COMP9313

Week 8

登登教育

-Elijah-

20T2

本节课主要是代码实操，为 Project 2 做准备

DataFrame

Naive Bayes

Stacking

DataFrame

DataFrame

- A DataFrame is a distributed collection of data organized into named columns.
- The DataFrames API is:
  - intended to enable wider audiences beyond “Big Data” engineers to leverage the power of distributed processing
  - inspired by data frames in R and Python (Pandas)
  - designed from the ground-up to support modern big data and data science applications
  - an extension to the existing RDD API

DataFrame 与 RDD 相比较：
结构上最大的不同就是有了 Column Name

但是注意！没有 Row Name，每一行为 Row( ) 的类型

创建的方式：可以读取文件直接得到，也可以从 RDD 得到

```
spark = SparkSession.builder.config(conf=conf).getOrCreate()
df = spark.read.format(”json”).load(”example.json")
```

```
trainingData = [["Chinese Beijing Chinese", "c"],\
                ["Chinese Chinese Nanjing", "c"],\
                ["Chinese Macao", "c"],\
                ["Australia Sydney Chinese","o"],\
               ]
```

trainRDD = trainRDD.map(lambda e: Row(descript=e[0], category=e[1]))

trainDF = spark.createDataFrame(trainRDD)

DataFrame 的操作，一会看演示

Naive Bayes

基于 text classification

的Naive bayes 分为两种：

1. Multinomial
2. Multivariate Bernoulli

Naïve Bayes Classifier

- Bayes' Rule:
  - For a document d and a class c

posteriori priori

$$P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)}$$

- We want to which class is most likely

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(c \mid d)$$

Example of Naïve Bayes Classifier

	Document	Words	Class
Training	1	Chinese Beijing Chinese	c
	2	Chinese Chinese Nanjing	c
	3	Chinese Macao	c
	4	Australia Sydney Chinese	o
Test	5	Chinese Chinese Chinese Australia Sydney	?

$$P(c_j) \neg \frac{|docs_j|}{|\text{total \# documents}|} \qquad P(w_k|c_j) \neg \frac{n_k + a}{n + a|Vocabulary|}$$

$$\mathrm{P(c)} = \frac{3}{4}$$

$$\mathrm{P}(j) = \frac{1}{4}$$

$$\mathrm{P}(Chinese|c) = \frac{5+1}{8+6} = \frac{3}{7}$$

$$\mathrm{P}(Australia|c) = \frac{0+1}{8+6} = \frac{1}{14}$$

$$\mathrm{P}(Sydney|\mathrm{c}) = \frac{0+1}{8+6} = \frac{1}{14}$$

$$\mathrm{P}(Chinese|o) = \frac{1+1}{3+6} = \frac{2}{9}$$

$$\mathrm{P}(Australia|o) = \frac{1+1}{3+6} = \frac{2}{9}$$

$$\mathrm{P}(Sydney|o) = \frac{1+1}{3+6} = \frac{2}{9}$$

$$\mathrm{P(c}|d5) \propto \frac{3}{4} * \left(\frac{3}{7}\right)^3 * \frac{1}{14} * \frac{1}{14} \approx 0.0003$$

$$\mathrm{P}(o|d5) \propto \frac{1}{4} * \left(\frac{2}{9}\right)^3 * \frac{2}{9} * \frac{2}{9} \approx 0.0001$$

不用手算，使用 SparkMLlib 轻松搞定

Classification Process 1: Preprocessing and Feature Engineering

category	descript
MISC	I've been there t...
REST	Stay away from th...
REST	Wow over 100 beer...
MISC	Having been a lon...
MISC	This is a consist...

