
COMP9313
Project 2
登登教育
-Elijah-
20T2

COMP9313 Project 2 Updated Spec and Clarifications

Hi all,

Please find attached the updated version of project 2 (COMP9313_Project2_spec_v2.ipynb). We will also update the

The new version of the spec offers a deterministic result by fixing the random seed when we generate group IDs. No
implementation is 100% correct. You still need to understand the whole procedure and make sure that you have follo

In addition, here are some clarifications that I want to make:

1. You are **NOT** required to submit your implementation of task 2.2. Your submission is for **task 1 ONLY**.
2. **Task 2.2 is an open question**. I understand that some of you have taken machine learning courses before whil
   understanding of how a stacking method works. That's why we set this task and ask you to report your findings
3. You need to import **any module** that has been used by your implementation in `submission.py`. Don't assum
4. The test environment for project 2 is the same as in lab3 (i.e., the one in lab 1 with NumPy installed)

Yifang

#pin

project2

PySpark MLlib – Example of NB

在同一个 Pipeline 里面的 Estimator 只能 fit 同一个 DataFrame

- build the pipeline

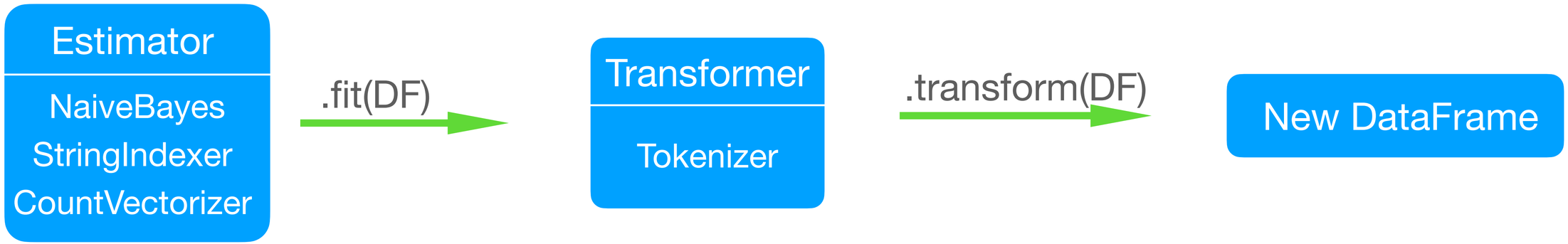
```
# white space expression tokenizer
WordTokenizer = Tokenizer(inputCol="descript", outputCol="words")

# 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])
```


Estimator
NaiveBayes
StringIndexer
CountVectorizer
.fit(DF)
Transformer
Tokenizer
.transform(DF)
New DataFrame

Task 1.1 (30 points): Build a Preprocessing Pipeline

In this task, you need to complete the `base_features_gen_pipeline()` function in `submission.py`, which outputs a pipeline (**NOTE**: not a pipeline model). The returned pipeline will be used to process the data, construct the feature vectors and labels.

More specifically, the function is defined as

```
def build_base_features_pipeline(input_descript_col="descript", input_category_col="category", output_featur
e_col="features", output_label_col="label"):
```

The function needs to tokenize each customer review (i.e., the `descript`) and generate bag of words count vectors as `features`. It also needs to convert the `category` into `label` which is an integer between 0 and 2.

The returned type of this function should be `pyspark.ml.pipeline.Pipeline`.

Data Preprocessing — Task 1.1

一步一步展开

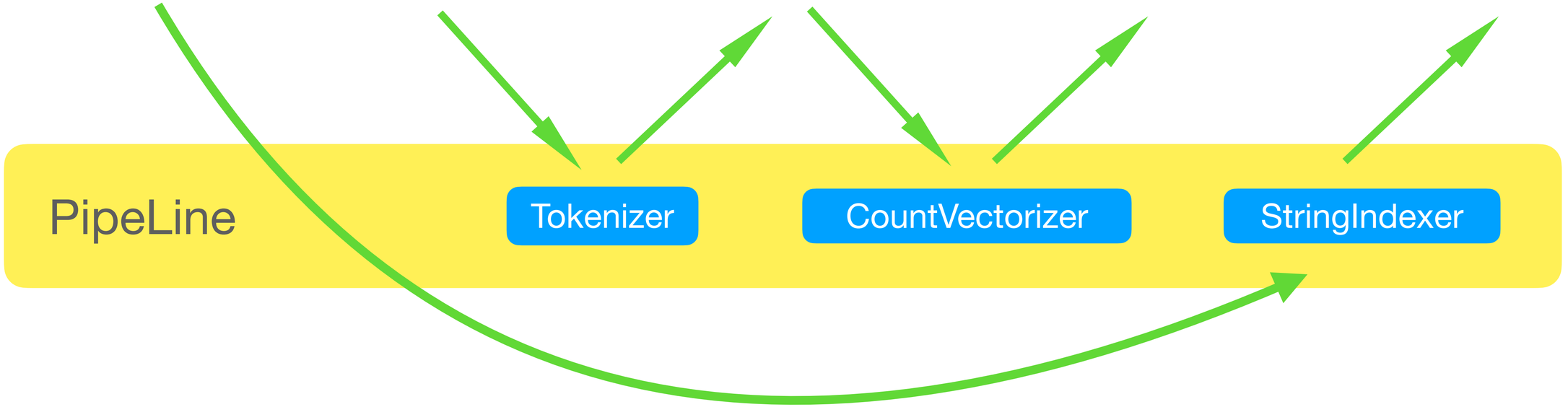
Read From CSV

id	category	descript
0	MISC	I've been there t...
1	FOOD	Stay away from th...
2	FOOD	Wow over 100 beer...
3	MISC	Having been a lon...
4	MISC	This is a consist...

words
[i've, been, ther...
[stay, away, from...
[wow, over, 100, ...
[having, been, a,...
[this, is, a, con...

