COMP9313: Big Data Management

Classification and PySpark MLlib

PySpark MLlib

- MLlib is Spark’s scalable machine learning library consisting of common learning algorithms and utilities
  - Basic Statistics
  - **Classification**
  - Regression
  - Clustering
  - Recommendation System
  - Dimensionality Reduction
  - Feature Extraction
  - Optimization
- It is more or less a spark version of sk-learn

Classification

- Classification
  - predicts categorical class labels
  - constructs a model based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data
- Prediction (aka. Regression)
  - models continuous-valued functions, i.e., predicts unknown or missing values
- Applications
  - medical diagnosis
  - credit approval
  - natural language processing

Classification and Regression

- Given a new object o, map it to a feature vector $\mathbf{x} = (x_1, x_2, \ldots, x_d)^\top$
- Predict the output (class label) $y \in \mathcal{Y}$
  - Binary classification
    - $\mathcal{Y} = \{0, 1\}$ (sometimes $\{-1, 1\}$)
  - Multi-class classification
    - $\mathcal{Y} = \{1,2, \ldots, C\}$
- Learn a classification function
  - $f(\mathbf{x})\colon\ \mathbb{R}^d \mapsto \mathcal{Y}$
- Regression: $f(\mathbf{x})\colon\ \mathbb{R}^d \mapsto \mathbb{R}$

Example of Classification – Text Categorization

- Given: document or sentence
  - E.g., A statement released by Scott Morrison said he has received advice … advising the upcoming sitting be cancelled.
- Predict: Topic
  - Pre-defined labels: Politics or not?
- How to learn the classification function?
  - $f(\mathbf{x})$: $\mathbb{R}^d \mapsto \mathcal{Y}$
  - How to convert document to $\mathbf{x} \in \mathbb{R}^d$ (e.g., feature vector)?
  - How to convert pre-defined labels to $\mathcal{Y} = \{0, 1\}$?

Example of Classification – Text Categorization

- Input object: a sequence of words
- Input features **x**
  - Bag of Words representation
  - Freq(Morrison) = 2, freq(Trump) = 0, …
  - $\mathbf{x} = [2, 1, 0, \ldots]^\top$
- Class labels: $\mathcal{Y}$
  - Politics: 1
  - Not politics: -1

Convert a Problem into Classification Problem

- Input
  - How to generate input feature vectors
- Output
  - Class labels
- Another example: image classification
  - Input: A matrix of RGB values
  - Input features: color histogram
    - E.g., pixel_count(red) = ?, pixel_count(blue) = ?
  - Output: class labels
    - Building: 1
    - Not building: -1

Supervised Learning

- How to get $f(\mathbf{x})$?
- In supervised learning, we are given a set of training examples:
  - $\mathcal{D} = \{(\mathbf{x}_i, y_i), i = 1, \dots, n\}$
- Identical independent distribution (i.i.d) assumption
  - A critical assumption for machine learning theory

Machine Learning Terminologies

- Supervised learning has input labelled data
  - #instances x #attributes matrix/table
  - #attributes = #features + 1
    - 1 (usu. the last attribute) is for the class label
- Labelled data split into 2 or 3 disjoint subsets
  - Training data (used to build a classifier)
  - Development data (used to select a classifier)
  - Testing data (used to evaluate the classifier)
- Output of the classifier
  - Binary classification: #labels = 2
  - Multi-label classification: #labels > 2

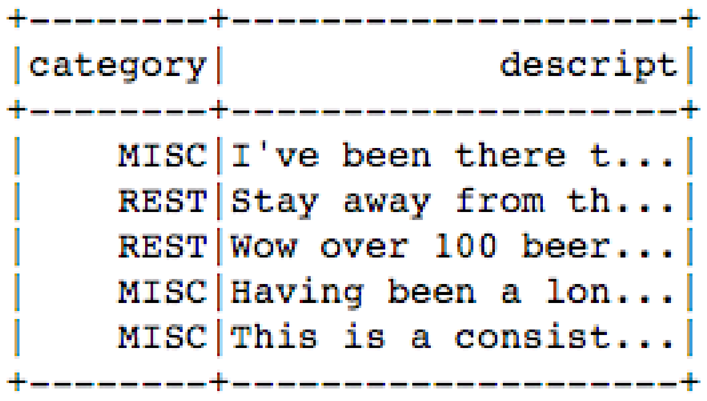
Machine Learning Terminologies

- Evaluate the classifier
  - False positive:
    - not politics but classified as politics
  - False negative
    - Politics but classified as not politics
  - True positive
    - Politics and classified as politics
- Precision $= \frac{tp}{tp+fp}$
- Recall $= \frac{tp}{tp+fn}$
- F1 score $= 2 \cdot \frac{precision \cdot recall}{precision + recall}$

Classification—A Two-Step Process

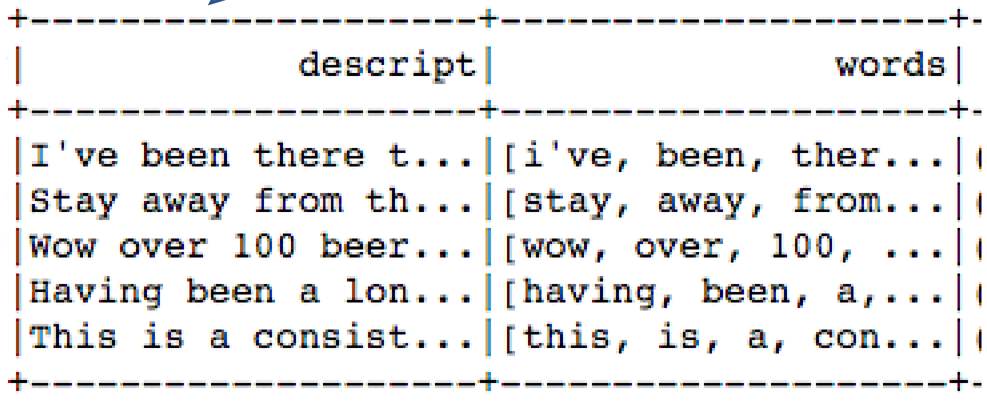
- Classifier construction
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for classifier construction is training set
  - The classifier is represented as classification rules, decision trees, or mathematical formulae
- Classifier usage: classifying future or unknown objects
  - Estimate accuracy of the classifier
    - The known label of test sample is compared with the classified result from the classifier
    - Accuracy rate is the percentage of test set samples that are correctly classified by the classifier
    - Test set is independent of training set, otherwise over-fitting will occur
  - If the accuracy is acceptable, use the classifier to classify data tuples whose class labels are not known

Classification Process 1: Preprocessing and Feature Engineering

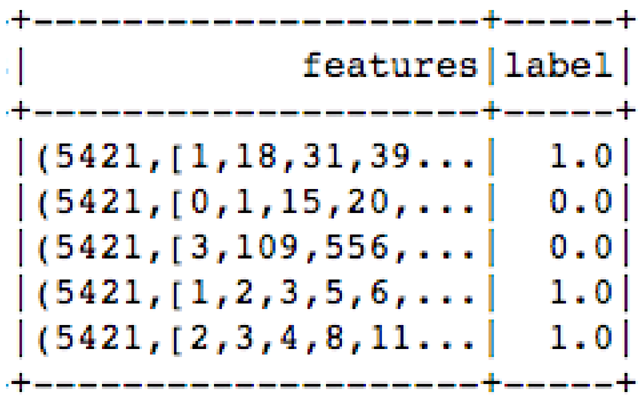


```
+--------+--------------------+
|category|            descript|
+--------+--------------------+
|    MISC|I've been there t...|
|    REST|Stay away from th...|
|    REST|Wow over 100 beer...|
|    MISC|Having been a lon...|
|    MISC|This is a consist...|
+--------+--------------------+
```