Raw Data

descript	words
I've been there t...	[i've, been, ther...
Stay away from th...	[stay, away, from...
Wow over 100 beer...	[wow, over, 100, ...
Having been a lon...	[having, been, a,...
This is a consist...	[this, is, a, con...

dense Vector

features	label
(5421,[1,18,31,39...	1.0
(5421,[0,1,15,20,...	0.0
(5421,[3,109,556,...	0.0
(5421,[1,2,3,5,6,...	1.0
(5421,[2,3,4,8,11...	1.0

Training Data

实际得到的是 Sparse Vector

[chinese, australia, sydney,macao, nanjing, beijing]

- `(6,[0,5],[2.0,1.0])` 是 `sparse Vector` 的形式
- 6 = `vocabulary size`
- `[0,5]` 是 index
- `[2.0,1.0]` 是 value
- 等价于 dense Vector `[2.0, 0.0, 0.0, 0.0, 0.0, 1.0]`

Demo

PySpark MLlib – Example of NB

一个 Estimator 需要先 fit（training 的过程）再 transform （use的过程）

- build the pipeline

```
# white space expression tokenizer
WordTokenizer = Tokenizer(inputCol="descript", outputCol="words")

# bag of words count
countVectors = CountVectorizer(inputCol="words", outputCol="features")

# label indexer
label_stringIdx = StringIndexer(inputCol = "category", outputCol =
"label")

# model
nb_model = NaiveBayes(featuresCol='features',
                      labelCol='label',
                      predictionCol='nb_prediction')

# build the pipeline
nb_pipeline = Pipeline(stages=[WordTokenizer, countVectors,
label_stringIdx, nb_model])
```

