one dataset for each class versus every other class

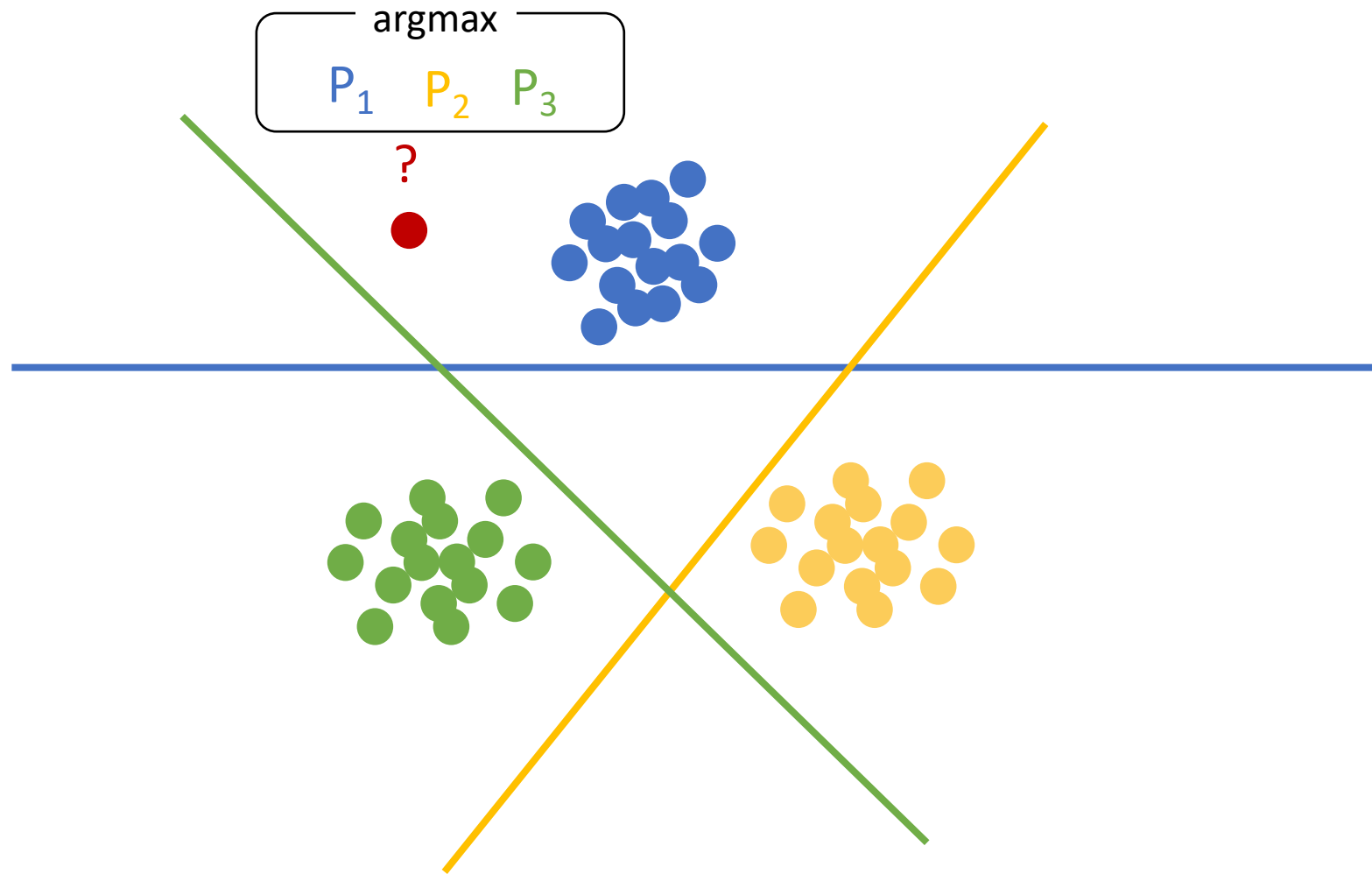
Multiclass Classification: OvA

splits the dataset into one binary dataset for each class



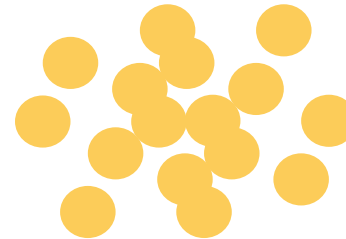
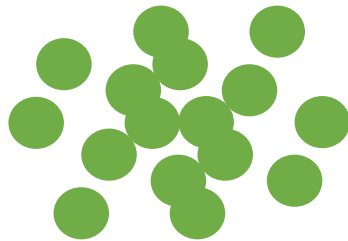
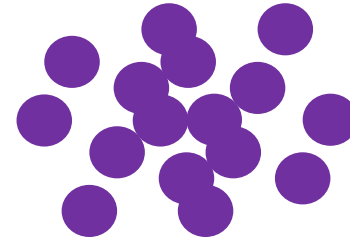
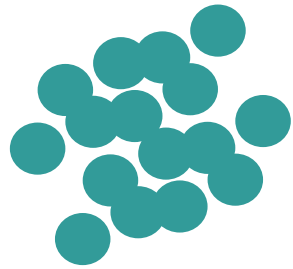
Multiclass Classification: OvA

