COMP9313 Project 2 Report

Term 2, 2020

Submitted

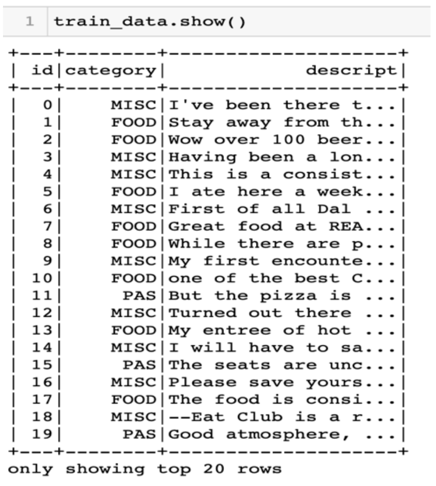
By

Vandhana Visakamurthy

Z5222191

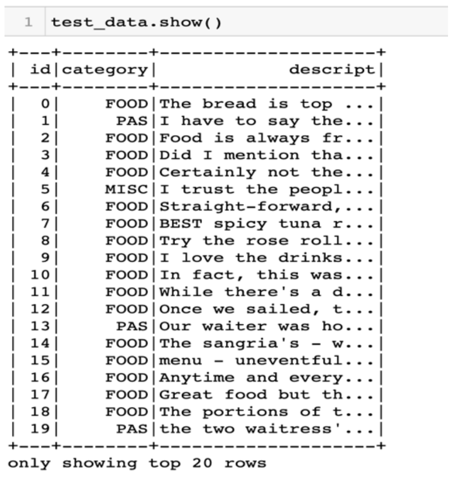
Objective

The dataset consists of food reviews by customers and there are a total of 2241 reviews in training set and 800 for development and testing respectively. The food reviews are classified into three categories – namely FOOD, PAS, and MISC.



```
train_data.show()
```

```
+---+--------+--------------------+
| 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...|
|  5|    FOOD|I ate here a week...|
|  6|    MISC|First of all Dal ...|
|  7|    FOOD|Great food at REA...|
|  8|    FOOD|While there are p...|
|  9|    MISC|My first encounte...|
| 10|    FOOD|one of the best C...|
| 11|     PAS|But the pizza is ...|
| 12|    MISC|Turned out there ...|
| 13|    FOOD|My entree of hot ...|
| 14|    MISC|I will have to sa...|
| 15|     PAS|The seats are unc...|
| 16|    MISC|Please save yours...|
| 17|    FOOD|The food is consi...|
| 18|    MISC|--Eat Club is a r...|
| 19|     PAS|Good atmosphere, ...|
+---+--------+--------------------+
only showing top 20 rows
```



```
test_data.show()
```

```
+---+--------+--------------------+
| id|category|            descript|
+---+--------+--------------------+
|  0|    FOOD|The bread is top ...|
|  1|     PAS|I have to say the...|
|  2|    FOOD|Food is always fr...|
|  3|    FOOD|Did I mention tha...|
|  4|    FOOD|Certainly not the...|
|  5|    MISC|I trust the peopl...|
|  6|    FOOD|Straight-forward,...|
|  7|    FOOD|BEST spicy tuna r...|
|  8|    FOOD|Try the rose roll...|
|  9|    FOOD|I love the drinks...|
| 10|    FOOD|In fact, this was...|
| 11|    FOOD|While there's a d...|
| 12|    FOOD|Once we sailed, t...|
| 13|     PAS|Our waiter was ho...|
| 14|    FOOD|The sangria's - w...|
| 15|    FOOD|menu - uneventful...|
| 16|    FOOD|Anytime and every...|
| 17|    FOOD|Great food but th...|
| 18|    FOOD|The portions of t...|
| 19|     PAS|the two waitress'...|
+---+--------+--------------------+
only showing top 20 rows
```

Figure 1. Training and testing data as input

The objective of this project is to use PySpark libraries such as MLib and SQL to clean, vectorize, and classify the reviews in the given dataset.

Task 1.1 (30 points): Build a Preprocessing Pipeline

The first task was to complete the code for build base features pipeline.

