10_model_tuning_solution

November 24, 2019

1 Answer Key to Model Tuning Quiz

```
In [1]: from pyspark.sql import SparkSession
        from pyspark.sql.functions import avg, col, concat, desc, explode, lit, min, max, split,
        from pyspark.sql.types import IntegerType

        from pyspark.ml import Pipeline
        from pyspark.ml.classification import LogisticRegression
        from pyspark.ml.evaluation import MulticlassClassificationEvaluator
        from pyspark.ml.feature import CountVectorizer, IDF, Normalizer, PCA, RegexTokenizer, St
        from pyspark.ml.regression import LinearRegression
        from pyspark.ml.tuning import CrossValidator, ParamGridBuilder

        import re
```

```
In [2]: spark = SparkSession.builder \
            .master("local") \
            .appName("Creating Features") \
            .getOrCreate()
```

```
In [3]: stack_overflow_data = 'Train_onetag_small.json'
```

```
In [4]: df = spark.read.json(stack_overflow_data)
        df.persist()
```

```
Out[4]: DataFrame[Body: string, Id: bigint, Tags: string, Title: string, oneTag: string]
```

2 Question

What is the accuracy of the best model trained with the parameter grid described above (and keeping all other parameters at their default value computed on the 10% untouched data?

2.0.1 Step 1. Train Test Split

As a first step break your data set into 90% of training data and set aside 10%. Set random seed to 42.

```
In [5]: rest, validation = df.randomSplit([0.9, 0.1], seed=42)
```

2.0.2 Step 2. Build Pipeline

```
In [6]: regexTokenizer = RegexTokenizer(inputCol="Body", outputCol="words", pattern="\\W")
        cv = CountVectorizer(inputCol="words", outputCol="TF", vocabSize=1000)
        idf = IDF(inputCol="TF", outputCol="features")
        indexer = StringIndexer(inputCol="oneTag", outputCol="label")

        lr =  LogisticRegression(maxIter=10, regParam=0.0, elasticNetParam=0)

        pipeline = Pipeline(stages=[regexTokenizer, cv, idf, indexer, lr])
```

2.0.3 Step 3. Tune Model

On the first 90% of the data let's find the most accurate logistic regression model using 3-fold cross-validation with the following parameter grid:

- CountVectorizer vocabulary size: `[1000, 5000]`
- LogisticRegression regularization parameter: `[0.0, 0.1]`
- LogisticRegression max Iteration number: `[10]`

```
In [7]: paramGrid = ParamGridBuilder() \
            .addGrid(cv.vocabSize,[1000, 5000]) \
            .addGrid(lr.regParam,[0.0, 0.1]) \
            .build()


        crossval = CrossValidator(estimator=pipeline,
                                  estimatorParamMaps=paramGrid,
                                  evaluator=MulticlassClassificationEvaluator(),
                                  numFolds=3)
```

```
In [8]: cvModel_q1 = crossval.fit(rest)
```

```
In [9]: cvModel_q1.avgMetrics
```

```
Out[9]: [0.30442826193831585,
         0.23194595198118026,
         0.36592646756682135,
         0.2842976418403327]
```

```
In [10]: results = cvModel_q1.transform(validation)
```

2.0.4 Step 4: Compute Accuracy of Best Model

```
In [11]: print(results.filter(results.label == results.prediction).count())
         print(results.count())
```

```
3892
9919
```

```
In [12]: 3896/9937.0

Out[12]: 0.3920700412599376

In [ ]:
```