Demo

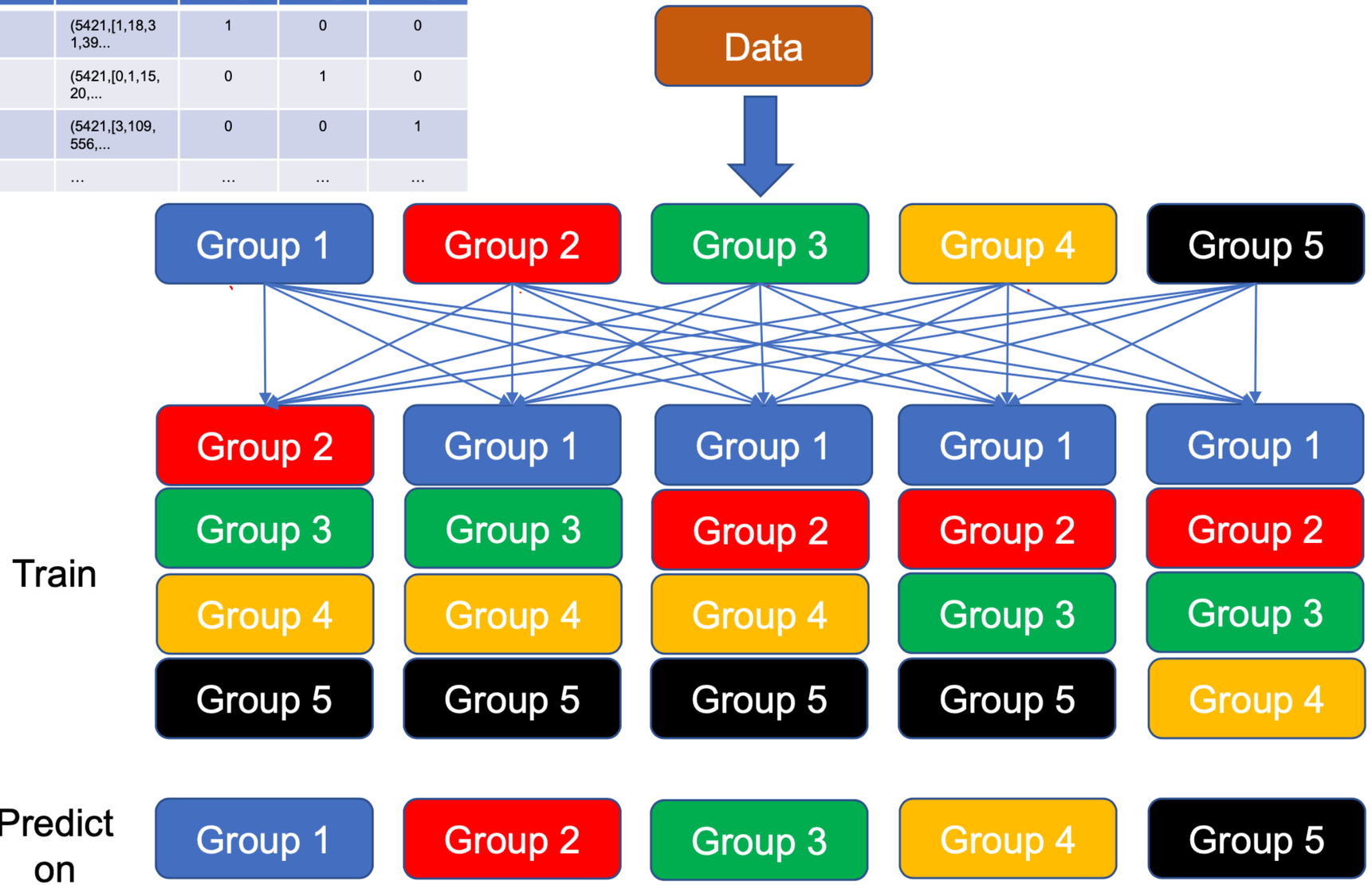
More on Pipeline

- A Transformer takes a dataframe as input and produces an augmented dataframe as output
  - `Tokenizer`
  - `CountVectorizer` 按代码来看其实 CountVector 也是一个 Estimator
- An Estimator must be first fit on the input dataframe to produce a model
  - After fit, we got a Transformer
  - `NaiveBayes`
- Pipelines and PipelineModels help to ensure that training and test data go through identical feature processing steps
  - E.g., when test data contains word that is not in training data

Stacking

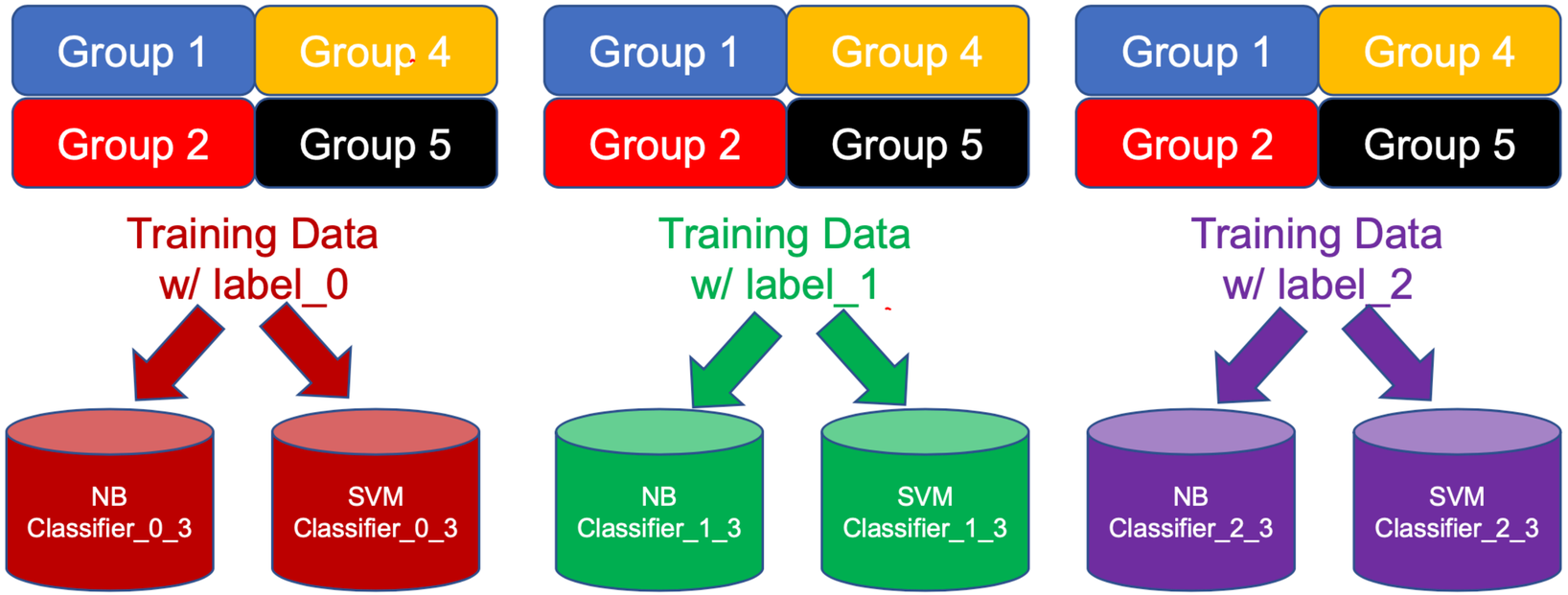
Step 1: prepare training data for base models