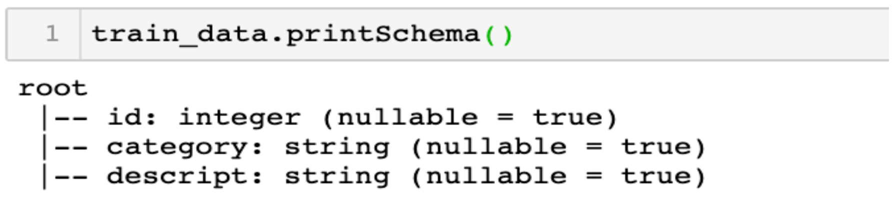
```
def build_base_features_pipeline(input_descript_col="de
script", input_category_col="category", output_feature_
col="features", output_label_col="label"):
```

This function aims to tokenize the words in every review, and then vectorize it and prepare them as feature vectors for classification.

The input columns of descript and category are mapped to features and label respectively. This is achieved through a series of steps that include:

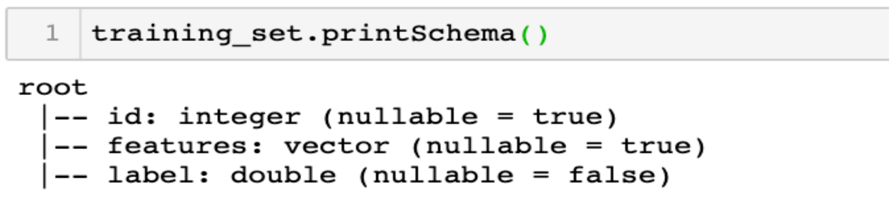
a) Tokenizing using a word tokenizer
b) Vectorizing using count vectorizer
c) Categorizing using labelling
d) Transforming the columns using a pipeline

Following these actions, the input data will be mapped into feature vectors and then labelled using gen_binary_labels. The output of Task 1.1 is as given below



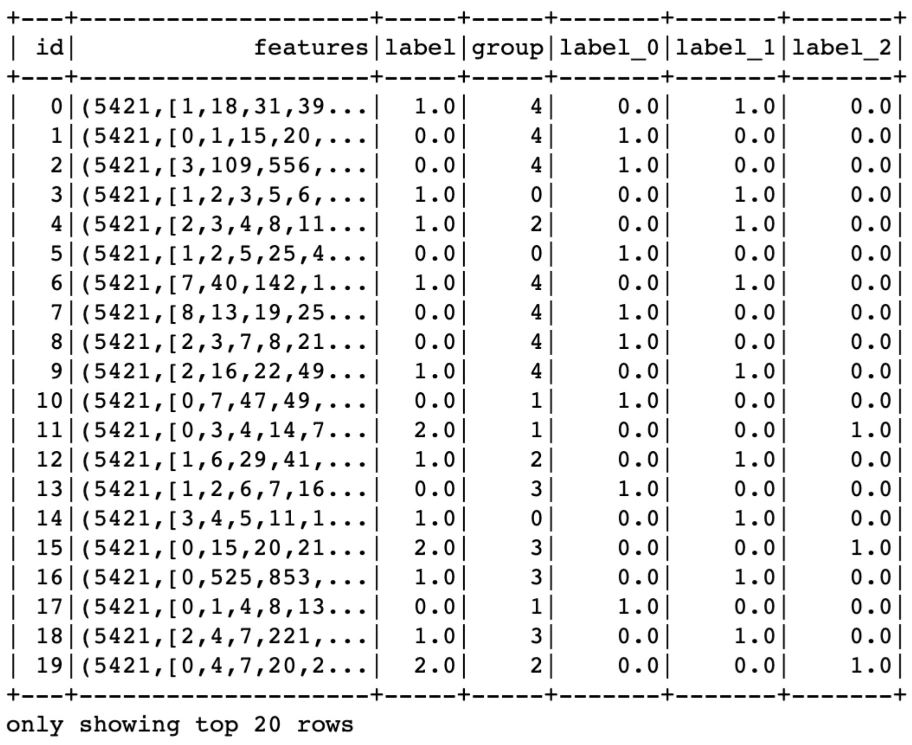
```
train_data.printSchema()
```

```
root
 |-- id: integer (nullable = true)
 |-- category: string (nullable = true)
 |-- descript: string (nullable = true)
```



```
training_set.printSchema()
```