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

label
1.0
0.0
0.0
1.0
1.0



Step 1. Data Preprocessing — Task 1.1

Sparse Vector

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

[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]`

```
training_set.select("features").show(5, truncate = False)
```

```
+-----------------------------------------------------------------------------------------------------------------------------------------------------------+
|features                                                                                                                                                   |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------+
|(5421,[1,18,31,39,41,58,73,110,192,263,1878],[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0])                                                               |
|(5421,[0,1,15,20,43,55,83,86,129,176,202,311,322,336,356,663,1889,2194,2211,3119,4380],[2.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0])|
|(5421,[3,109,556,721,1001,3896,4238],[1.0,1.0,1.0,1.0,1.0,1.0,1.0])                                                                                        |
|(5421,[1,2,3,5,6,39,92,178,302,357,375,836,938,1421,2652,3680,5384],[1.0,1.0,2.0,1.0,1.0,1.0,3.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0])                  |
|(5421,[2,3,4,8,11,24,28,48,151,243,460,582],[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0])                                                             |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------+
only showing top 5 rows
```

Demo

Stacking 流程

代码实操

调优方案

Stacking 流程

id	category	descript
0	MISC	I've been there t...
1	FOOD	Stay away from th...
2	FOOD	Wow over 100 beer...
3	MISC	Having been a lon...
4	MISC	This is a consist...

words
[i've, been, ther...
[stay, away, from...
[wow, over, 100, ...
[having, been, a,...
[this, is, a, con...

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

label
1.0
0.0
0.0
1.0
1.0

```
# assign random groups and binarize the labels

def gen_binary_labels(df):
    df = df.withColumn('label_0', (df['label'] == 0).cast(DoubleType()))
    df = df.withColumn('label_1', (df['label'] == 1).cast(DoubleType()))
    df = df.withColumn('label_2', (df['label'] == 2).cast(DoubleType()))
    return df
training_set = training_set.withColumn('group', (rand()*5).cast(IntegerType()))
training_set = gen_binary_labels(training_set)
```

id	category	descript	words	features	label	group	label_0	label_1	label_2
0	MISC	I've been there t...	[i've, been, ther...	(5421,[1,18,31,39...	1.0	2	0.0	1.0	0.0
1	FOOD	Stay away from th...	[stay, away, from...	(5421,[0,1,15,20,...	0.0	3	1.0	0.0	0.0
2	FOOD	Wow over 100 beer...	[wow, over, 100, ...	(5421,[3,109,556,...	0.0	4	1.0	0.0	0.0
3	MISC	Having been a lon...	[having, been, a,...	(5421,[1,2,3,5,6,...	1.0	0	0.0	1.0	0.0
4	MISC	This is a consist...	[this, is, a, con...	(5421,[2,3,4,8,11...	1.0	2	0.0	1.0	0.0

Task 1.2 (30 points): Generate Meta Features for Training

In this task, you need to complete the `gen_meta_features()` function in `submission.py`, which outputs a dataframe with generated meta featur for training the meta classifier.

More specifically, the function is defined as

```
def gen_meta_features(training_df, nb_0, nb_1, nb_2, svm_0, svm_1, svm_2):
```

The description of **input** parameters are as below:

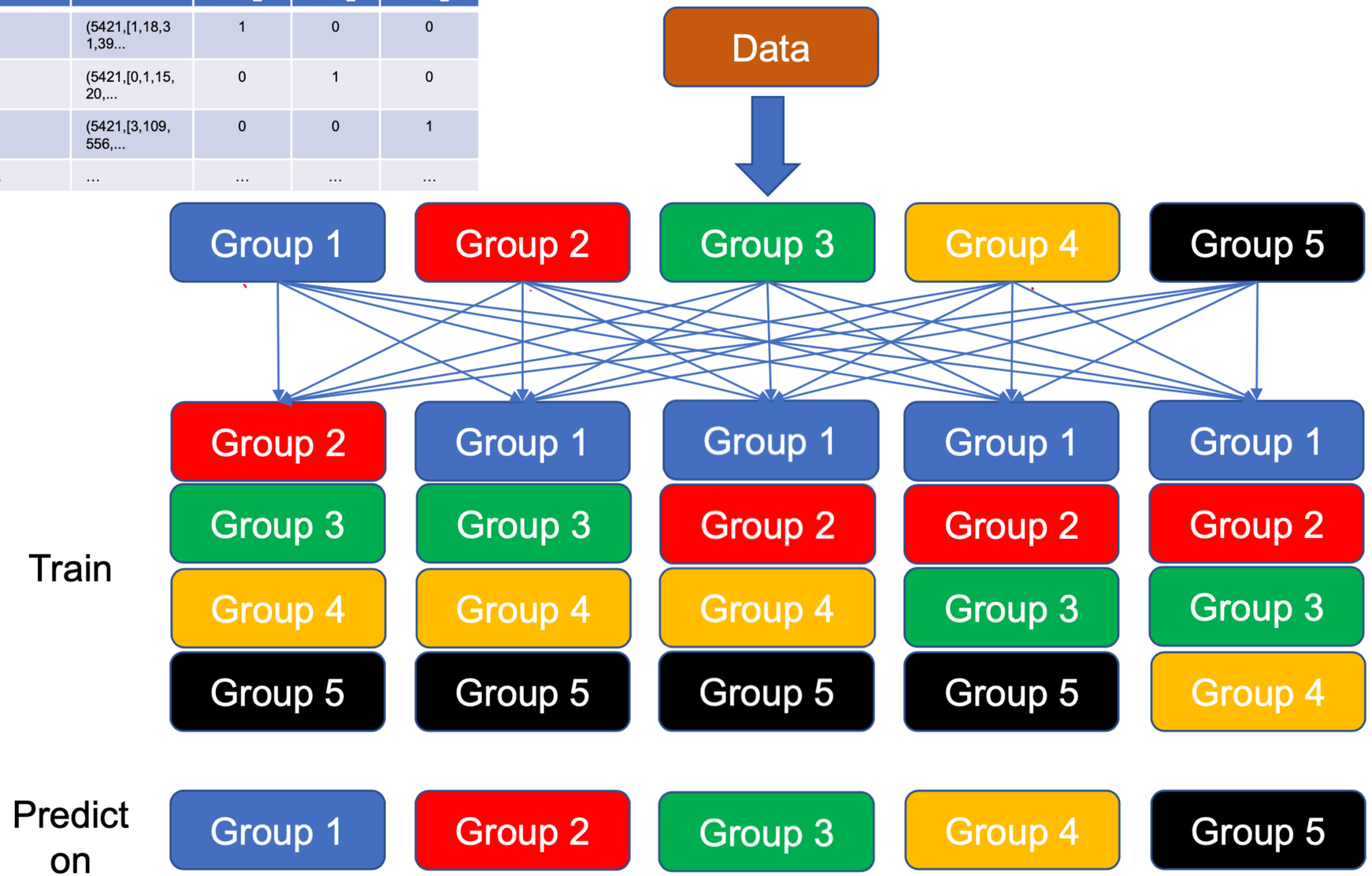
- `training_df`: the dataframe contains features, labels, and group ids for training data. The schema of `training_df` is:

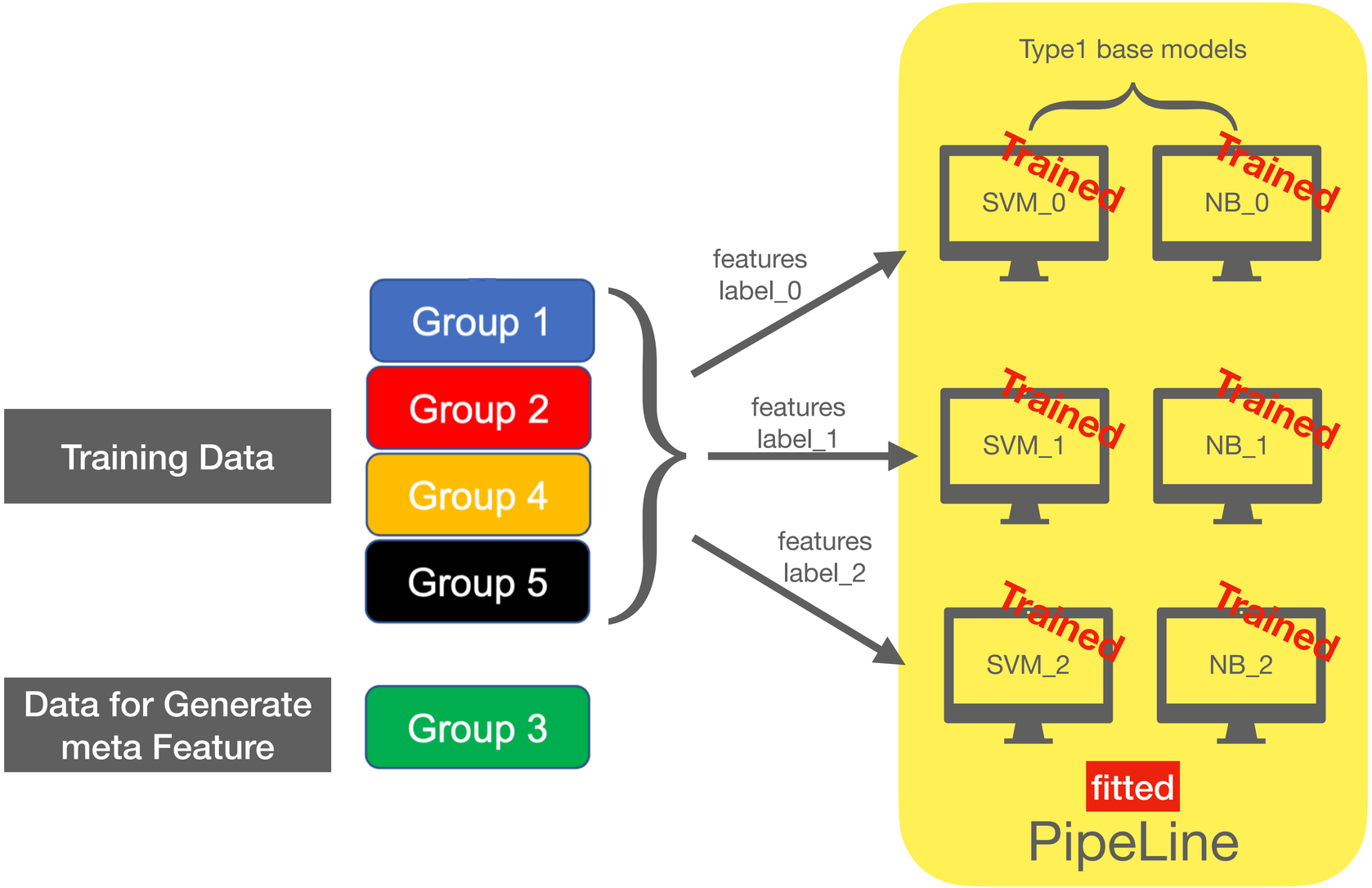
  ```
  root
  |-- id: integer (nullable = true)
  |-- features: vector (nullable = true)
  |-- label: double (nullable = false)
  |-- label_0: double (nullable = false)
  |-- label_1: double (nullable = false)
  |-- label_2: double (nullable = false)
  |-- group: integer (nullable = true)
  ```