Group	features	label_0	label_1	label_2
1	(5421,[1,18,31,39...	1	0	0
4	(5421,[0,1,15,20,...	0	1	0
2	(5421,[3,109,556,...	0	0	1
…	…	…	…	…



对于每一个 Group

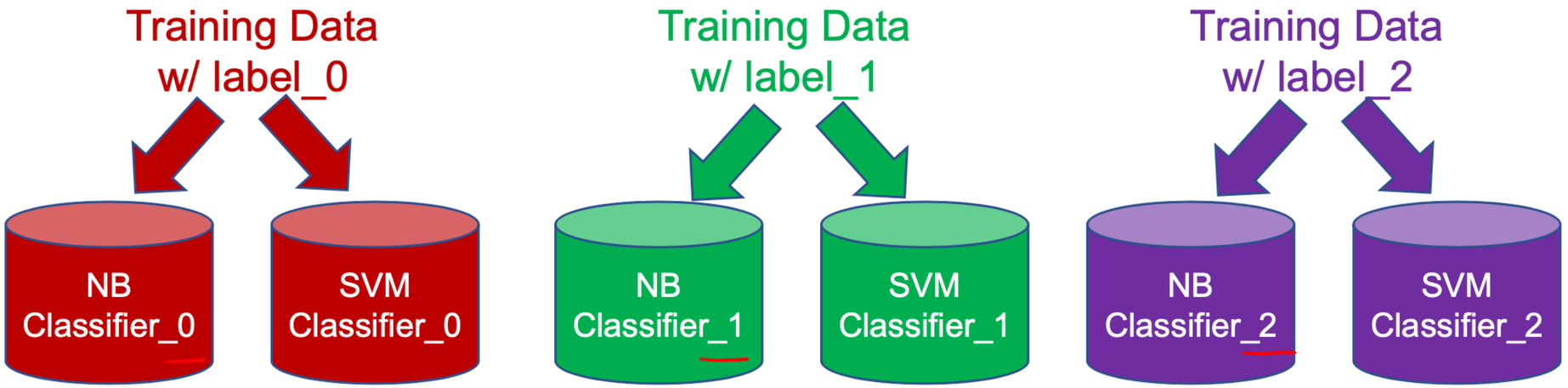
Step 2: learn base classifiers



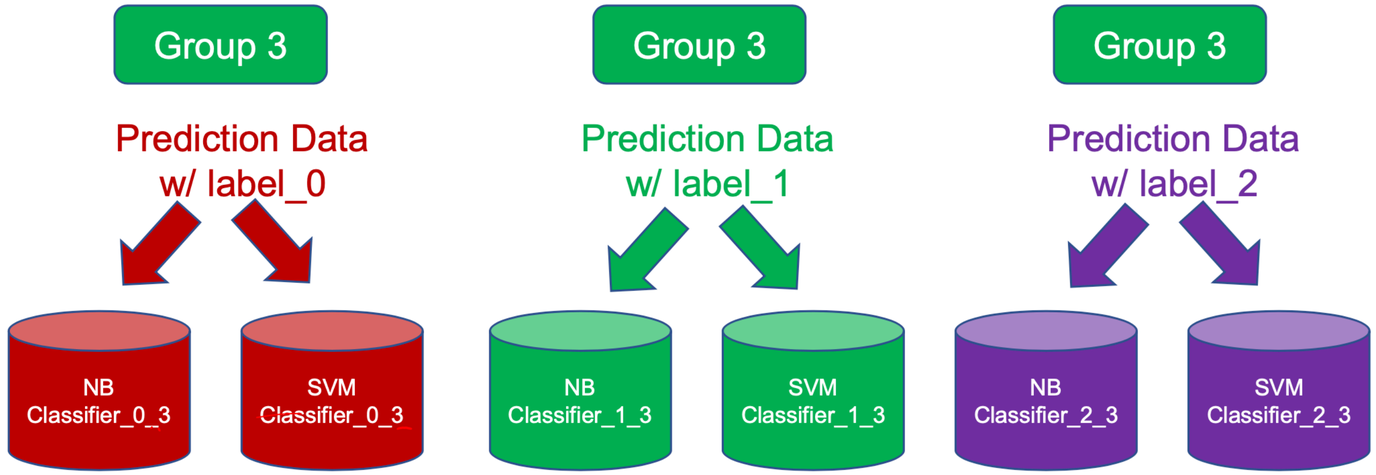
The above procedure will be repeated k times