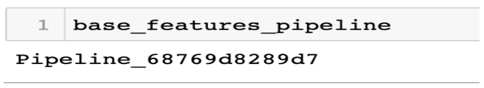
```
root
 |-- id: integer (nullable = true)
 |-- features: vector (nullable = true)
 |-- label: double (nullable = false)
```



```
+---+--------------------+-----+-----+-------+-------+-------+
| id|            features|label|group|label_0|label_1|label_2|
+---+--------------------+-----+-----+-------+-------+-------+
|  0|(5421,[1,18,31,39...|  1.0|    4|    0.0|    1.0|    0.0|
|  1|(5421,[0,1,15,20,...|  0.0|    4|    1.0|    0.0|    0.0|
|  2|(5421,[3,109,556,...|  0.0|    4|    1.0|    0.0|    0.0|
|  3|(5421,[1,2,3,5,6,...|  1.0|    0|    0.0|    1.0|    0.0|
|  4|(5421,[2,3,4,8,11...|  1.0|    2|    0.0|    1.0|    0.0|
|  5|(5421,[1,2,5,25,4...|  0.0|    0|    1.0|    0.0|    0.0|
|  6|(5421,[7,40,142,1...|  1.0|    4|    0.0|    1.0|    0.0|
|  7|(5421,[8,13,19,25...|  0.0|    4|    1.0|    0.0|    0.0|
|  8|(5421,[2,3,7,8,21...|  0.0|    4|    1.0|    0.0|    0.0|
|  9|(5421,[2,16,22,49...|  1.0|    4|    0.0|    1.0|    0.0|
| 10|(5421,[0,7,47,49,...|  0.0|    1|    1.0|    0.0|    0.0|
| 11|(5421,[0,3,4,14,7...|  2.0|    1|    0.0|    0.0|    1.0|
| 12|(5421,[1,6,29,41,...|  1.0|    2|    0.0|    1.0|    0.0|
| 13|(5421,[1,2,6,7,16...|  0.0|    3|    1.0|    0.0|    0.0|
| 14|(5421,[3,4,5,11,1...|  1.0|    0|    0.0|    1.0|    0.0|
| 15|(5421,[0,15,20,21...|  2.0|    3|    0.0|    0.0|    1.0|
| 16|(5421,[0,525,853,...|  1.0|    3|    0.0|    1.0|    0.0|
| 17|(5421,[0,1,4,8,13...|  0.0|    1|    1.0|    0.0|    0.0|
| 18|(5421,[2,4,7,221,...|  1.0|    3|    0.0|    1.0|    0.0|
| 19|(5421,[0,4,7,20,2...|  2.0|    2|    0.0|    0.0|    1.0|
+---+--------------------+-----+-----+-------+-------+-------+
only showing top 20 rows
```

Figure 2. Output of Task 1.1

The return type of this task is a pipeline.



```
base_features_pipeline
```

```
Pipeline_68769d8289d7
```