Raw Data



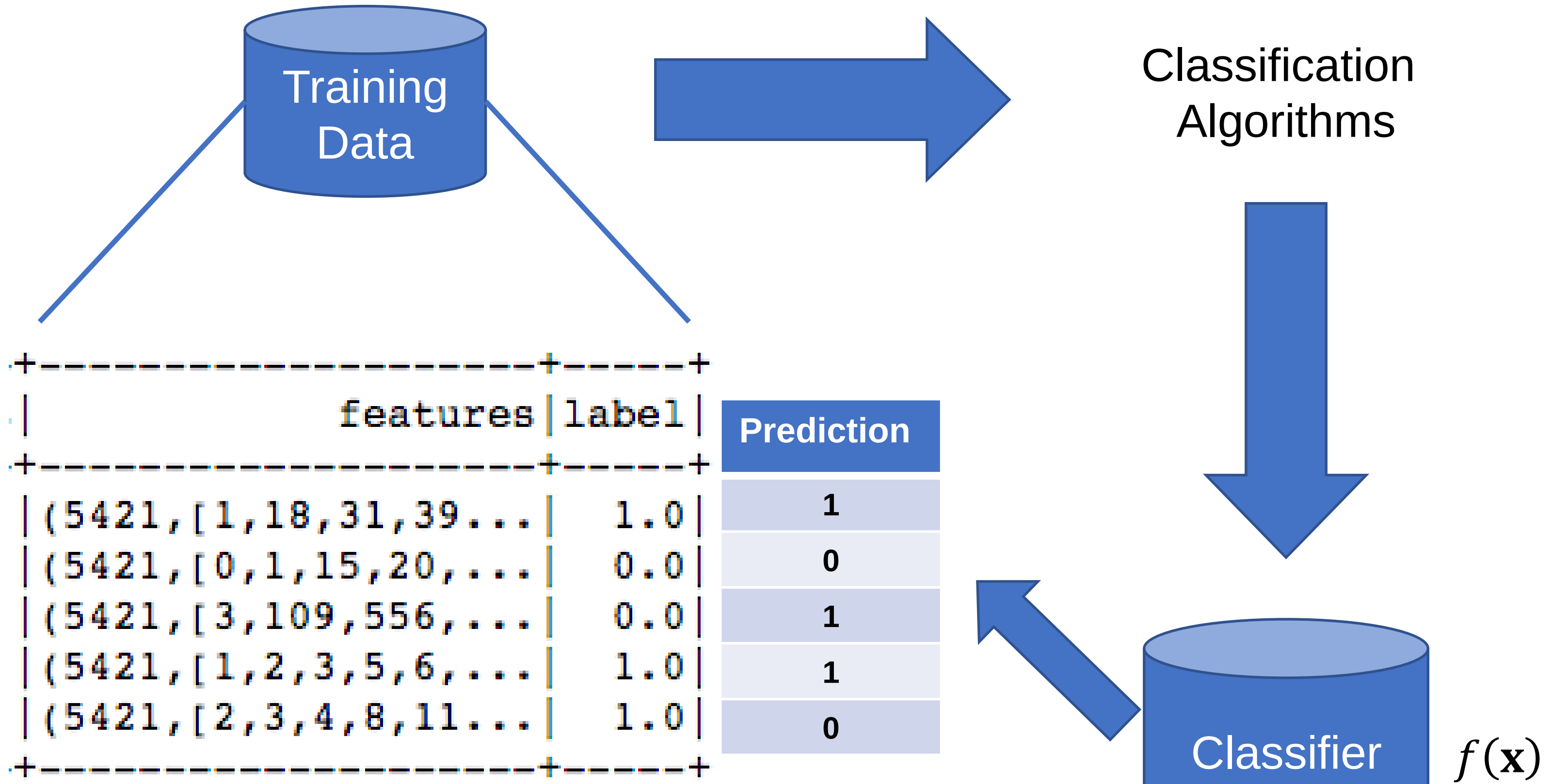
```
+--------------------+--------------------+-
|            descript|               words|
+--------------------+--------------------+-
|I've been there t...|[i've, been, ther...|
|Stay away from th...|[stay, away, from...|
|Wow over 100 beer...|[wow, over, 100, ...|
|Having been a lon...|[having,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



```
+--------------------+-----+
|            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

Classification Process 2: Train a classifier

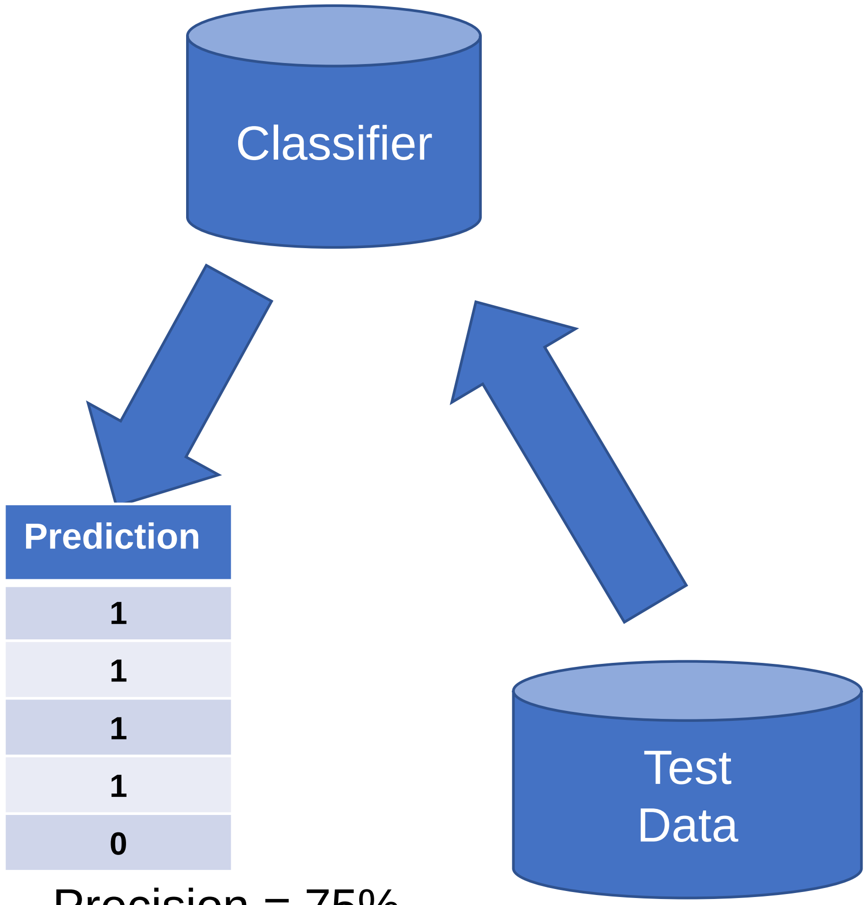


```
+--------------------+-----+
|            features|label|
+--------------------+-----+
|(5421,[1,18,31,39...|  1.0|
|(5421,[0,1,15,20,...|  0.0|
|(5421,[3,109,556,...|  0.0|
|(5421,[1,2,3,5,6,...|  1.0|
|(5421,[2,3,4,8,11...|  1.0|
+--------------------+-----+
```

Prediction
1
0
1
1
0

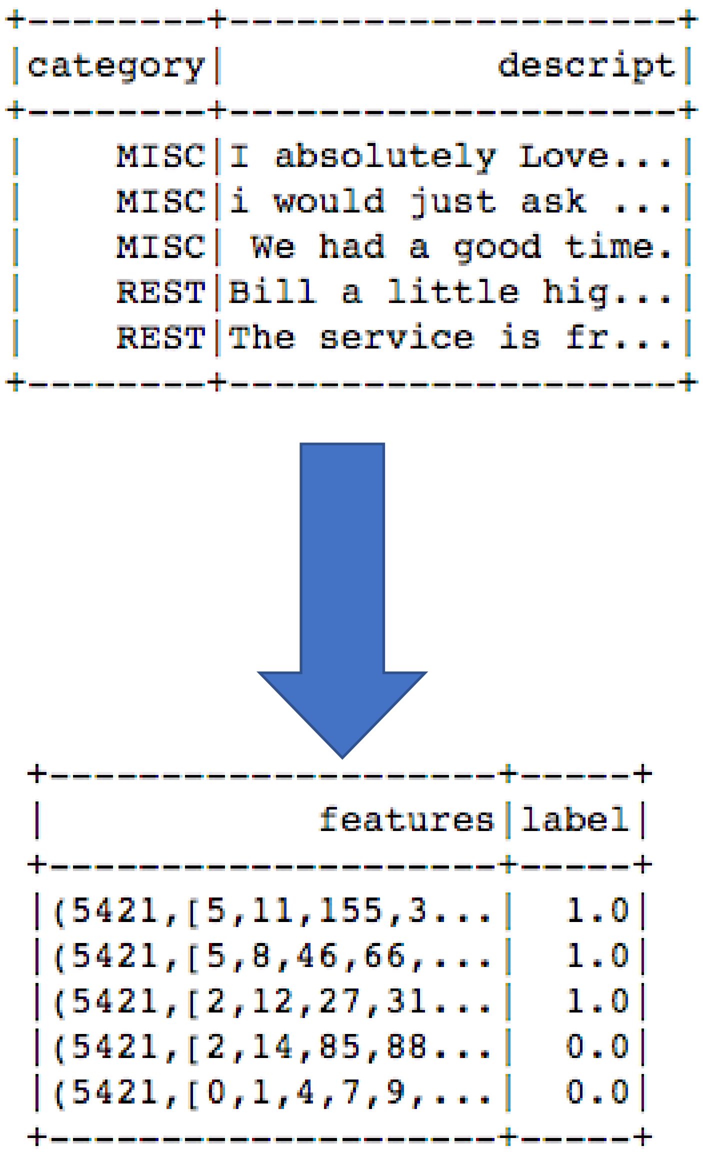
Precision = 0.66
Recall = 0.66
F1 = 0.66