Step 2: learn base classifiers

features	label_0	Label_1	Label_2
(5421,[1,18,31,39...	1	0	0
(5421,[0,1,15,20,...	0	1	0
(5421,[3,109,556,...	0	0	1
…	…	…	…

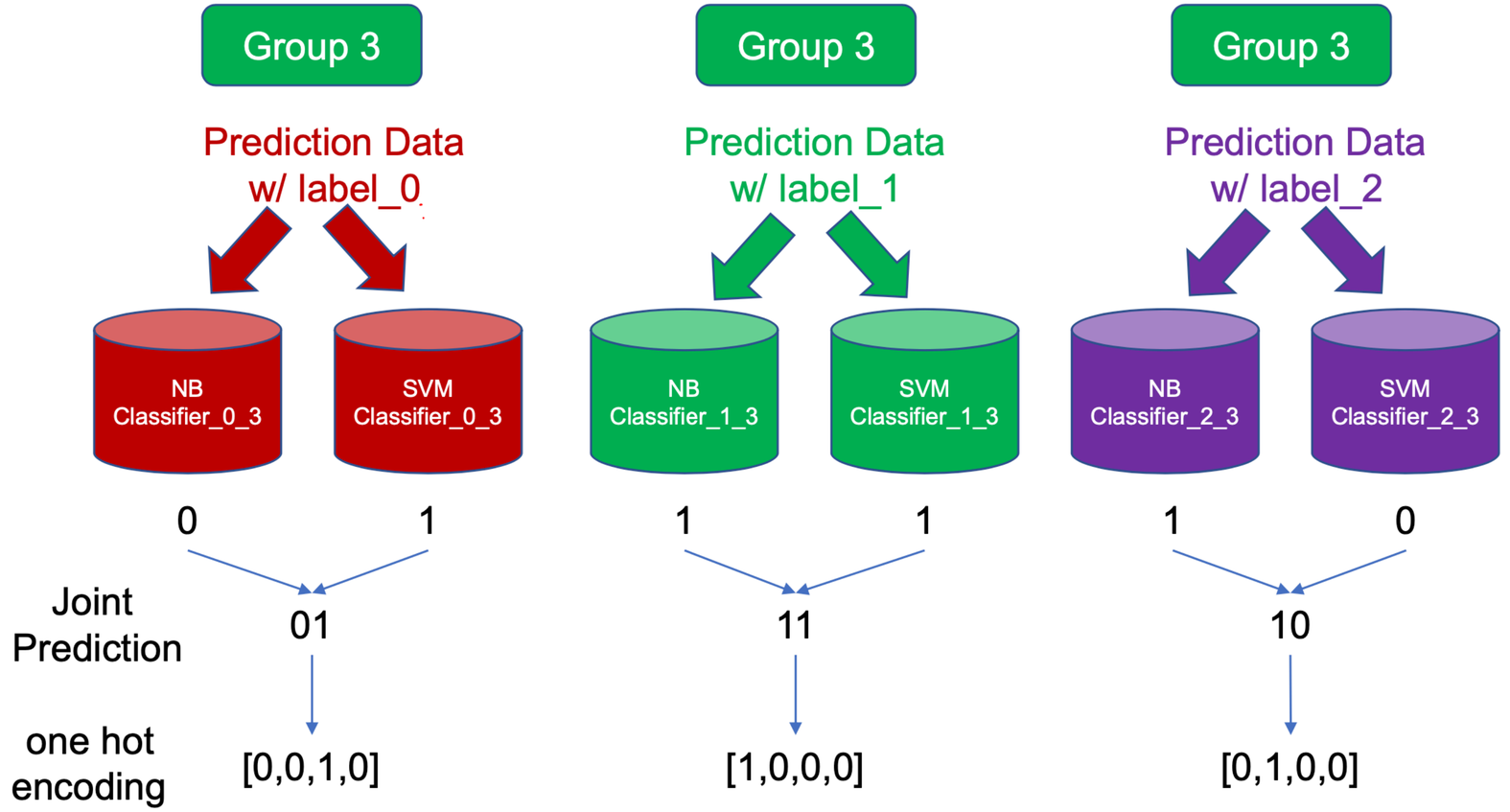