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

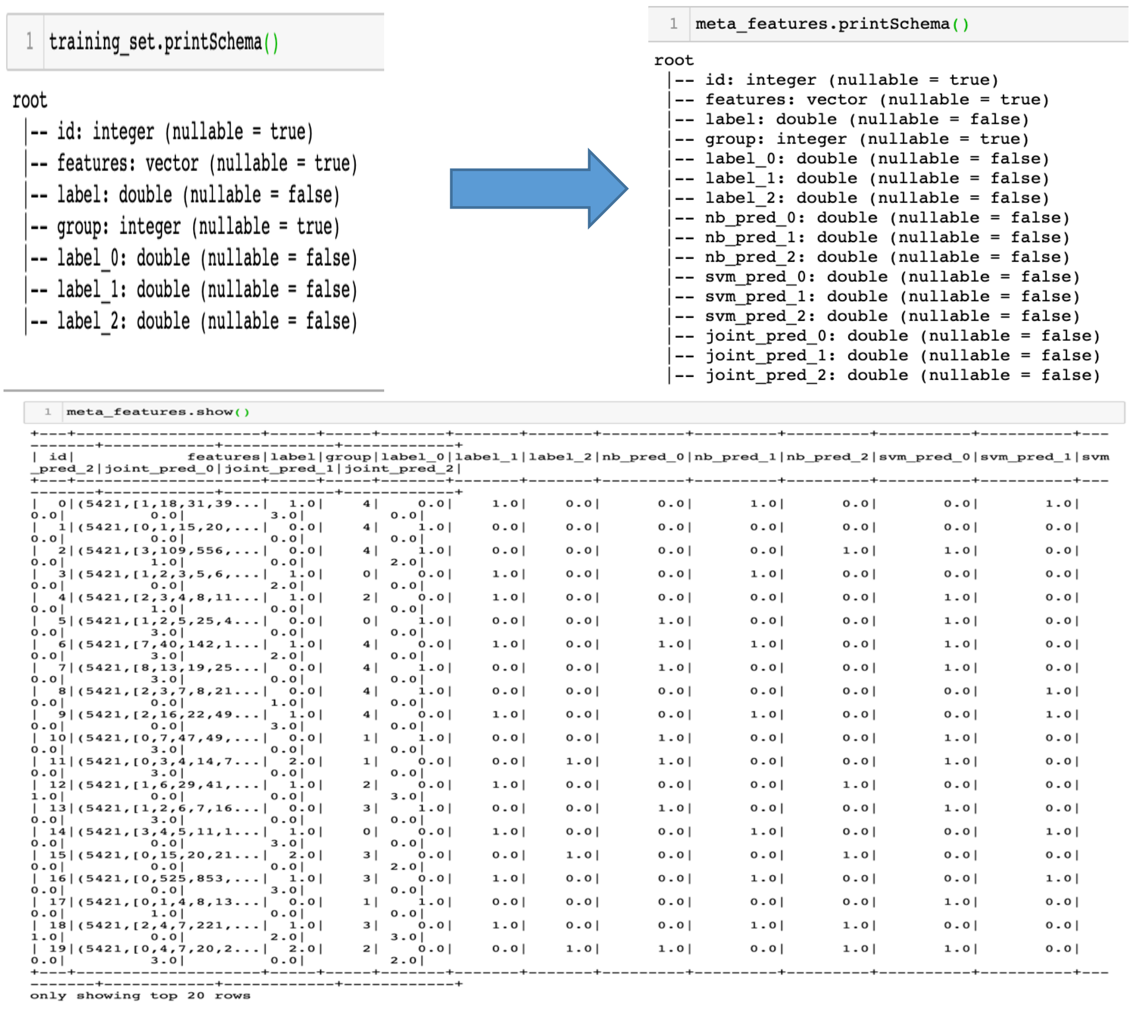
The objective of this Task is to use the training set from Task 1.1 and use that to obtain prediction values using the defined models using Naïve Bayes and SVM.

```python
def test_prediction(test_df, base_features_pipeline_mod
el, gen_base_pred_pipeline_model, gen_meta_feature_pipe
line_model, meta_classifier):
```

After obtaining the predictions of nb_x and svm_x using the respective label_x , the join t prediction values are obtained using the respective nb and svm prediction values.

The unnecessary columns are dropped to retain only the necessary columns. Join is used to vertically stack dataframe columns and union is used to horizontally stack the datafra me columns.

The joint prediction values are calculated by accessing the dataframe columns and using if else conditions called when and otherwise. This is achieved using sql.functions and w ithColumn function.



```
training_set.printSchema()
```