splits the dataset into one binary dataset for each class



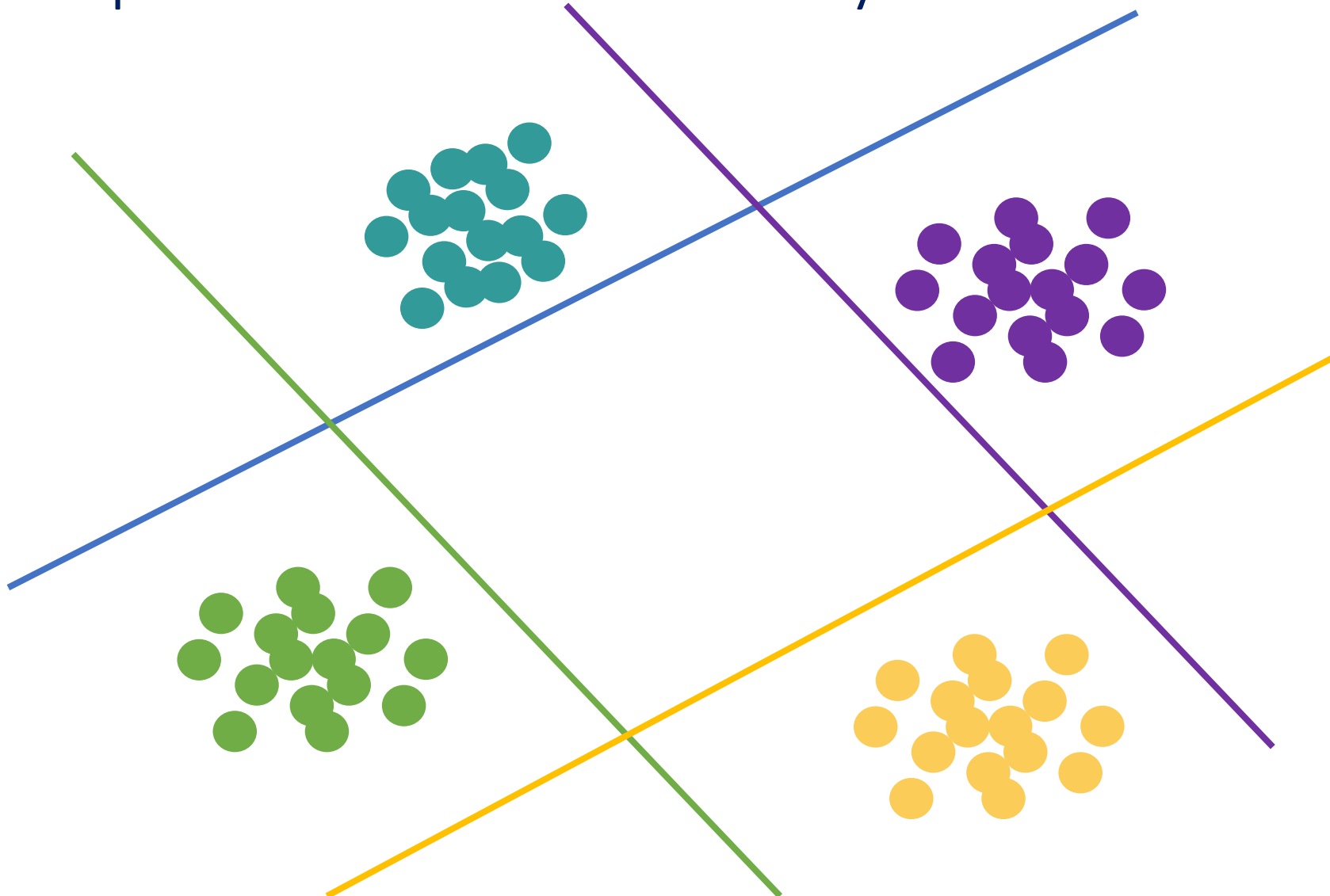
Multiclass Classification: OvA

splits the dataset into one binary dataset for each class



Multiclass Classification: OvA

splits the dataset into one binary dataset for each class



Multiclass Classification: OvA

```
from sklearn.linear_model import LogisticRegression
```

```
model = LogisticRegression(multi_class='ovr')
```

```
model.fit(X, y)
```

`linear_model.LogisticRegression`([penalty, ...]) Logistic Regression (aka logit, MaxEnt) classifier.

`linear_model.LogisticRegressionCV`(*[, Cs, ...]) Logistic Regression CV (aka logit, MaxEnt) classifier.