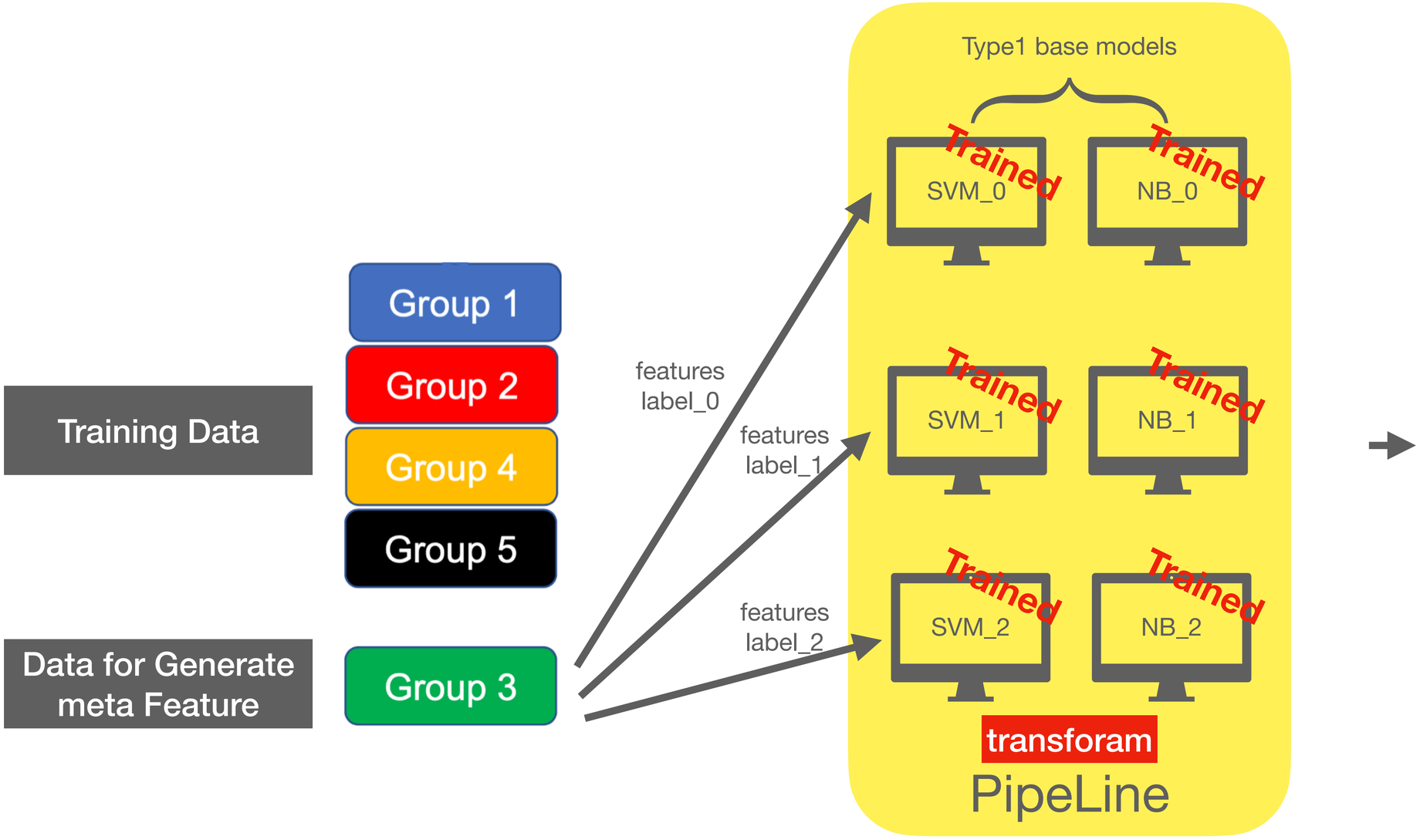
  where `features` and `label` are generated using the pipeline built in Task 1.1. `label_x` corresponds to the binary label of label x (e.g., `label_0==0` means that `label!=0`). `group` is the group id as defined in the lecture slides (i.e., L7P45).
- nb_x: the predefined x-th Naive Bayes model (i.e., the one will be trained using `label_x`)
- svm_x: the predefined x-th SVM model (i.e., the one will be trained using `label_x`)

Generate Meta Features for Training — Task 1.2

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
…	…	…	…	…




Training Data
Data for Generate meta Feature
Group 1
Group 2
Group 4
Group 5
Group 3
features label_0
features label_1
features label_2
Type1 base models
SVM_0
NB_0
SVM_1
NB_1
SVM_2
NB_2
Trained
fitted
PipeLine