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

```
meta_features.printSchema()
```

```
root
 |-- id: integer (nullable = true)
 |-- features: vector (nullable = true)
 |-- label: double (nullable = false)
 |-- group: integer (nullable = true)
 |-- label_0: double (nullable = false)
 |-- label_1: double (nullable = false)
 |-- label_2: double (nullable = false)
 |-- nb_pred_0: double (nullable = false)
 |-- nb_pred_1: double (nullable = false)
 |-- nb_pred_2: double (nullable = false)
 |-- svm_pred_0: double (nullable = false)
 |-- svm_pred_1: double (nullable = false)
 |-- svm_pred_2: double (nullable = false)
 |-- joint_pred_0: double (nullable = false)
 |-- joint_pred_1: double (nullable = false)
 |-- joint_pred_2: double (nullable = false)
```

```
meta_features.show()
```

```
+---+--------------------+-----+-----+-------+-------+-------+---------+---------+---------+----------+----------+---
-------+------------+------------+------------+
| id|            features|label|group|label_0|label_1|label_2|nb_pred_0|nb_pred_1|nb_pred_2|svm_pred_0|svm_pred_1|svm
_pred_2|joint_pred_0|joint_pred_1|joint_pred_2|
+---+--------------------+-----+-----+-------+-------+-------+---------+---------+---------+----------+----------+---
-------+------------+------------+------------+
|  0|(5421,[1,18,31,39...|  1.0|    4|    0.0|    1.0|    0.0|      0.0|      1.0|      0.0|       0.0|       1.0|
0.0|         0.0|         3.0|         0.0|
|  1|(5421,[0,1,15,20,...|  0.0|    4|    1.0|    0.0|    0.0|      0.0|      0.0|      0.0|       0.0|       0.0|
0.0|         0.0|         0.0|         0.0|
|  2|(5421,[3,109,556,...|  0.0|    4|    1.0|    0.0|    0.0|      0.0|      0.0|      1.0|       1.0|       0.0|
0.0|         1.0|         0.0|         2.0|
|  3|(5421,[1,2,3,5,6,...|  1.0|    0|    0.0|    1.0|    0.0|      0.0|      1.0|      0.0|       0.0|       0.0|
0.0|         0.0|         2.0|         0.0|
|  4|(5421,[2,3,4,8,11...|  1.0|    2|    0.0|    1.0|    0.0|      0.0|      0.0|      0.0|       1.0|       0.0|
0.0|         1.0|         0.0|         0.0|
|  5|(5421,[1,2,5,25,4...|  0.0|    0|    1.0|    0.0|    0.0|      1.0|      0.0|      0.0|       1.0|       0.0|
0.0|         3.0|         0.0|         0.0|
|  6|(5421,[7,40,142,1...|  1.0|    4|    0.0|    1.0|    0.0|      1.0|      1.0|      0.0|       1.0|       0.0|
0.0|         3.0|         2.0|         0.0|
|  7|(5421,[8,13,19,25...|  0.0|    4|    1.0|    0.0|    0.0|      1.0|      0.0|      0.0|       1.0|       0.0|
0.0|         3.0|         0.0|         0.0|
|  8|(5421,[2,3,7,8,21...|  0.0|    4|    1.0|    0.0|    0.0|      0.0|      0.0|      0.0|       0.0|       1.0|
0.0|         0.0|         1.0|         0.0|
|  9|(5421,[2,16,22,49...|  1.0|    4|    0.0|    1.0|    0.0|      0.0|      1.0|      0.0|       0.0|       1.0|
0.0|         0.0|         3.0|         0.0|
| 10|(5421,[0,7,47,49,...|  0.0|    1|    1.0|    0.0|    0.0|      1.0|      0.0|      0.0|       1.0|       0.0|
0.0|         3.0|         0.0|         0.0|
| 11|(5421,[0,3,4,14,7...|  2.0|    1|    0.0|    0.0|    1.0|      1.0|      0.0|      0.0|       1.0|       0.0|
0.0|         3.0|         0.0|         0.0|
| 12|(5421,[1,6,29,41,...|  1.0|    2|    0.0|    1.0|    0.0|      0.0|      0.0|      1.0|       0.0|       0.0|
1.0|         0.0|         0.0|         3.0|
| 13|(5421,[1,2,6,7,16...|  0.0|    3|    1.0|    0.0|    0.0|      1.0|      0.0|      0.0|       1.0|       0.0|
0.0|         3.0|         0.0|         0.0|
| 14|(5421,[3,4,5,11,1...|  1.0|    0|    0.0|    1.0|    0.0|      0.0|      1.0|      0.0|       0.0|       1.0|
0.0|         0.0|         3.0|         0.0|
| 15|(5421,[0,15,20,21...|  2.0|    3|    0.0|    0.0|    1.0|      0.0|      0.0|      1.0|       0.0|       0.0|
0.0|         0.0|         0.0|         2.0|
| 16|(5421,[0,525,853,...|  1.0|    3|    0.0|    1.0|    0.0|      0.0|      1.0|      0.0|       0.0|       1.0|
0.0|         0.0|         3.0|         0.0|
| 17|(5421,[0,1,4,8,13...|  0.0|    1|    1.0|    0.0|    0.0|      0.0|      0.0|      0.0|       1.0|       0.0|
0.0|         1.0|         0.0|         0.0|
| 18|(5421,[2,4,7,221,...|  1.0|    3|    0.0|    1.0|    0.0|      0.0|      1.0|      1.0|       0.0|       0.0|
1.0|         0.0|         2.0|         3.0|
| 19|(5421,[0,4,7,20,2...|  2.0|    2|    0.0|    0.0|    1.0|      1.0|      0.0|      1.0|       1.0|       0.0|
0.0|         3.0|         0.0|         2.0|
+---+--------------------+-----+-----+-------+-------+-------+---------+---------+---------+----------+----------+---
-------+------------+------------+------------+
only showing top 20 rows
```