Step 3: generate features for meta model



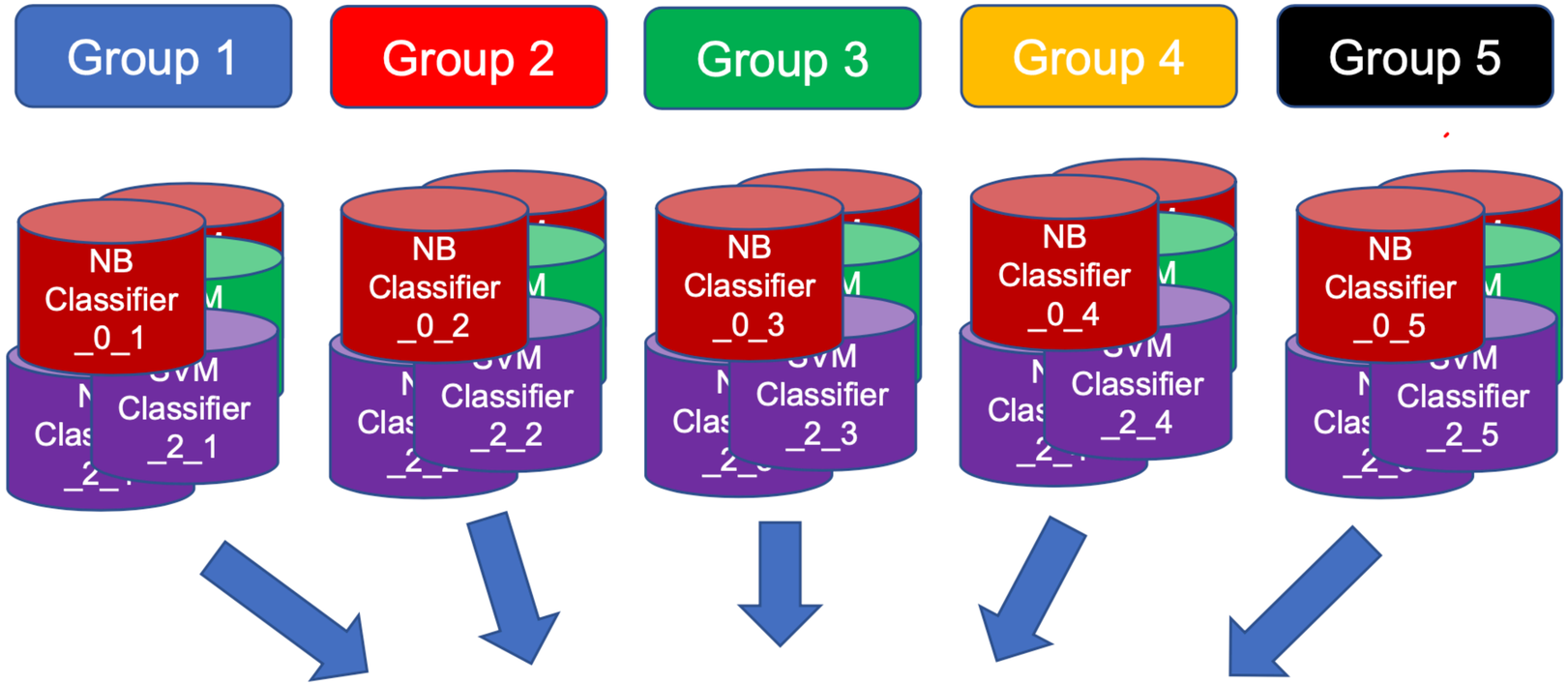
Single Pred.	0	1	1	1	1	0
one hot encoding	[0,1]	[1,0]	[1,0]	[1,0]	[1,0]	[0,1]