Classification Process 3: Evaluate the Classifier



Prediction
1
1
1
1
0

Precision = 75%
Recall = 100%
F1 = 0.86



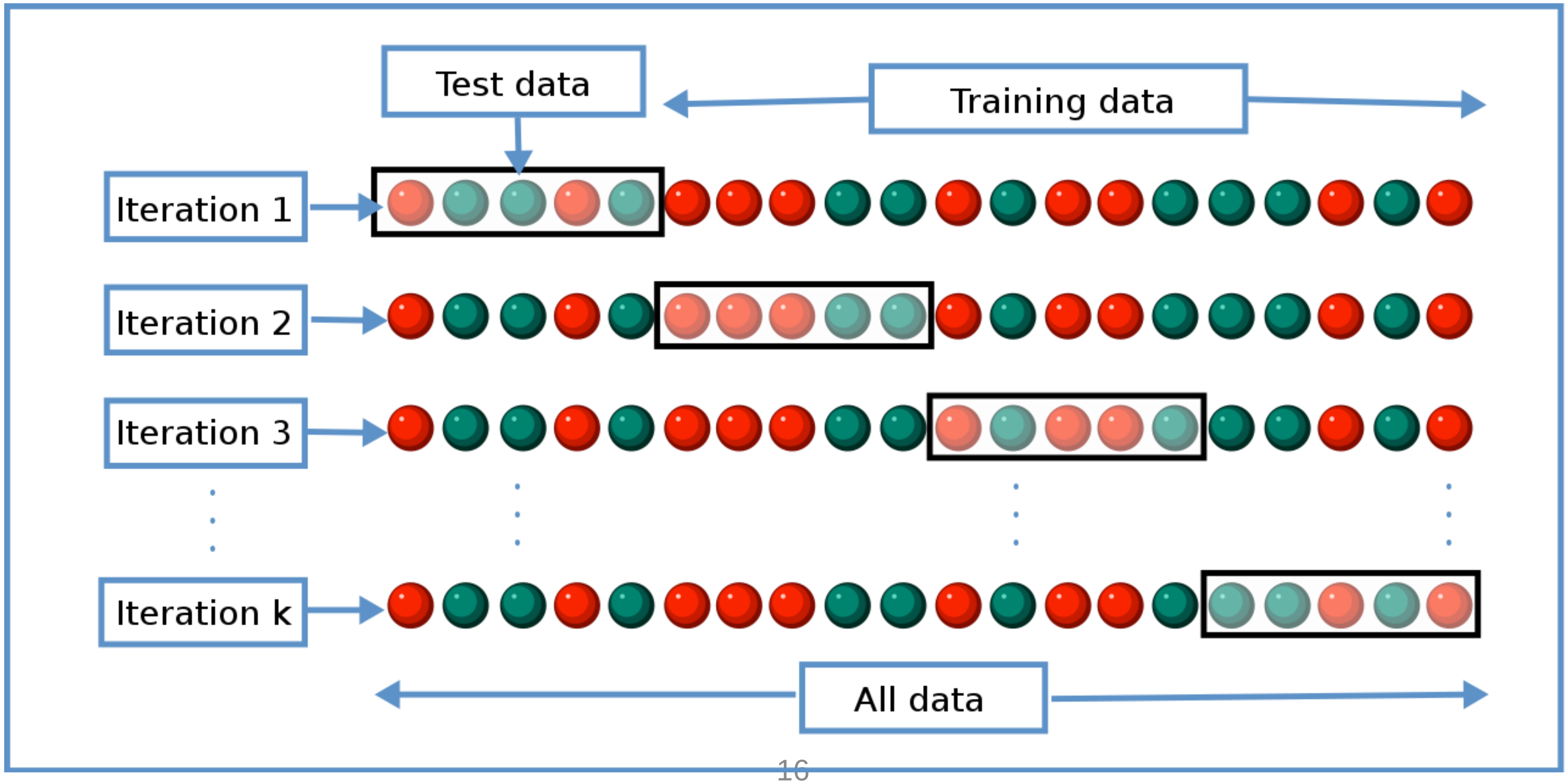
```
+--------+--------------------+
|category|             descript|
+--------+--------------------+
|    MISC|I absolutely Love...|
|    MISC|i would just ask ...|
|    MISC| We had a good time.|
|    REST|Bill a little hig...|
|    REST|The service is fr...|
+--------+--------------------+
```

```
+--------------------+-----+
|            features|label|
+--------------------+-----+
|(5421,[5,11,155,3...|  1.0|
|(5421,[5,8,46,66,...|  1.0|
|(5421,[2,12,27,31...|  1.0|
|(5421,[2,14,85,88...|  0.0|
|(5421,[0,1,4,7,9,...|  0.0|
+--------------------+-----+
```

How to Judge a Model?

- Based on training error or testing error?
  - Testing error
    - Otherwise, this is a kind of data scooping => overfitting
- What if there are multiple models to choose from?
  - Further split a “development set” from the training set
- Can we trust the error values on the development set?
  - Need “large” dev set => less data for training
  - k-fold cross-validation

k-fold cross-validation



Text Classification

- Assigning subject categories, topics, or genres
- Spam detection
- Authorship identification
- Age/gender identification
- Language Identification
- Sentiment analysis
- …

- We will do text classification in Project 2

Text Classification: Problem Definition

- Input
  - Document or sentence d

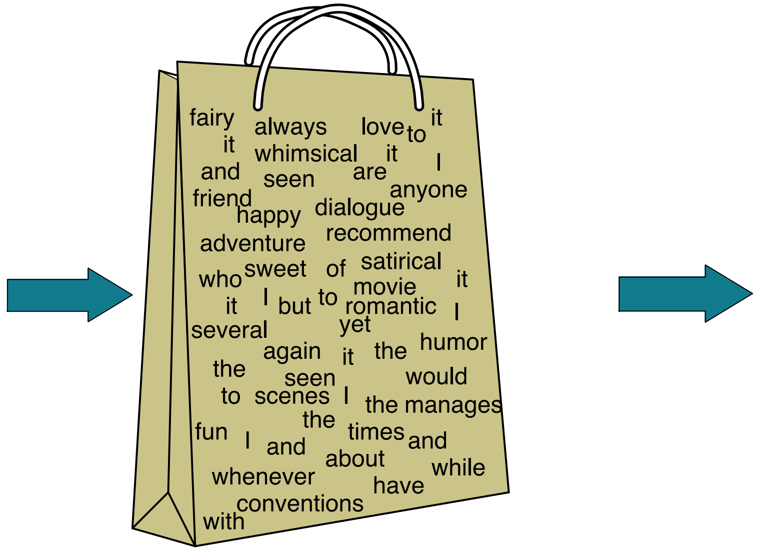
- Output
  - Class label $c \in \{\mathrm{c}_1, \mathrm{c}_2, \ldots\}$

- Classification methods:
  - Naïve bayes
  - Logistic regression
  - Support-vector machines
  - …

Naïve Bayes: Intuition

- Simple ("naïve") classification method based on Bayes rule
- Relies on very simple representation of document
  - Bag of words

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!



it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
…	…

Naïve Bayes Classifier

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

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

- We want to which class is most likely

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

Naïve Bayes Classifier

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

MAP is "maximum a posteriori" = most likely class

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

Bayes Rule

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

Dropping the denominator

$$= \underset{c \in C}{\operatorname{argmax}}\, P(x_1, x_2, \ldots, x_n \mid c)P(c)$$

Document d represented as features x1..xn

O(|X|^n•|C|) parameters. Could only be estimated if a very, very large number of training examples was available.