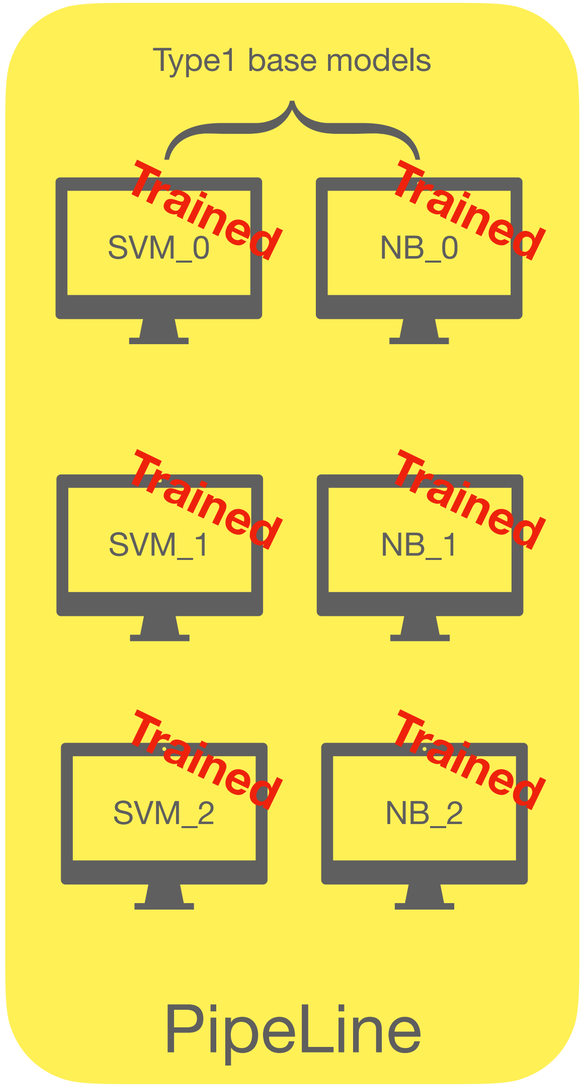


Group 3

nb_pred_0	svm_pred_0
0.0	0.0
0.0	0.0
0.0	1.0
0.0	0.0
1.0	1.0

nb_pred_1	svm_pred_1
0.0	0.0
1.0	1.0
0.0	0.0
1.0	1.0
0.0	0.0

nb_pred_2	svm_pred_2
0.0	1.0
0.0	0.0
0.0	0.0
0.0	0.0
0.0	0.0



Group 3

可以使用 udf 生成

```
+---------+----------+
|nb_pred_0|svm_pred_0|
+---------+----------+
|      0.0|       0.0|
|      0.0|       0.0|
|      0.0|       1.0|
|      0.0|       0.0|
|      1.0|       1.0|
+---------+----------+
```

```
+------------+
|joint_pred_0|
+------------+
|         0.0|
|         0.0|
|         1.0|
|         0.0|
|         3.0|
+------------+
```

```
+---------+----------+
|nb_pred_1|svm_pred_1|
+---------+----------+
|      0.0|       0.0|
|      1.0|       1.0|
|      0.0|       0.0|
|      1.0|       1.0|
|      0.0|       0.0|
+---------+----------+
```

```
+------------+
|joint_pred_1|
+------------+
|         0.0|
|         3.0|
|         0.0|
|         3.0|
|         0.0|
+------------+
```

```
+---------+----------+
|nb_pred_2|svm_pred_2|
+---------+----------+
|      0.0|       1.0|
|      0.0|       0.0|
|      0.0|       0.0|
|      0.0|       0.0|
|      0.0|       0.0|
+---------+----------+
```

```
+------------+
|joint_pred_2|
+------------+
|         1.0|
|         0.0|
|         0.0|
|         0.0|
|         0.0|
+------------+
```

Group 3

nb_pred_0	svm_pred_0	nb_pred_1	svm_pred_1	nb_pred_2	svm_pred_2	joint_pred_0	joint_pred_1	joint_pred_2
0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0
0.0	0.0	1.0	1.0	0.0	0.0	0.0	3.0	0.0
0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
0.0	0.0	1.0	1.0	0.0	0.0	0.0	3.0	0.0
1.0	1.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0

Group 1

nb_pred_0	svm_pred_0	nb_pred_1	svm_pred_1	nb_pred_2	svm_pred_2	joint_pred_0	joint_pred_1	joint_pred_2
0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0
0.0	0.0	1.0	1.0	0.0	0.0	0.0	3.0	0.0
0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
0.0	0.0	1.0	1.0	0.0	0.0	0.0	3.0	0.0
1.0	1.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0

Group 2

nb_pred_0	svm_pred_0	nb_pred_1	svm_pred_1	nb_pred_2	svm_pred_2	joint_pred_0	joint_pred_1	joint_pred_2
0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0
0.0	0.0	1.0	1.0	0.0	0.0	0.0	3.0	0.0
0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
0.0	0.0	1.0	1.0	0.0	0.0	0.0	3.0	0.0
1.0	1.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0

Group 3

nb_pred_0	svm_pred_0	nb_pred_1	svm_pred_1	nb_pred_2	svm_pred_2	joint_pred_0	joint_pred_1	joint_pred_2
0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0
0.0	0.0	1.0	1.0	0.0	0.0	0.0	3.0	0.0
0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
0.0	0.0	1.0	1.0	0.0	0.0	0.0	3.0	0.0
1.0	1.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0

Group 4

nb_pred_0	svm_pred_0	nb_pred_1	svm_pred_1	nb_pred_2	svm_pred_2	joint_pred_0	joint_pred_1	joint_pred_2
0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0
0.0	0.0	1.0	1.0	0.0	0.0	0.0	3.0	0.0
0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
0.0	0.0	1.0	1.0	0.0	0.0	0.0	3.0	0.0
1.0	1.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0