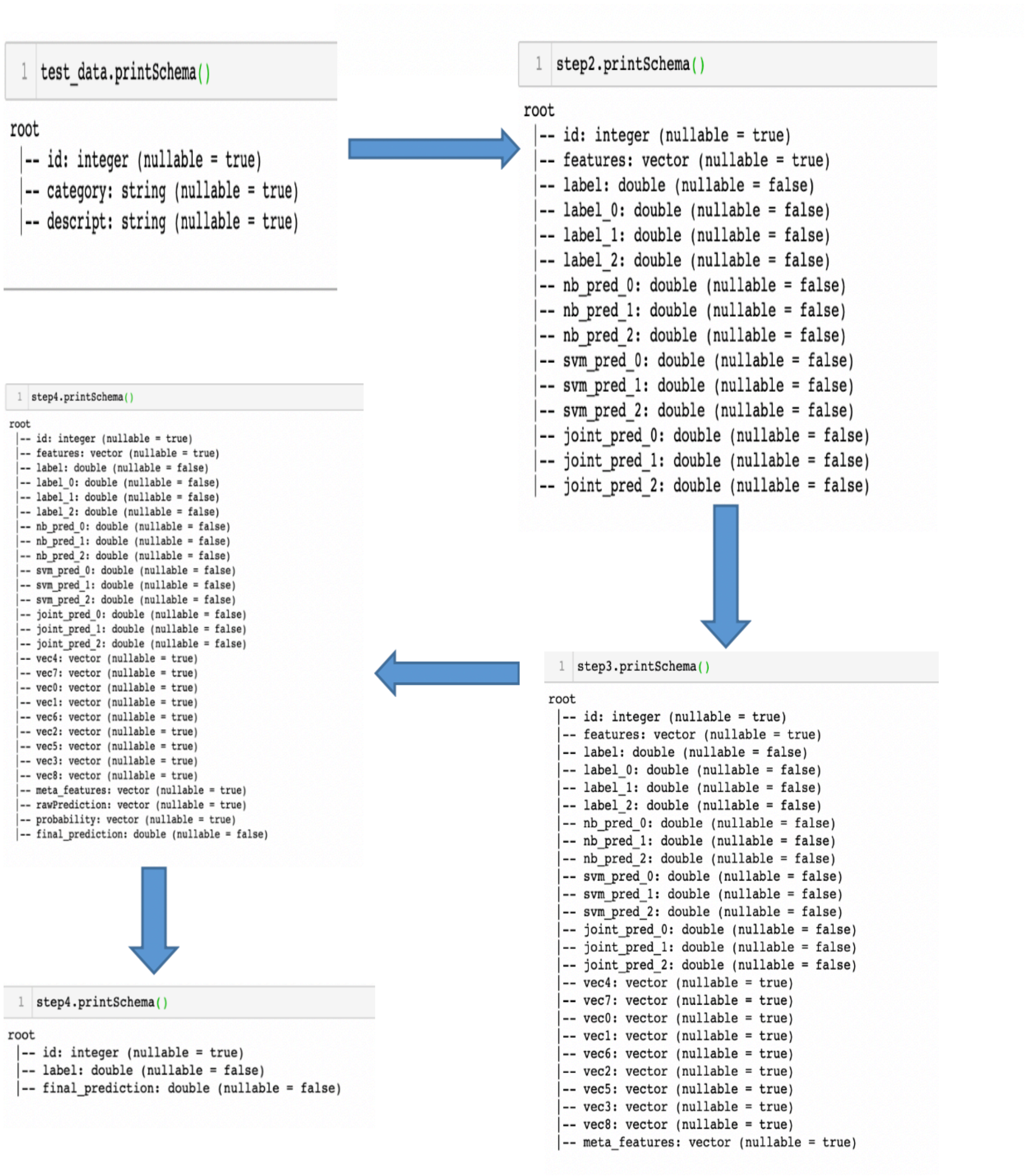
Figure 3. Output of Task 1.2

Task 1.3 (30 points): Obtain the prediction for the test data

The objective of this task is to stack all the functions developed in Task 1.1. and 1.2 to build a model to predict the values in test_data.

```
def test_prediction(test_df, base_features_pipeline_mod
el, gen_base_pred_pipeline_model, gen_meta_feature_pipe
line_model, meta_classifier):
```