Multinomial Naïve Bayes Independence Assumptions

$$P(x_1, x_2, \dots, x_n \mid c)$$

- **Bag of Words assumption**: Assume position doesn't matter
- **Conditional Independence**: Assume the feature probabilities $P(x_i \mid c_j)$ are independent given the class *c.*

$$P(x_1, \dots, x_n \mid c) = P(x_1 \mid c) \cdot P(x_2 \mid c) \cdot P(x_3 \mid c) \cdot \ldots \cdot P(x_n \mid c)$$

Multinomial Naïve Bayes Classifier

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c) P(c)$$

$$\Downarrow$$

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c_j) \prod_{x \in X} P(x \mid c)$$

positions $\leftarrow$ all word positions in test document

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

Learning the Multinomial Naïve Bayes Model

- First attempt: maximum likelihood estimates
  - simply use the frequencies in the data

$$\hat{P}(c_j) = \frac{doccount(C = c_j)}{N_{doc}}$$

$$\hat{P}(w_i \mid c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}$$

fraction of times word w_i appears among all words in documents of topic c_j

- Create mega-document for topic *j* by concatenating all docs in this topic
  - Use frequency of *w* in mega-document

Problem with Maximum Likelihood

- What if we have seen no training documents with the word *fantastic* and classified in the topic *positive*?

$$\hat{P}(\text{"fantastic"} \mid \text{positive}) = \frac{count(\text{"fantastic", positive})}{\sum_{w \in V} count(w, \text{positive})} = 0$$

- Zero probabilities cannot be conditioned away, no matter the other evidence!

$$c_{MAP} = \text{argmax}_c \, \hat{P}(c) \prod_i \hat{P}(x_i \mid c)$$

Laplace (add-1) smoothing for Naïve Bayes

- Reserve a small amount of probability for unseen probabilities n (conditional)
- probabilities of observed events have to be adjusted to make the total probability equals 1.0

$$\hat{P}(w_i \mid c) = \frac{count(w_i, c)}{\sum_{w \in V} \left(count(w, c)\right)}$$

$$= \frac{count(w_i, c) + 1}{\left(\sum_{w \in V} count(w, c)\right) + |V|}$$

Multinomial Naïve Bayes: Learning

- From training corpus, extract *Vocabulary*

- Calculate $P(c_j)$ terms
  - For each c_j in C do

    $docs_j \leftarrow$ all docs with class $= c_j$

$$P(c_j) \neg \frac{|docs_j|}{|\text{total \# documents}|}$$

- Calculate $P(w_k \mid c_j)$ terms
  - $Text_j \leftarrow$ single doc containing all $docs_j$
  - For each word w_k in *Vocabulary*

    $n_k \leftarrow$ # of occurrences of w_k in $Text_j$

$$P(w_k \mid c_j) \neg \frac{n_k + a}{n + a\,|Vocabulary|}$$

- Since log(xy) = log(x) + log(y), it is better to perform all computations by summing logs of probabilities rather than multiplying probabilities.

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	?

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

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

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

$$\mathrm{P}(Chinese|c) = \frac{5+1}{8+6} = \frac{3}{7} \qquad \mathrm{P}(Chinese|o) = \frac{1+1}{3+6} = \frac{2}{9}$$

$$\mathrm{P}(Australia|c) = \frac{0+1}{8+6} = \frac{1}{14} \qquad \mathrm{P}(Australia|o) = \frac{1+1}{3+6} = \frac{2}{9}$$

$$\mathrm{P}(Sydney|\mathrm{c}) = \frac{0+1}{8+6} = \frac{1}{14} \qquad \mathrm{P}(Sydney|o) = \frac{1+1}{3+6} = \frac{2}{9}$$

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

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

Summary: Naïve Bayes

- Very Fast, low storage requirements
- Robust to irrelevant features
  - Irrelevant features cancel each other without affecting results
- Very good in domains with many equally important features
- Optimal if the independence assumption hold
  - If assumed independence is correct, then it is the Bayes optimal classifier for problem
- A good dependable baseline for text classification

PySpark MLlib – Example of NB

- Create a SparkSession and read data

```
conf = SparkConf().setMaster("local[*]").setAppName("lab3")
spark = SparkSession.builder.config(conf=conf).getOrCreate()

train_data = spark.read.load("lab3train.csv", format="csv",
sep="\t", inferSchema="true", header="true")

dev_data = spark.read.load("lab3dev.csv", format="csv",
sep="\t", inferSchema="true", header="true")

train_data.show(5)
//+--------+--------------------+
//|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...|
//+--------+--------------------+
```

```
category	descript
MISC	I've been there three times and have always had wonderful experiences.
REST	Stay away from the two specialty rolls on the menu, though- too much avocado and rice will fill you up right quick.
REST	Wow over 100 beers to choose from.
MISC	Having been a long time Ess-a-Bagel fan, I was surpised to find myself return time and time again to Murray's.
…
```

PySpark MLlib – Example of NB

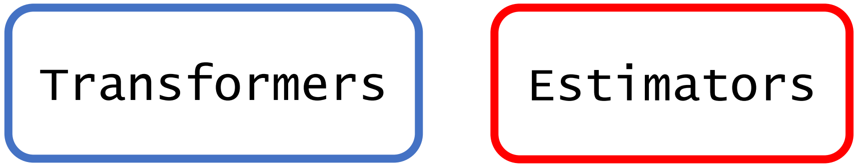
- build the pipeline

```
# 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])
```

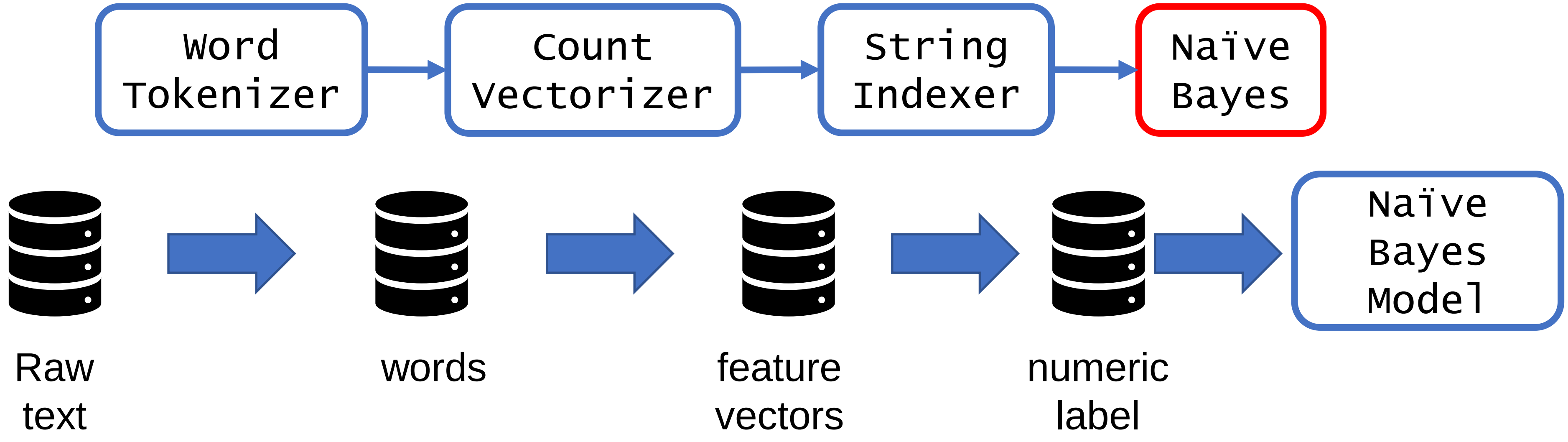
Pipeline

- In machine learning, it is common to run a sequence of algorithms to process and learn from data
- A Pipeline is specified as a sequence of stages
  - each stage is either a Transformer or an Estimator
  - stages are run in order
  - the input DataFrame is transformed as it passes through each stage
- Transformer stages
  - the transform() method is called on the DataFrame
- Estimator stages
  - the fit() method is called to produce a Transformer (which becomes part of the PipelineModel, or fitted Pipeline)
  - then Transformer’s transform() method is called on the DataFrame

Pipeline



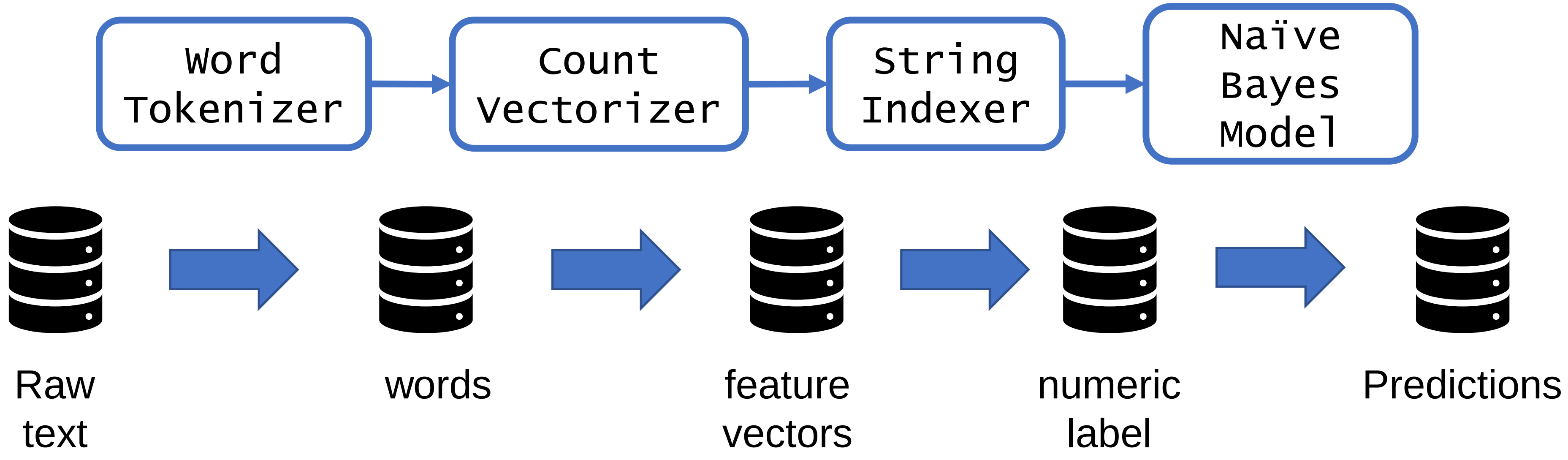
Pipeline (Estimator)