Group 5

nb_pred_0	svm_pred_0	nb_pred_1	svm_pred_1	nb_pred_2	svm_pred_2	joint_pred_0	joint_pred_1	joint_pred_2
0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0
0.0	0.0	1.0	1.0	0.0	0.0	0.0	3.0	0.0
0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
0.0	0.0	1.0	1.0	0.0	0.0	0.0	3.0	0.0

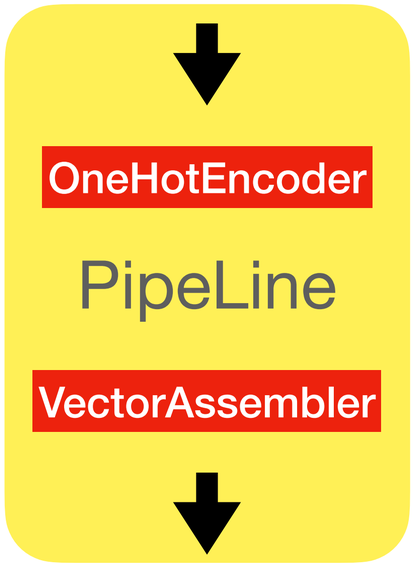
还没完，得到的 9 个 Column 要进一步处理合并变成 meta feature

Group 3

nb_pred_0	svm_pred_0	nb_pred_1	svm_pred_1	nb_pred_2	svm_pred_2	joint_pred_0	joint_pred_1	joint_pred_2
0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0
0.0	0.0	1.0	1.0	0.0	0.0	0.0	3.0	0.0
0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
0.0	0.0	1.0	1.0	0.0	0.0	0.0	3.0	0.0
1.0	1.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0

Demo

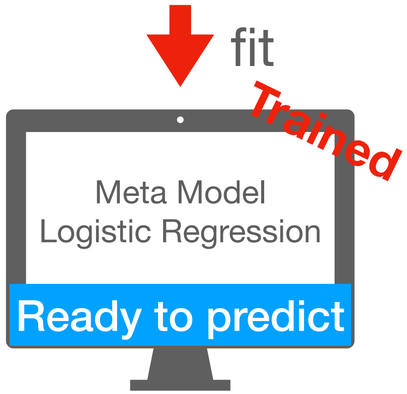
nb_pred_0	svm_pred_0	nb_pred_1	svm_pred_1	nb_pred_2	svm_pred_2	joint_pred_0	joint_pred_1	joint_pred_2
0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0
0.0	0.0	1.0	1.0	0.0	0.0	0.0	3.0	0.0
0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
0.0	0.0	1.0	1.0	0.0	0.0	0.0	3.0	0.0
1.0	1.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0



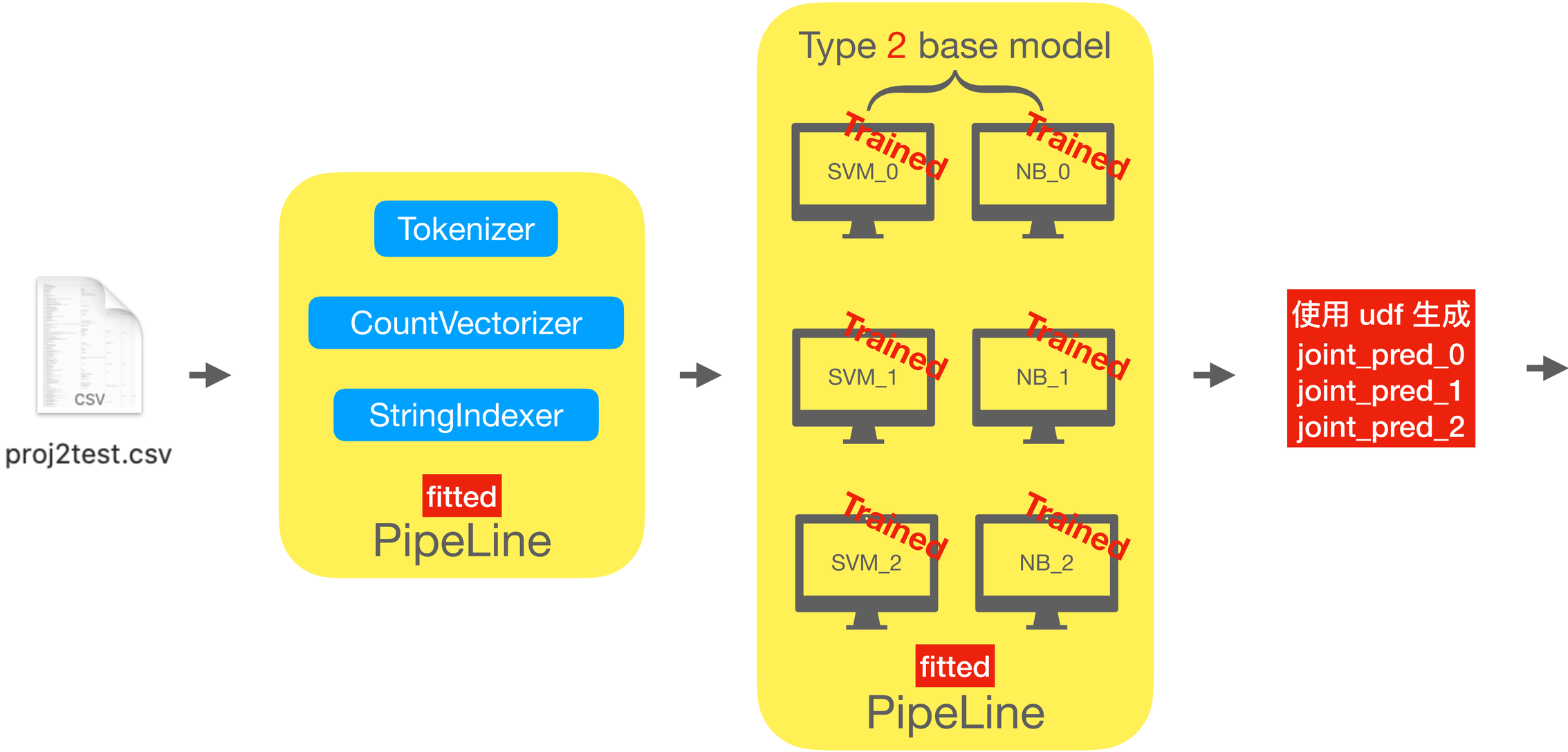
meta_features
(15,[0,1,2,3,4,6,9,13],[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0])
(15,[0,2,3,5,6,12],[1.0,1.0,1.0,1.0,1.0,1.0])
(15,[0,1,2,4,5,7,9,12],[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0])
(15,[0,2,3,5,6,12],[1.0,1.0,1.0,1.0,1.0,1.0])
(15,[1,2,4,5,9,12],[1.0,1.0,1.0,1.0,1.0,1.0])