`linear_model.PassiveAggressiveClassifier`(*) Passive Aggressive Classifier

`linear_model.Perceptron`(*[, penalty, alpha, ...]) Read more in the [User Guide](#).

`linear_model.RidgeClassifier`([alpha, ...]) Classifier using Ridge regression.

`linear_model.RidgeClassifierCV`([alphas, ...]) Ridge classifier with built-in cross-validation.

`linear_model.SGDClassifier`([loss, penalty, ...]) Linear classifiers (SVM, logistic regression, etc.) with SGD

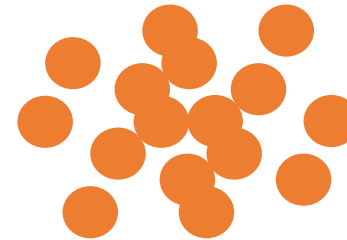
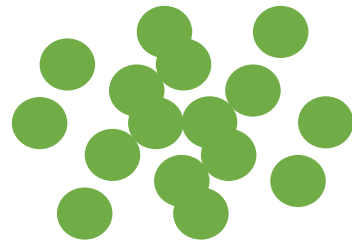
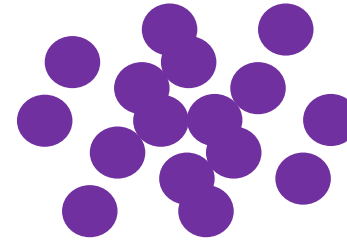
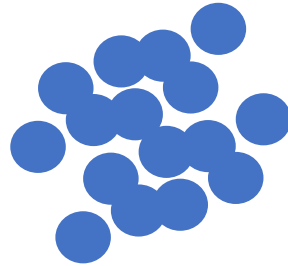
```
model = LogisticRegression()
```

```
ovr = OneVsRestClassifier(model)
```

```
ovr.fit(X, y)
```