Pipeline.fit()

Pipeline

PipelineModel (Transformer)



PipelineModel.transform()

More on Pipeline

- A Transformer takes a dataframe as input and produces an augmented dataframe as output
    - `Tokenizer`
    - `CountVectorizer`
- 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

PySpark MLlib – Example of NB

- Train and evaluate the classifier

```
# train the classifier
model = nb_pipeline.fit(train_data)

# get prediction on development
dev_res = model.transform(dev_data)

# init the evaluator
evaluator =
MulticlassClassificationEvaluator(predictionCol="nb_prediction",
                                  metricName='f1')

# evaluate the result
print('F1 of NB classifier: ', evaluator.evaluate(dev_res))

// F1 of NB classifier : 0.82472850
```

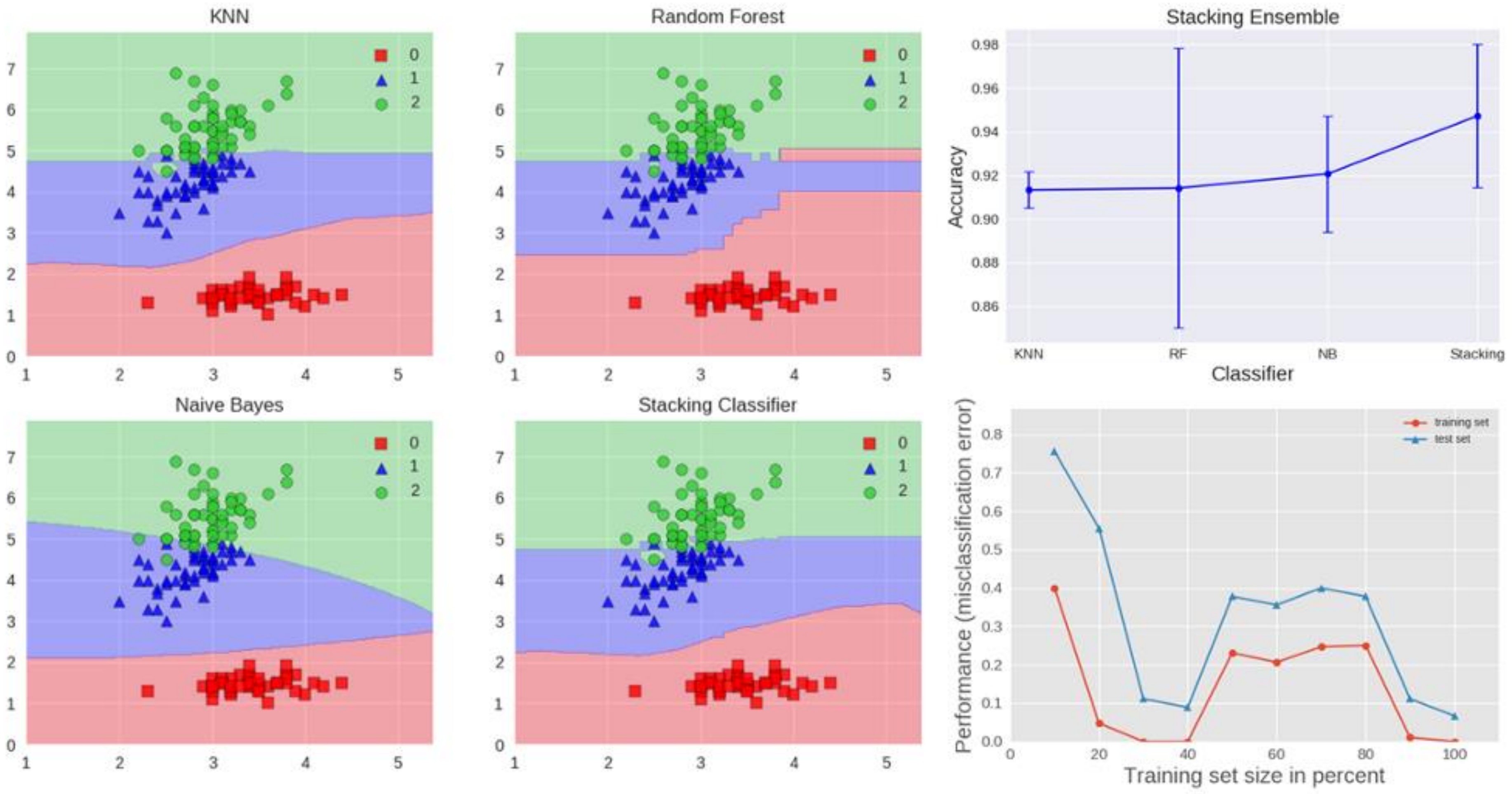
Ensemble Learning

- Ensemble learning improves machine learning results by combining several models
    - ensemble methods placed first in many machine learning competitions
- Three major types
    - Decrease variance (bagging)
    - Decrease bias (boosting)
    - Improve predictions (stacking)
- two groups based on how the base learners are genrated
    - Sequential: e.g., Adaboost
    - Parallel: e.g., Random Forest

Stacking

- Stacking is an ensemble learning technique that
  - combines multiple classification or regression models (base models)
  - via a meta-classifier or a meta-regressor (meta-model)
- Base models
  - trained based on a complete training set
  - often consists of different learning algorithms
- Meta model
  - trained on the outputs of the base level model as features
  - sometimes additional meta-features are used to further improve the performance

Stacking Example



Source

Stacking Framework

- Training
  - Step 1: prepare training data for base models
  - Step 2: learn base classifiers
  - Step 3: generate features for meta model
  - Step 4: learn meta classifier
- Testing
  - Step 5: generate features for meta classifier based on base classifiers
  - Step 6: prediction using meta classifier

Step 1: prepare training data for base models

- We need two types of base classifiers
  - Type 1: offer the meta features when training
  - Type 2: offer the meta features when testing
  - Why different?
- Type 1: k-fold cross validation
  - Split into k groups of data
    - k-1 used to train the base classifier
    - 1 used to obtain the prediction and generate the meta features
  - What if train as a whole?
    - Overfitting!
- Type 2: train as a whole

Step 1: prepare training data for base models

Text	Category
I've been there t...	MISC
Stay away from th...	PAS
I ate this a week...	FOOD
…	…

Randomly Partition into k Groups

Group	Text	Category
1	I've been there t...	MISC
4	Stay away from th...	PAS
2	I ate here a week...	FOOD
…	…	…

Generate features and labels

Group	features	label
1	(5421,[1,18,31,39...	0.0
4	(5421,[0,1,15,20,...	1.0
2	(5421,[3,109,556,...	2.0
…	…	…

Step 1: prepare training data for base models

Text	Category
I've been there t...	MISC
Stay away from th...	PAS
I ate this a week...	FOOD
…	…

Randomly Partition into k Groups

Group	Text	Category
1	I've been there t...	MISC
4	Stay away from th...	PAS
2	I ate here a week...	FOOD
…	…	…

Generate features and labels

Group	features	label
1	(5421,[1,18,31,39...	0.0
4	(5421,[0,1,15,20,...	1.0
2	(5421,[3,109,556,...	2.0
…	…	…

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,[3,10 9,556,...	0	0	1
…	…	…	…	…

Step 1: prepare training data for base models

Group	features	label
1	(5421,[1,18,31,39...	0.0
4	(5421,[0,1,15,20,...	1.0
2	(5421,[3,109,556,...	2.0
…	…	…

label = 0?

Group	features	label_0
1	(5421,[1,18,31,39...	1.0
4	(5421,[0,1,15,20,...	0.0
2	(5421,[3,109,556,...	0.0
…	…	…

label = 1?