Obtain the prediction for the test data — Task 1.3

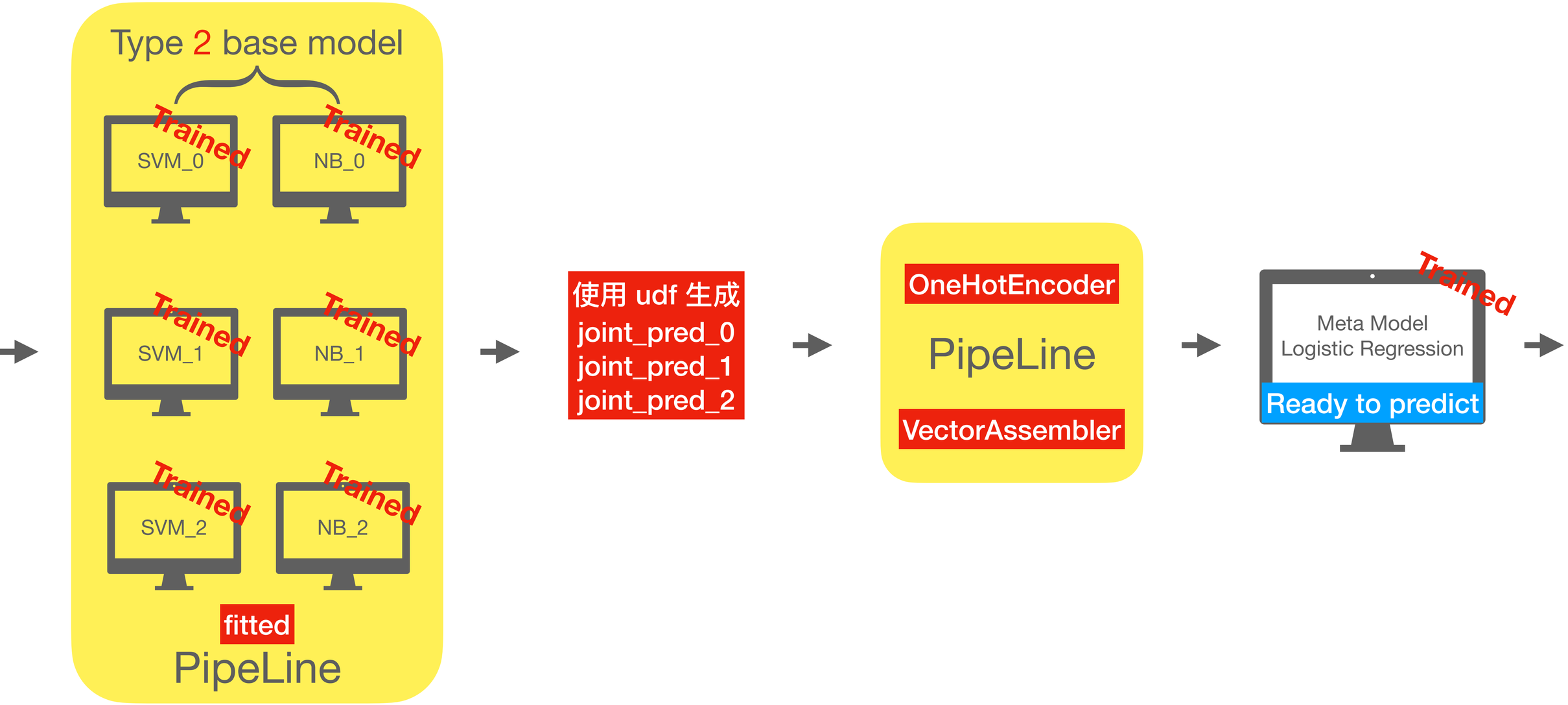
```
+-----------------------------------------------------------+  +-----+
|meta_features                                              |  |label|
+-----------------------------------------------------------+  +-----+
|(15,[0,1,2,3,4,6,9,13],[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0])|  |1.0  |
|(15,[0,2,3,5,6,12],[1.0,1.0,1.0,1.0,1.0,1.0])             |  |1.0  |
|(15,[0,1,2,4,5,7,9,12],[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0])|  |1.0  |
|(15,[0,2,3,5,6,12],[1.0,1.0,1.0,1.0,1.0,1.0])             |  |0.0  |
|(15,[1,2,4,5,9,12],[1.0,1.0,1.0,1.0,1.0,1.0])             |  |2.0  |
+-----------------------------------------------------------+  +-----+
```



Obtain the prediction for the test data — Task 1.3



Obtain the prediction for the test data — Task 1.3



代码实操

Demo

调优方案

Task 2: Report (10 points)

You are also required to submit a report named `report.pdf`. Specifically, in the report, you are at least expected to answer the following questions:

1. Evaluation of your stacking model on the test data.
2. How would you improve the performance (e.g., F1) of the stacking model.

