云计算与大数据项目

MLlib 下逻辑回归模型（LR）实现句子的单字预测

课程信息：	云计算与大数据
指导老师：	Dr Jun Liu & Prof. Gang Cheng
学生姓名：	王润一
校园卡号：	320190940491
学生邮箱：	wangrunyi19@lzu.edu.cn

目录

1. 项目概述与环境简介

本项目将采用机器学习中的逻辑回归模型（Logistic Regression，LR 模型）实现对于句子的单字预测，查看所需要预测的句子中是否包含“Spark”这个单字。项目数据集较小，利用 MLlib 中的 dataframe 创建相关函数，采用手动输入方式进行原始数据集输入，使用准确率（Accuracy）作为算法评价指标，来判断输出结果的可靠性。

语言环境：Python3.8.2
编译器：Jupyter Notebook
机器学习环境：pySpark MLlib

2. 实验原理与算法简介

2.1 Spark 优势与特点

作为大数据计算框架 MapReduce 的继任者，Spark 具备以下优势特性：

- 高效性。不同于 MapReduce 将中间计算结果放入磁盘中，Spark 采用内存存储中间计算结果，减少了迭代运算的磁盘 IO，并通过并行计算 DAG 图的优化，减少了不同任务之间的依赖，降低了延迟等待时间。内存计算下，Spark 比 MapReduce 快 100 倍。

- 易用性。不同于 MapReduce 仅支持 Map 和 Reduce 两种编程算子，Spark 提供了超过 80 种不同的 Transformation 和 Action 算子，如：MapReduce 等，并且采用函数式编程风格，实现相同的功能需要的代码量极大缩小。

- 通用性。Spark 提供了统一的解决方案。Spark 可以用于批处理，交互式查询（Spark SQL）实时流处理（Spark Streaming）机器学习（Spark MLlib）和图计算（GraphX）。

- 兼容性。Spark 能够跟很多开源工程兼容使用。如 Spark 可以使用 Hadoop 的 YARN 和 Apache Mesos 作为它的资源管理和调度器，并且 Spark 可以读取多种数据源，如 HDFS、HBase、MySQL 等。

2.2 Spark MLlib 简介

MLlib 是 Spark 的机器学习库，包括以下主要功能：

- 实用工具：线性代数，统计，数据处理等工具
- 特征工程：特征提取，特征转换，特征选择
- 常用算法：分类，回归，聚类，协同过滤，降维
- 模型优化：模型评估，参数优化。

MLlib 库包括两个不同的部分：

- pyspark.mllib 包含基于 rdd 的机器学习算法 API，目前不再更新，以后将被丢弃，不建议使用。
- pyspark.ml 包含基于 DataFrame 的机器学习算法 API，可以用来构建机器学习工作流 Pipeline，推荐使用。

MLlib 相关的基本概念：

- DataFrame: MLlib 中数据的存储形式，其列可以存储特征向量，标签，以及原始的文本，图像。

- Transformer：转换器。具有 transform 方法。通过附加一个或多个列将一个 DataFrame 转换成另外一个 DataFrame。

- Estimator：估计器。具有 fit 方法。它接受一个 DataFrame 数据作为输入后经过训练，产生一个转换器 Transformer。

- Pipeline：流水线。具有 setStages 方法。顺序将多个 Transformer 和 1 个 Estimator 串联起来，得到一个流水线模型。

2.3 逻辑回归模型 (Logistic Regression) 简介

逻辑回归模型 (Logistic Regression) 算法特点：

- 逻辑回归模型需要选择 predictor 以及它们的具体形式，这其中包含了他们之间的关联项，这一点也保证了逻辑回归在偏小的数据集上面也能得到不错的结果。
- 逻辑回归虽然名字中有回归，但实际上不能用于连续数值的预测，模型建立起来之后，对于大量新数据的分类的计算都会变得十分快速且消耗的计算资源很少。
- 逻辑回归的 precditors 可以是 categorical 的，也可以是 numerical 的，这一点上同多元线性回归（multiple linear regression）一致，只不过多元线性回归是对连续的 Y 值进行预测，然而逻辑回归仅仅是把记录进行分类。逻辑回归的决策边缘是线性的，因此逻辑回归是线性模型。