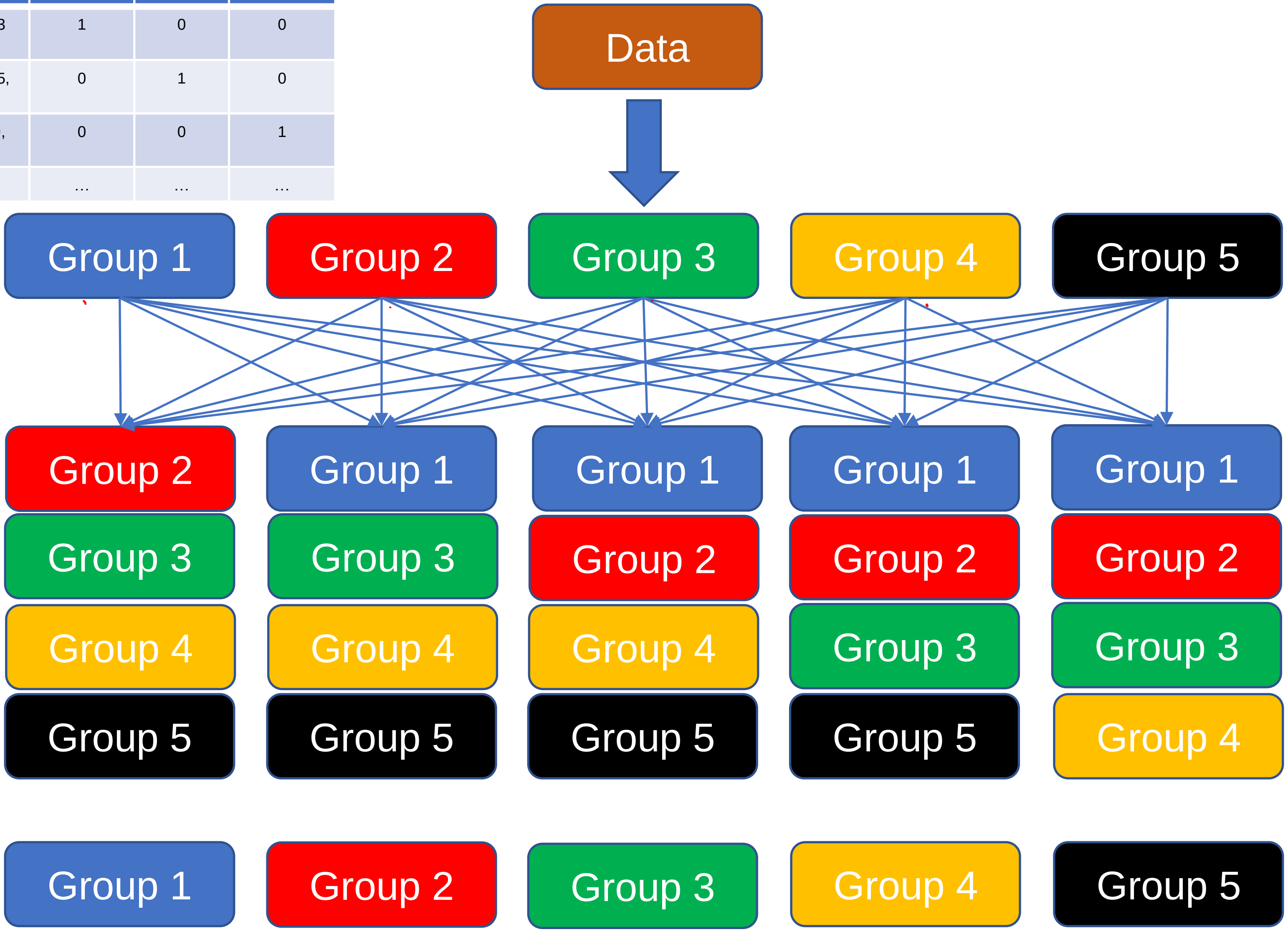
Group	features	label_1
1	(5421,[1,18,31,39...	0.0
4	(5421,[0,1,15,20,...	1.0
2	(5421,[3,109,556,...	0.0
…	…	…

label = 2?

Group	features	label_2
1	(5421,[1,18,31,39...	0.0
4	(5421,[0,1,15,20,...	0.0
2	(5421,[3,109,556,...	1.0
…	…	…

Step 1: prepare training data for base models

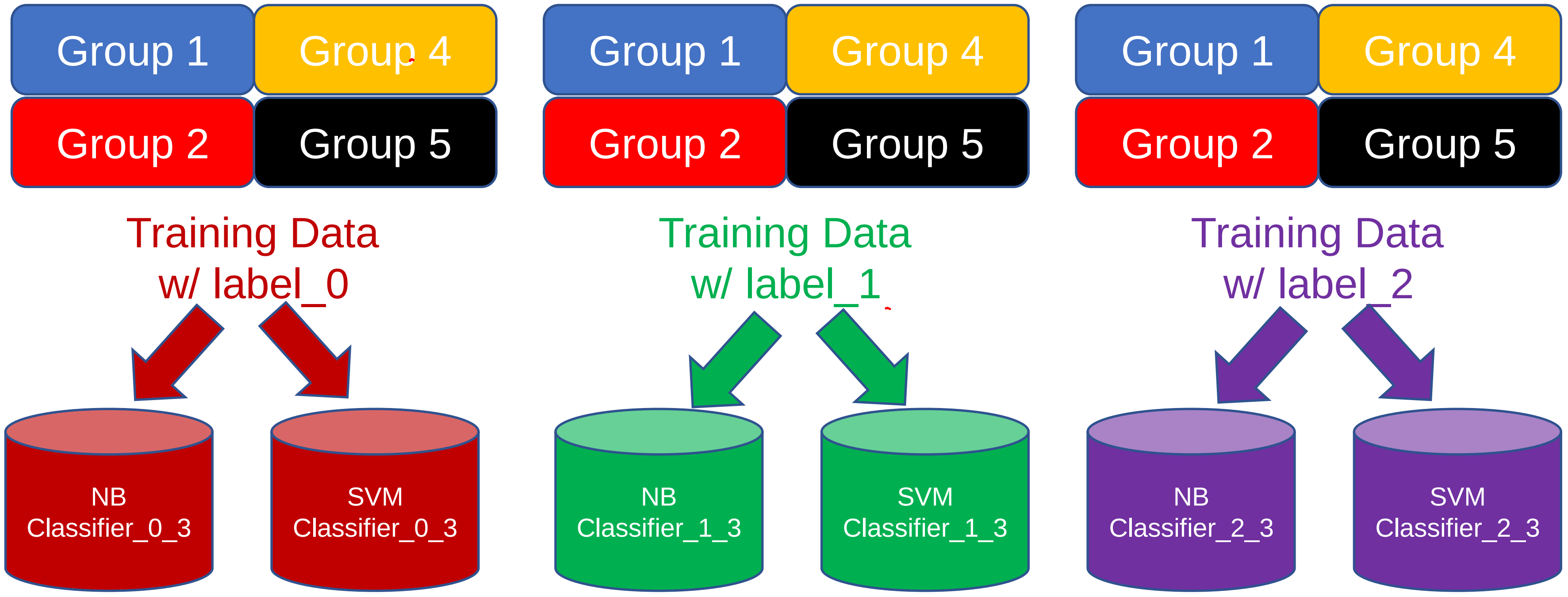
Group	features	label_0	label_1	label_2
1	(5421,[1,18,31,39...	1	0	0
4	(5421,[0,1,15,20,...	0	1	0
2	(5421,[3,109,556,...	0	0	1
…	…	…	…	…



Step 2: learn base classifiers

- Any model can be used as base model
  - Some model can only handle binary classification problem, e.g., SVM
  - Build $|C|$ one-vs-rest classifiers
    - Each classifier predicts whether the sample is class c_i or not.
- In Project 2, we use naïve bayes and SVM as base models, and $|C| = 3$
  - How many classifiers do we need to train?