The pipelines and models are already trained and hence can be used directly to predict v
alues



```
test_data.printSchema()
```

```
root
 |-- id: integer (nullable = true)
 |-- category: string (nullable = true)
 |-- descript: string (nullable = true)
```

```
step2.printSchema()
```

```
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)
 |-- nb_pred_0: double (nullable = false)
 |-- nb_pred_1: double (nullable = false)
 |-- nb_pred_2: double (nullable = false)
 |-- svm_pred_0: double (nullable = false)
 |-- svm_pred_1: double (nullable = false)
 |-- svm_pred_2: double (nullable = false)
 |-- joint_pred_0: double (nullable = false)
 |-- joint_pred_1: double (nullable = false)
 |-- joint_pred_2: double (nullable = false)
```

```
step3.printSchema()
```

```
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)
 |-- nb_pred_0: double (nullable = false)
 |-- nb_pred_1: double (nullable = false)
 |-- nb_pred_2: double (nullable = false)
 |-- svm_pred_0: double (nullable = false)
 |-- svm_pred_1: double (nullable = false)
 |-- svm_pred_2: double (nullable = false)
 |-- joint_pred_0: double (nullable = false)
 |-- joint_pred_1: double (nullable = false)
 |-- joint_pred_2: double (nullable = false)
 |-- vec4: vector (nullable = true)
 |-- vec7: vector (nullable = true)
 |-- vec0: vector (nullable = true)
 |-- vec1: vector (nullable = true)
 |-- vec6: vector (nullable = true)
 |-- vec2: vector (nullable = true)
 |-- vec5: vector (nullable = true)
 |-- vec3: vector (nullable = true)
 |-- vec8: vector (nullable = true)
 |-- meta_features: vector (nullable = true)
```

```
step4.printSchema()
```

```
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)
 |-- nb_pred_0: double (nullable = false)
 |-- nb_pred_1: double (nullable = false)
 |-- nb_pred_2: double (nullable = false)
 |-- svm_pred_0: double (nullable = false)
 |-- svm_pred_1: double (nullable = false)
 |-- svm_pred_2: double (nullable = false)
 |-- joint_pred_0: double (nullable = false)
 |-- joint_pred_1: double (nullable = false)
 |-- joint_pred_2: double (nullable = false)
 |-- vec4: vector (nullable = true)
 |-- vec7: vector (nullable = true)
 |-- vec0: vector (nullable = true)
 |-- vec1: vector (nullable = true)
 |-- vec6: vector (nullable = true)
 |-- vec2: vector (nullable = true)
 |-- vec5: vector (nullable = true)
 |-- vec3: vector (nullable = true)
 |-- vec8: vector (nullable = true)
 |-- meta_features: vector (nullable = true)
 |-- rawPrediction: vector (nullable = true)
 |-- probability: vector (nullable = true)
 |-- final_prediction: double (nullable = false)
```

```
step4.printSchema()
```

```
root
 |-- id: integer (nullable = true)
 |-- label: double (nullable = false)
 |-- final_prediction: double (nullable = false)
```

Figure 4. Output and flowchart of Task 1.3

Task 2

Evaluation of the Stacking Model

Evaluation of the stacking model is calculated using the metric F1 score.

The F1 Score is the 2*((precision * recall)/(precision + recall)).

Precision is defined as the measure of how correct the model is and recall is measure of completeness of a classifier.

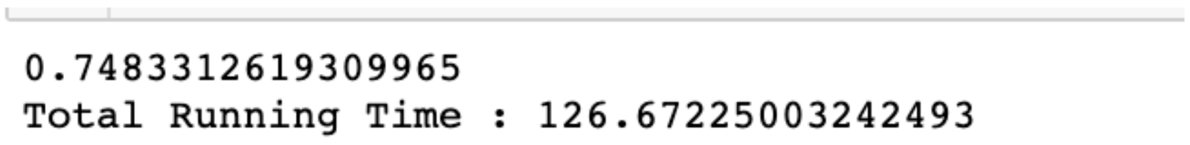
Precision is the number of True Positives divided by the number of True Positives and False Positives.

Recall is the number of True Positives divided by the number of True Positives and the number of False Negatives.