For task 2.2, you may try from the following directions:

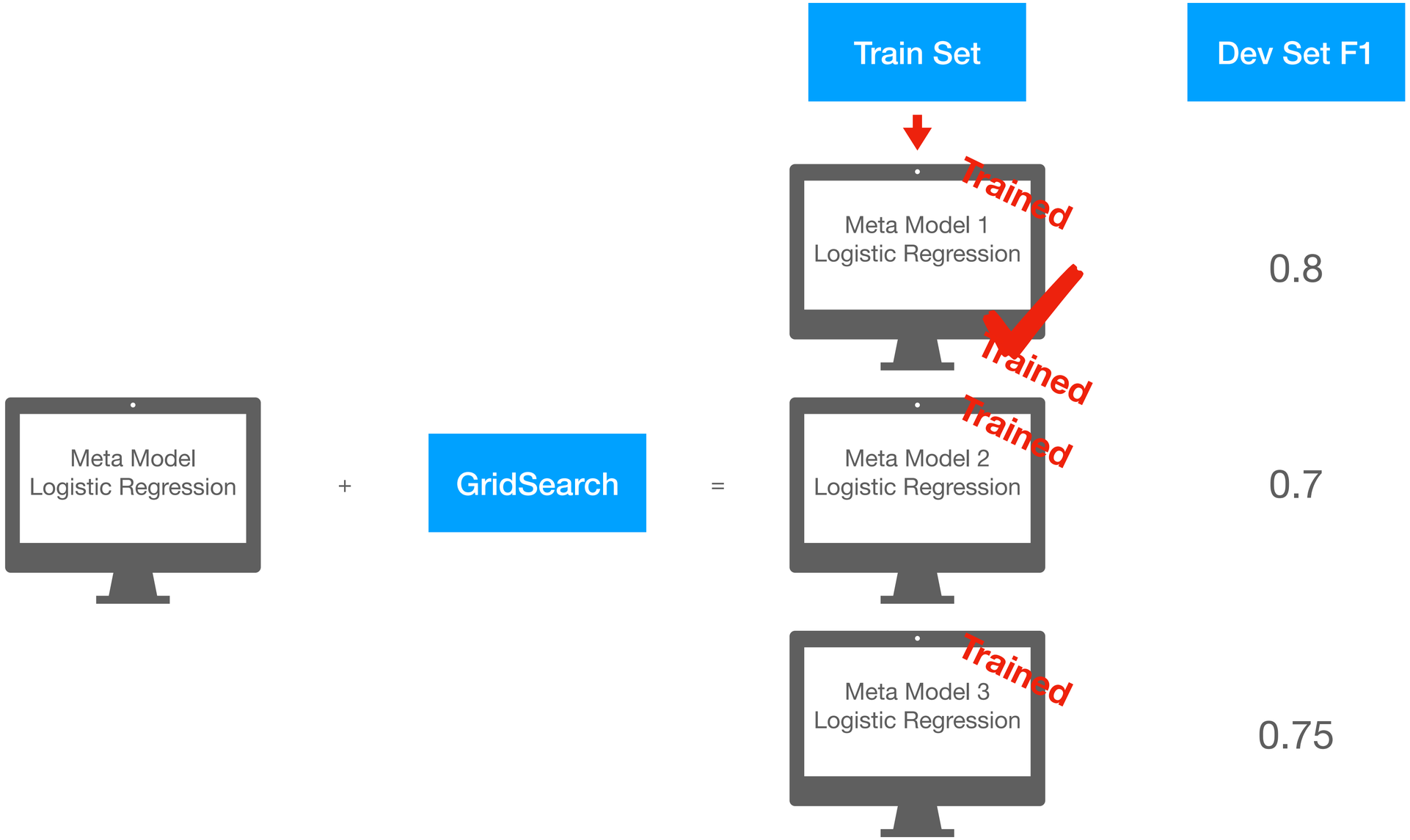
- the base feature generation
- the meta feature generation
- the hyper-parameters of base and meta models

Hint: make proper use of the development data.

1. You are **NOT** required to submit your implementation of task 2.2. Your submission is for **task 1 ONLY**.
2. **Task 2.2 is an open question**. I understand that some of you have taken machine learning courses before while some of you have not. So it is unfair to set a bar for all of you. On the other hand, I do hope tl understanding of how a stacking method works. That's why we set this task and ask you to report your findings. For this task, we expect to see your efforts rather than some numbers.

Task 2.2 More Working on Preprocessing

Task 2.2 Tuning hyper para of meta model


Train Set
Dev Set F1
Meta Model
Logistic Regression
+
GridSearch
=
Meta Model 1
Logistic Regression
Trained
0.8
Trained
Meta Model 2
Logistic Regression
Trained
0.7
Meta Model 3
Logistic Regression
Trained
0.75


Train Set
Dev Set F1
ReTrained
Meta Model 1
Logistic Regression
Test Set F1

Task 2.2 generate more meta features, or use different base/meta classifiers.

不帮忙 Debug, 有需要联系 1v1