Multiclass Classification: OvO

splits the dataset into one dataset for each class versus every other class



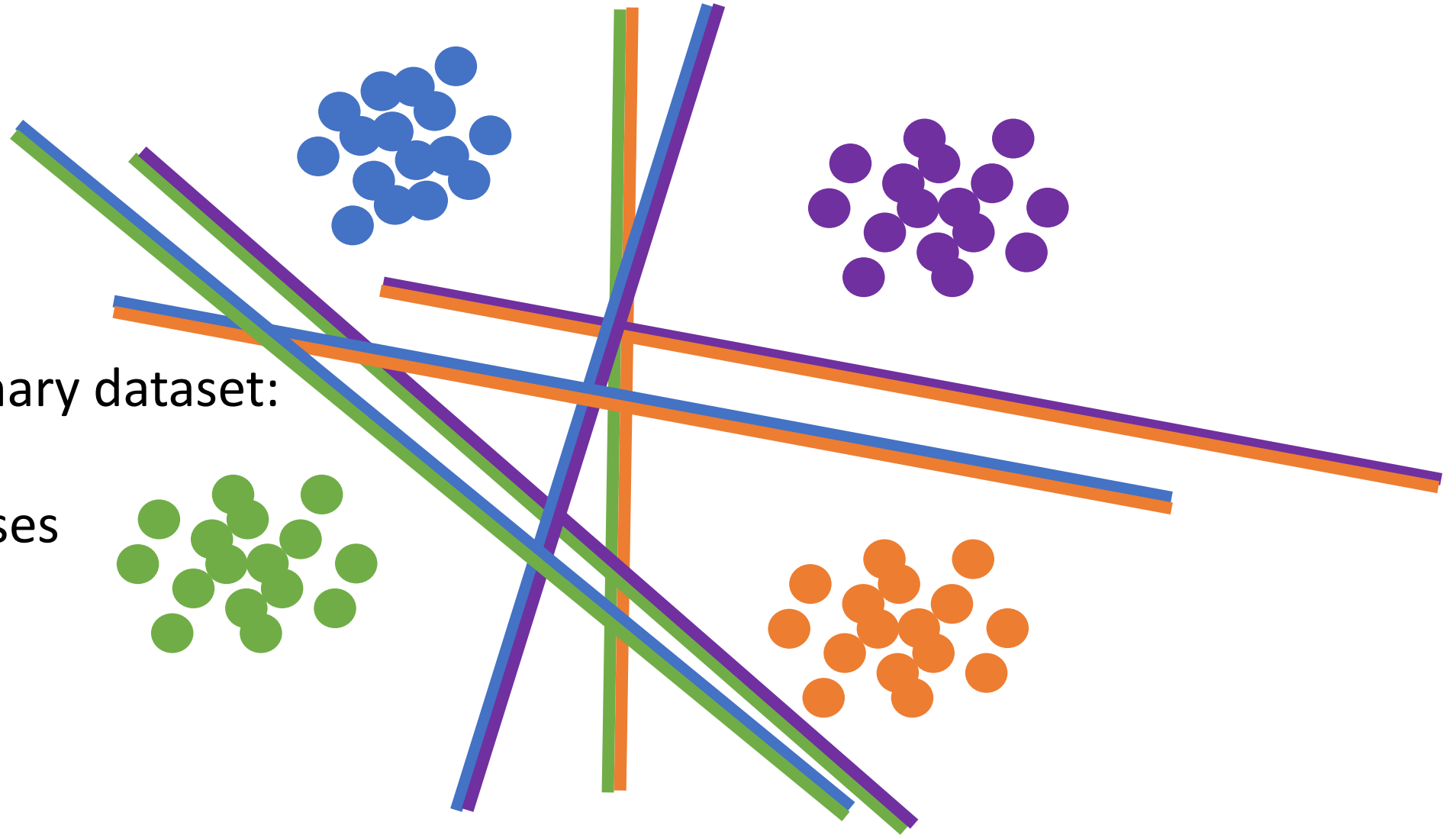
Multiclass Classification: OvO

splits the dataset into one dataset for each class versus every other class

The number of binary dataset:

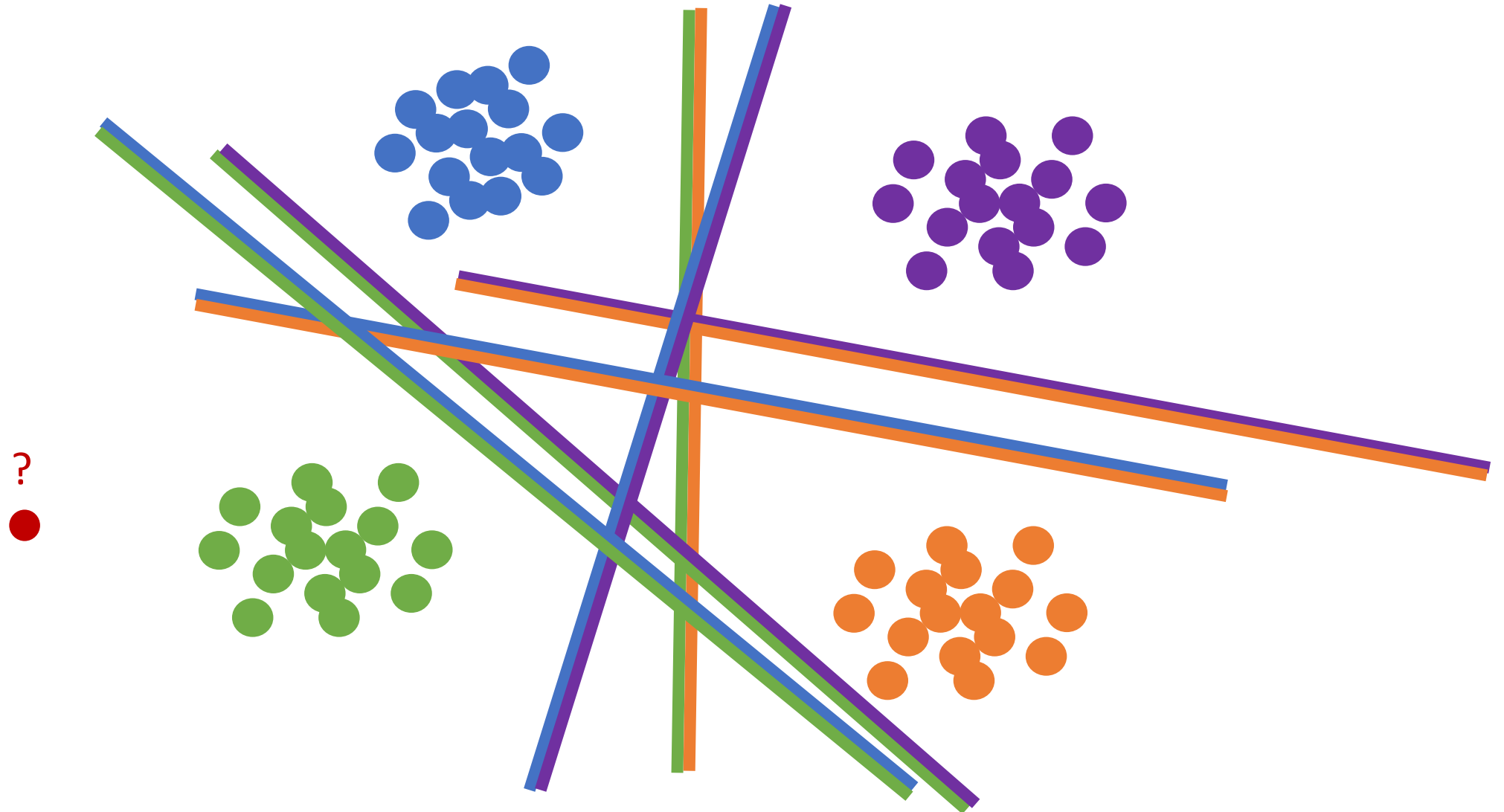
$$(K * (K - 1)) / 2$$

K: Number of classes



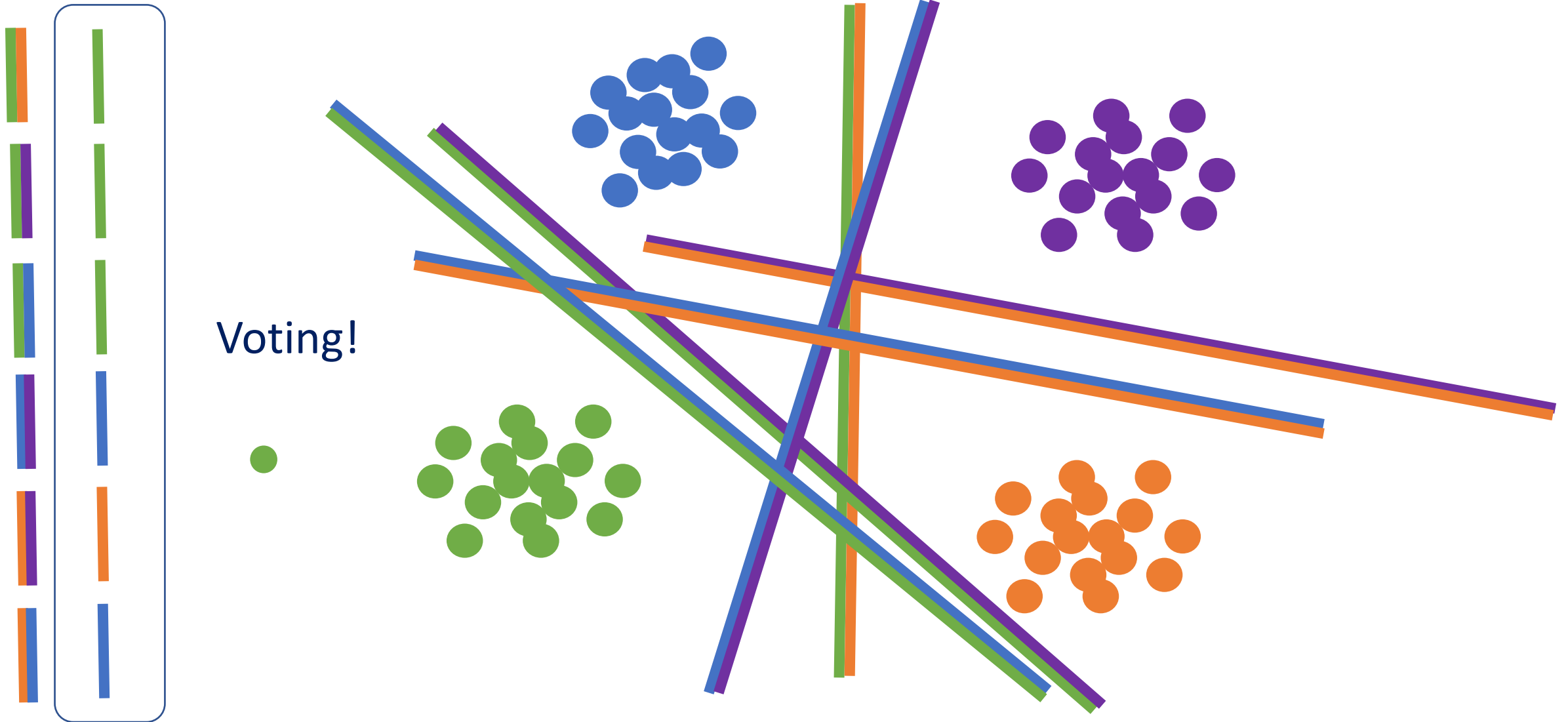
Multiclass Classification: OvO

splits the dataset into one dataset for each class versus every other class



Multiclass Classification: OvO

splits the dataset into one dataset for each class versus every other class



Multiclass Classification: OvO

```
from sklearn.svm import SVC  
  
model = SVC(decision_function_shape='ovo')  
  
model.fit(X, y)
```

```
from sklearn.svm import SVC  
  
model = SVC()  
  
ovo = OneVsOneClassifier(model)  
  
ovo.fit(X, y)
```