The F1 score of the model built is given below along with the running time of the program.

```
F1 score:0.7483312619309965
```

The running time is calculated by using the time the program starts execution from imports to the time the final evaluation is printed.



```
0.7483312619309965
Total Running Time : 126.67225003242493
```

Figure 5. Output consisting of Running Time and F1 evaluation score

Improving the Performance of the Stacking Model:

The performance(F-1 score) of the Stacking Model can be improved using the following methods.

1. Removing Stop Words

   While preprocessing the data, stop works can be removed from the reviews after tokenizing the reviews and before preparing the feature vectors for the models. Removing the stop words from the reviews, can essentially help consider words that actually matter as feature vectors and vectorize only those words.

2. Count Vectorizer Vs. TF-IDF vectorizer

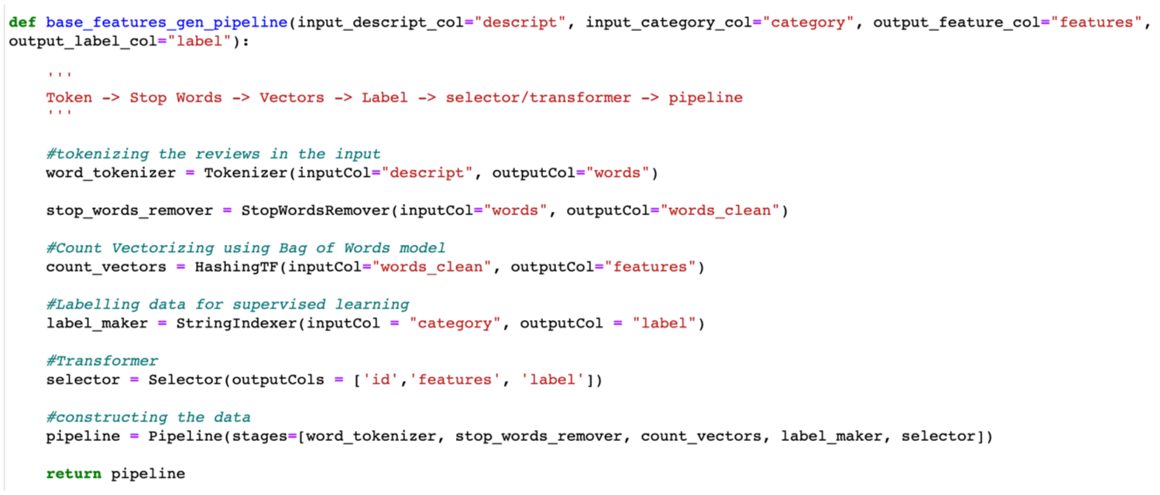
Count Vectorizer is a method that counts the frequency of the words in the document whereas TF-IDF Vectorizer assigns a vector value proportional to the frequency of term occurrence in a given document. Using TF-IDF could help improve the performance of the given model as it is used to classify text data.

3. Hyperparameter tuning

   Any model can be altered to cater to the given data set and feature vectors. These models have hyper parameters depending on the model. Cross Validation and randomness in the dataset can be increased using random functions.

   GridSearchCV and RandomSearchCV can be used to establish the right hyperparameters to get the best accuracy in prediction.

I tried to improve the efficiency by removing stop words and using TF-IDF using HashingTF and StopWordsRemover from sql.features



```
def base_features_gen_pipeline(input_descript_col="descript", input_category_col="category", output_feature_col="features",
output_label_col="label"):
    '''
    Token -> Stop Words -> Vectors -> Label -> selector/transformer -> pipeline
    '''

    #tokenizing the reviews in the input
    word_tokenizer = Tokenizer(inputCol="descript", outputCol="words")

    stop_words_remover = StopWordsRemover(inputCol="words", outputCol="words_clean")

    #Count Vectorizing using Bag of Words model
    count_vectors = HashingTF(inputCol="words_clean", outputCol="features")

    #Labelling data for supervised learning
    label_maker = StringIndexer(inputCol = "category", outputCol = "label")

    #Transformer
    selector = Selector(outputCols = ['id','features', 'label'])

    #constructing the data
    pipeline = Pipeline(stages=[word_tokenizer, stop_words_remover, count_vectors, label_maker, selector])

    return pipeline
```

Figure 6. Code to improve efficiency using preprocessing

The F-1 score increases by 1 point by including Stop word remover and TF hashing. F-1 score of close to 0.75 is achieved with a running time of 140 seconds.