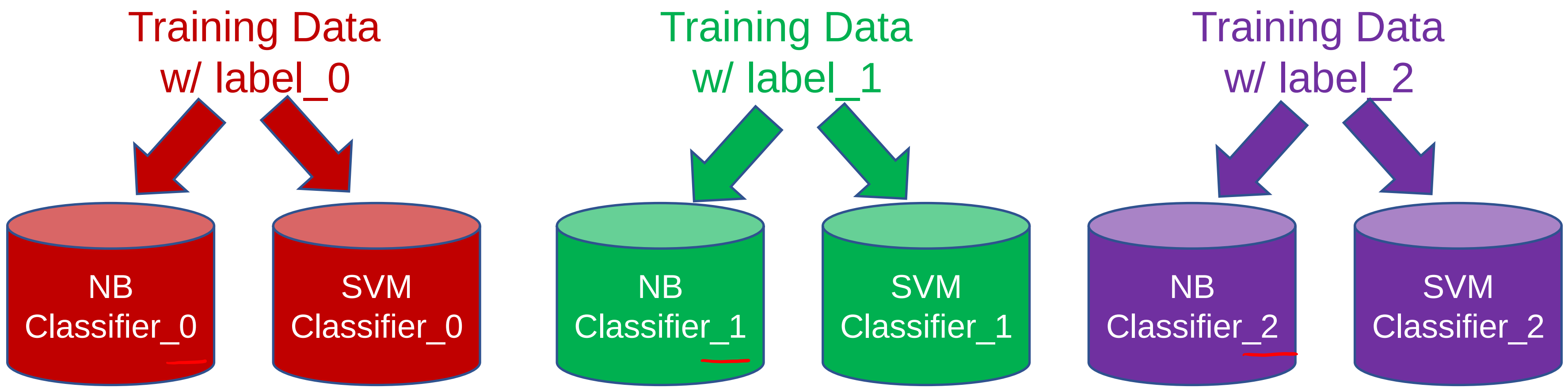
Step 2: learn base classifiers



The above procedure will be repeated k times

Step 2: learn base classifiers

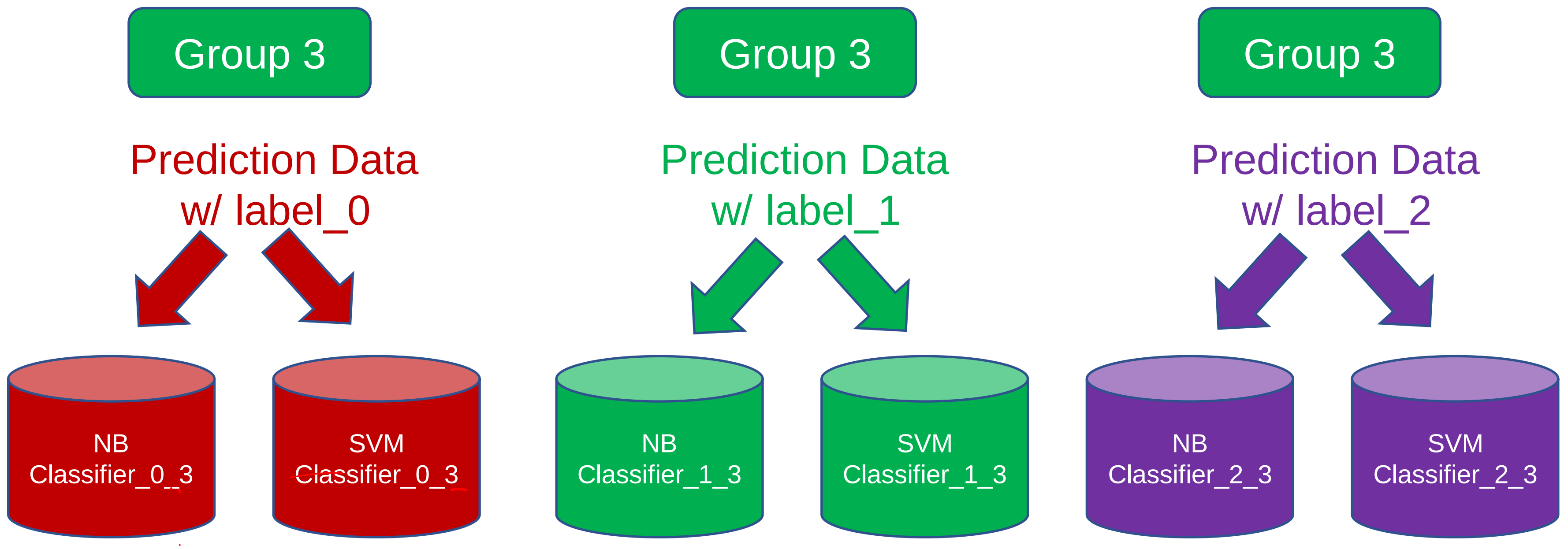
features	label_0	Label_1	Label_2
(5421,[1,18,31,39...	1	0	0
(5421,[0,1,15,20,...	0	1	0
(5421,[3,109,556,...	0	0	1
…	…	…	…



Step 3: generate features for meta model

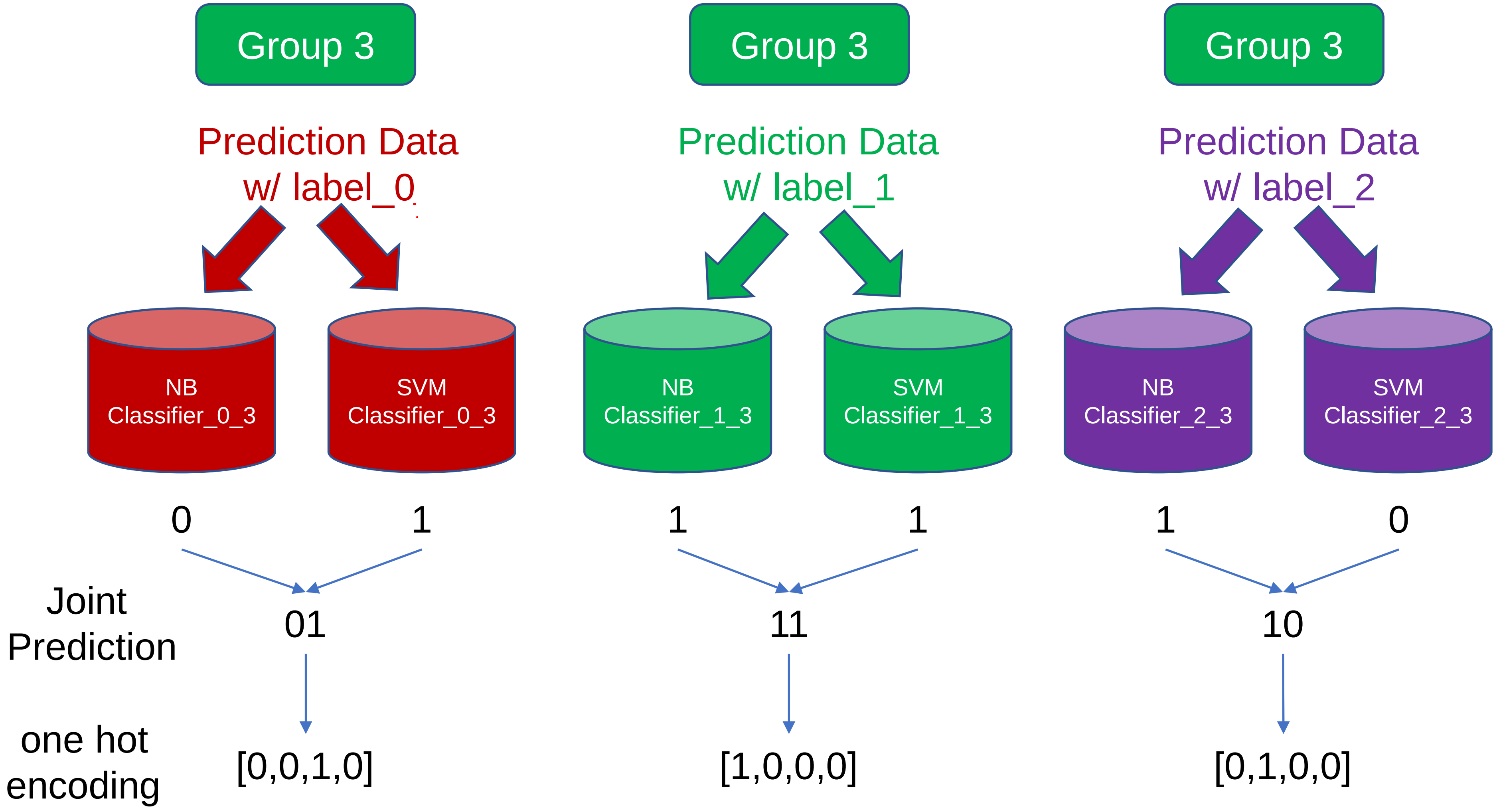
- We consider two types of meta-features
  - The prediction result from each base classifier
  - The joint prediction result from base classifiers
- Single prediction result from each base classifier
  - Generate |C|*2 features
- joint prediction result from classifiers with same label system
  - Generate |C| features
- all of them are one-hot-encoded