Lab



Input:

Iris Dataset: comes packaged with scikit-learn
contains feature measurements of three different species of iris flowers.
The features are sepal length, sepal width, petal length, and petal width;

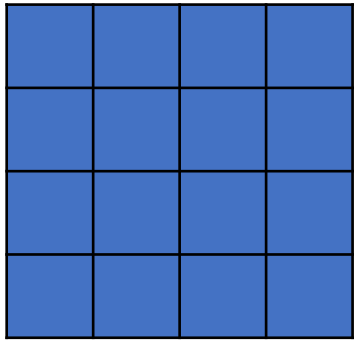
Target:

Is the species of iris flower (class 1: Iris Setosa, class 2: Iris Versicolor, class 3: Iris Virginica).

Cross Fold-validation

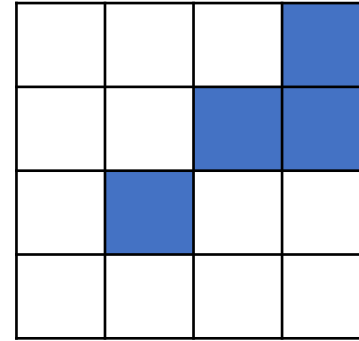
1. Randomly shuffle the training data.
2. Split the dataset into k groups. For each group do the following:
 1. Use the group as a test set.
 2. Use the remaining data as a training set.
 3. Train a model on the training data, and evaluate using the test set.
 4. Record the model's overall performance (with whatever metric you're using), and scrap the model.
 5. Repeat for all k groups.

Cross Fold-validation

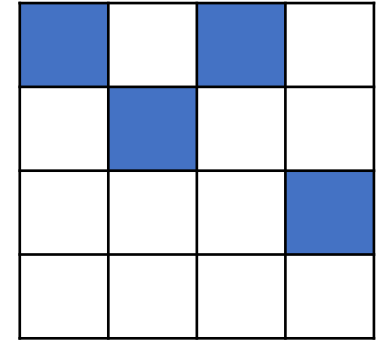


16 samples (data points)

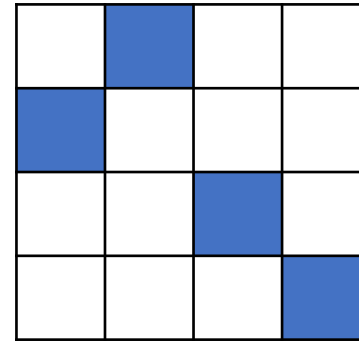
4-fold cross validation



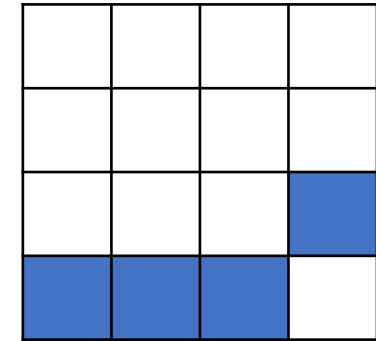
Fold 1



Fold 2



Fold 3



Fold 4

Model 1: Train on Fold 1 + Fold 2 + Fold 3. Test on Fold 4

Model 2: Train on Fold 1 + Fold 2 + Fold 4. Test on Fold 3

Model 3: Train on Fold 1 + Fold 3 + Fold 4. Test on Fold 2

Model 4: Train on Fold 2 + Fold 3 + Fold 4. Test on Fold 1

Cross Fold-validation

```
from sklearn.model_selection import KFold  
  
kfold = KFold(4, True, 1)  
  
for train, test in kfold.split(data):
```

```
from sklearn.model_selection import cross_val_score
```

Lab



Objective: create a classifier that identifies the producer of a wine based on various properties of the wine

This exercise will have minimal guidance and will allow you to really demonstrate your machine learning skills.