Step 3: generate features for meta model



Meta feature: [0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0]

Step 3: generate features for meta model



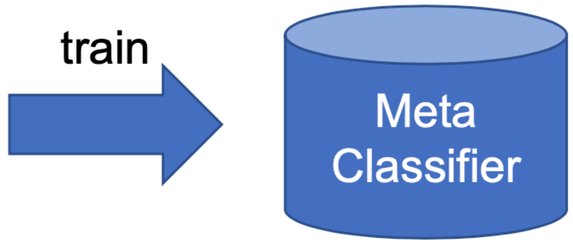
Group	meta_features	label
1	([0, 1, 1, 0, 1, 0, 1, …	0.0
4	([0, 1, 0, 0, 1, 1, 1, …	1.0
2	([0, 0, 1, 0, 1, 0, 1, …	2.0
…	…	…

Step 4: Learn Meta Classifier

- Use meta features as features, learn meta classifier on the whole dataset
- In project 2 we use logistic regression as meta model

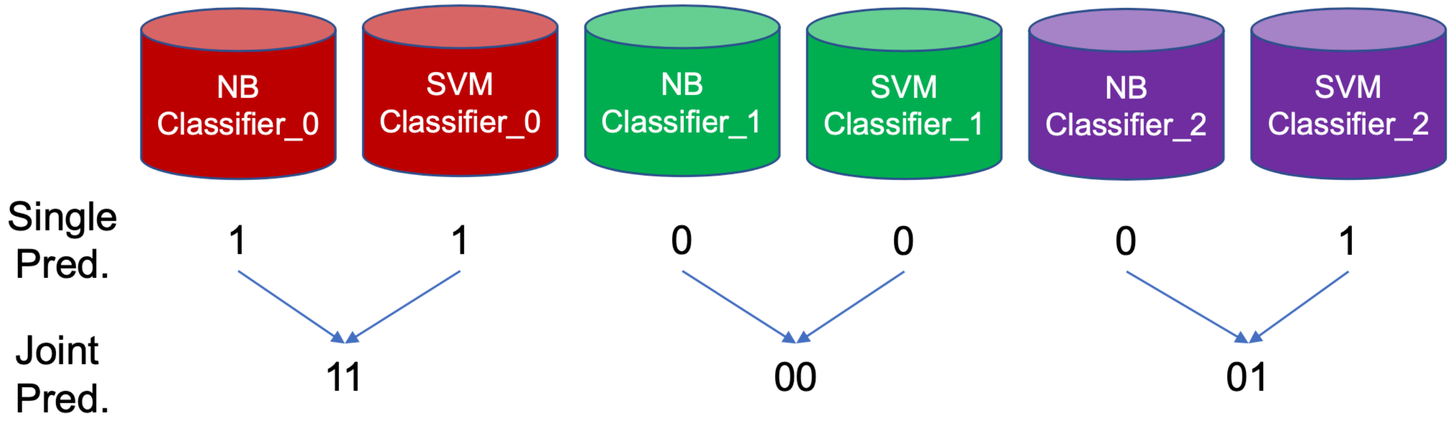
multi

Group	meta_features	label
1	([0, 1, 1, 0, 1, 0, 1, …	0.0
4	([0, 1, 0, 0, 1, 1, 1, …	1.0
2	([0, 0, 1, 0, 1, 0, 1, …	2.0
…	…	…



Step 5: generate meta features for prediction

features
(5421,[1,18,21,39...
(5421,[0,1,5,13,...
(5421,[3,10,56,...
…



Meta feature: [1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0]

Step 6: prediction using meta classifier

- Use the meta classifier trained in step 4 to predict labels for the test data with meta features generated in step 5.