# Project 2 Report

z5242692 Chenqu Zhao

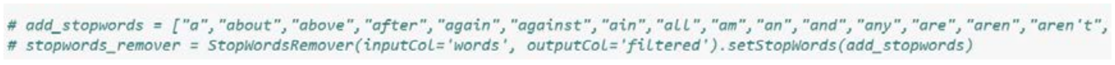
1. F1 score: 0.7483312619309965

2. I try to improve the performance of the stacking model in the following aspects. The ideas, codes and results are shown below.

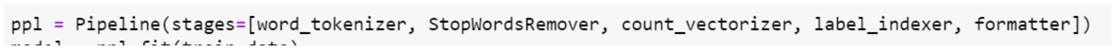
   **(1) Remove stop words**

   Ideas: Remove stop words before generating count vectors.

   Implementation:



```
# add_stopwords = ["a","about","above","after","again","against","ain","all","am","an","and","any","are","aren","aren't",
# stopwords_remover = StopWordsRemover(inputCol='words', outputCol='filtered').setStopWords(add_stopwords)
```



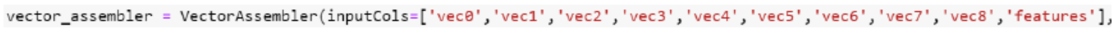
```
ppl = Pipeline(stages=[word_tokenizer, StopWordsRemover, count_vectorizer, label_indexer, formatter])
```

   Result: Disappointingly, the result of F1 score does not increase but drops instead. So, I decide to discard this operation.

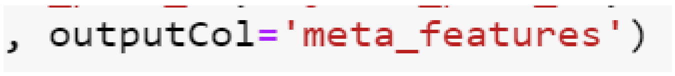
   **(2) Generate more meta features**

   Ideas: Combine the meta features with the original features of training set to generate a new feature vector with higher dimension. Then, train the meta classifier with the new meta features.

   Implementation:



```
vector_assembler = VectorAssembler(inputCols=['vec0','vec1','vec2','vec3','vec4','vec5','vec6','vec7','vec8','features'],
```



```
, outputCol='meta_features')
```

   Result:

   The F1 score increases as the figure below.

```
0.751540443024568
```

## (3) Use random forest classifier to train the meta classifier

Ideas: On the basis of step two, after reading some reports, I realize that linear regression may overfit the data when training meta classifier. To optimize this, I pick random forest as the meta classifier which is a special case of bagging. The number of trees and max depth are chosen from the best performance of several attempts.

Implementation:

```
rf_model = RandomForestClassifier(featuresCol='meta_features', labelCol='label', predictionCol='final_prediction',
```

```
, numTrees=41, maxDepth=11,seed=42)
```

```
meta_classifier = rf_model.fit(meta_features)
```

Result:

The F1 score increases as the figure below.

```
0.7584541074977676
```

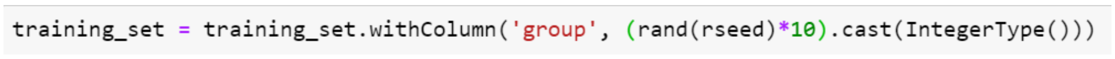
## (4) The usage of development set

Ideas: On the basis of step three, combine the train set with the development set and use 10-fold cross validation. The development set is added to optimize the base model to avoid overfitting. And according to the theory of cross validation, under some range, the larger the value of k is, the better the performance is.

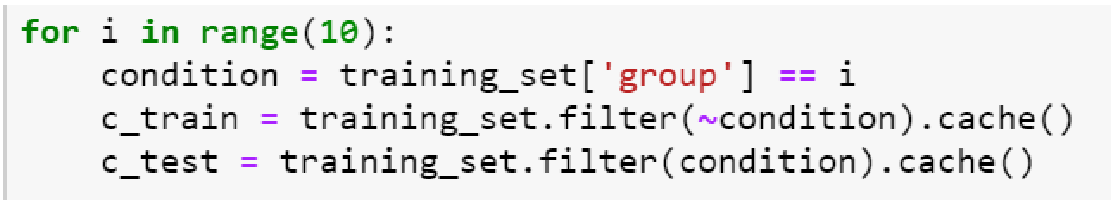
Implementation:

```
dev_data = spark.read.load("proj2dev.csv", format="csv", sep="\t", inferSchema="true", header="true")
```

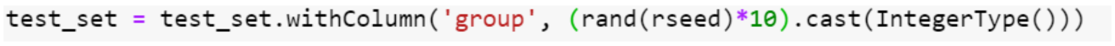
```
train_data = train_data.union(dev_data)
```



```
training_set = training_set.withColumn('group', (rand(rseed)*10).cast(IntegerType()))
```



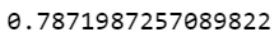
```
for i in range(10):
    condition = training_set['group'] == i
    c_train = training_set.filter(~condition).cache()
    c_test = training_set.filter(condition).cache()
```



```
test_set = test_set.withColumn('group', (rand(rseed)*10).cast(IntegerType()))
```

Result:

The F1 score increases as the figure below.



```
0.7871987257089822
```