Step 3: generate features for meta model



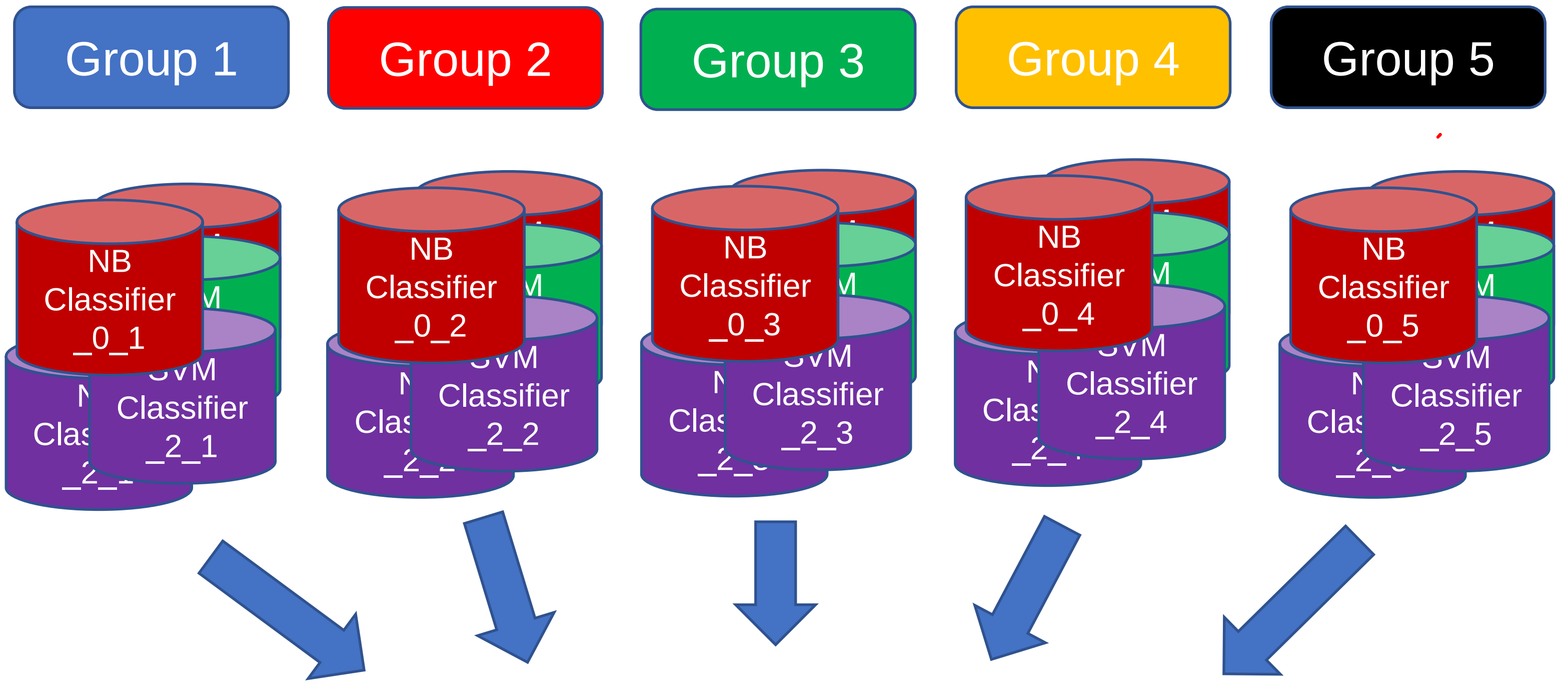
Single Pred.	0	1	1	1	1	0
one hot encoding	[0,1]	[1,0]	[1,0]	[1,0]	[1,0]	[0,1]

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

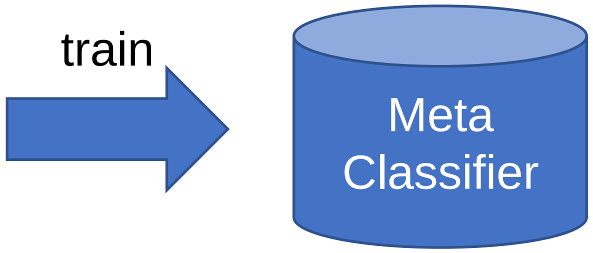


Group	meta_features	label
1	([0, 1, 1, 0, 1, 0, 1, …	0.0
4	([0, 1, 0, 0, 1, 1, 1, …	1.0
2	([0, 0, 1, 0, 1, 0, 1, …	2.0
…	…	…

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

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
…	…	…

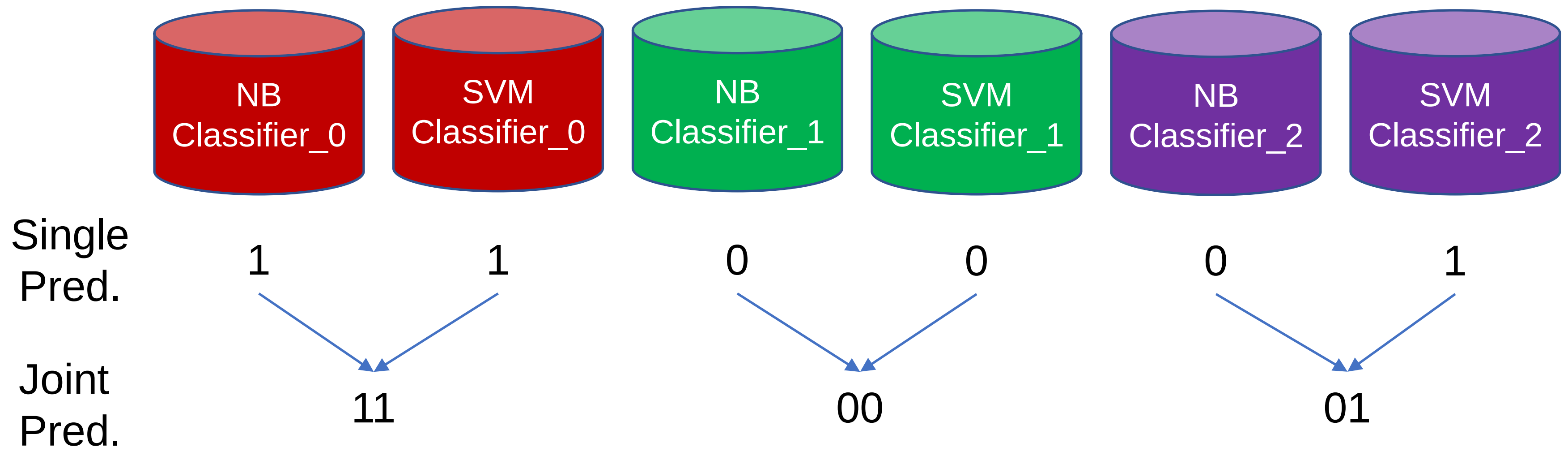


Step 5: generate meta features for prediction

- In Step 2, we have learnt base classifiers for meta feature generation in testing phase
- Before using meta classifier to predict, we need to generate meta features in a similar way as in Step 3

Step 5: generate meta features for prediction

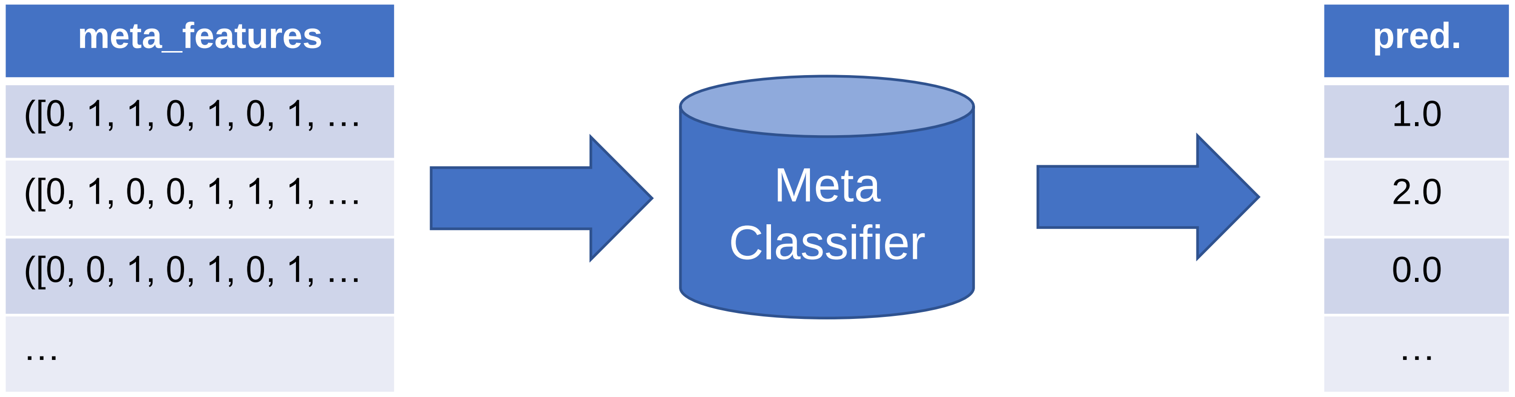
features
(5421,[1,18,21,39...
(5421,[0,1,5,13,...
(5421,[3,10,56,...
…



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.



meta_features
([0, 1, 1, 0, 1, 0, 1, …
([0, 1, 0, 0, 1, 1, 1, …
([0, 0, 1, 0, 1, 0, 1, …
…

pred.
1.0
2.0
0.0
…

Project 2

- To be released in week 8
  - We are adjusting the difficulty based on project 1
- Implement a stacking model
  - 3 labels (FOOD, PAS, MISC)
  - SVM, Naïve Bayes as base models
  - Logistic Regression as meta model
- Use Dataframe and pipeline to help you with the implementation
- No running time requirements (of course, shouldn’t be too slow)
- Deadline: the end of week 10
  - Time is enough if you don’t rush it in the last 3 days…