逻辑回归主要用于下列两种情况：

- 新数据的类别未知，需要根据其 predictors 将新数据的类别分到已知的类别中（分类问题）
- 数据的类别已知，需要知道能区分开这些类别的 factor（profilling 问题）

逻辑回归的基本思路，模型：

- 首先获取各个类的 propensities（属于该类的概率），比如 Y=1 的 propensity 就是
- 然后设置 cutoff value，并根据该值同 propensities 比较的结果，将新数据分类，比如若则分类为 0，否则为 1

逻辑回归模型的效果评估：

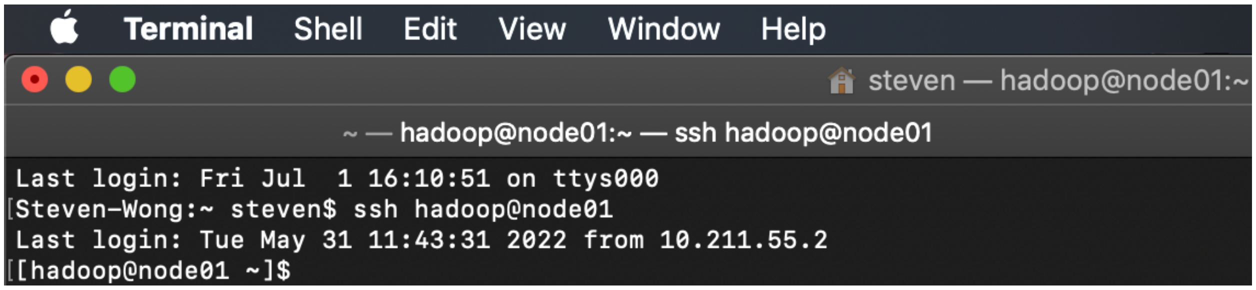
分类问题（Classification）的评估也适用于逻辑回归模型（logistic regression），比如混淆矩阵（confusion matrix），以及准确率（Accuracy），在本项目中我们采用准确率（Accuracy）作为最终的评价指标。

3. 项目实现和具体操作

3.1 前期工作和数据集准备

首先，使用 Parallel Desktop 启动 Linux 虚拟机 node01，打开 Mac 终端 Terminal，使用 ssh 进行远程免密登录。

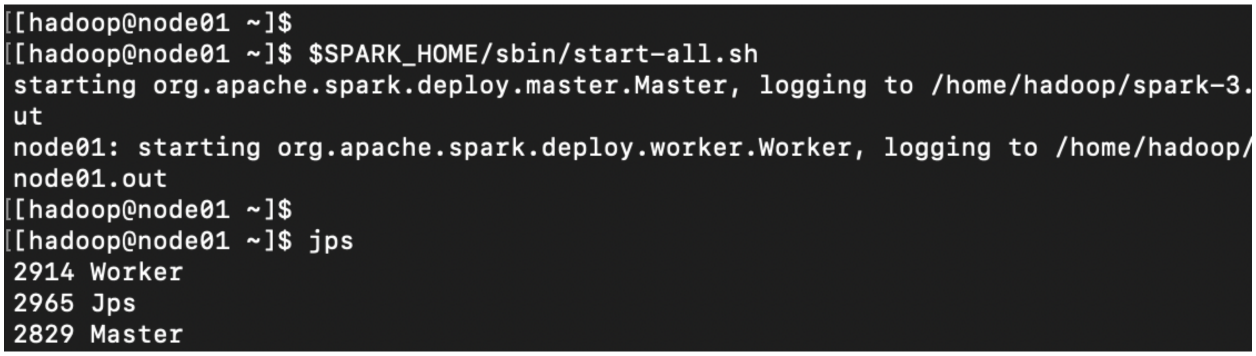
$ ssh hadoop@node01


Terminal Shell Edit View Window Help
steven — hadoop@node01:~
~ — hadoop@node01:~ — ssh hadoop@node01

```
Last login: Fri Jul  1 16:10:51 on ttys000
[Steven-Wong:~ steven$ ssh hadoop@node01
Last login: Tue May 31 11:43:31 2022 from 10.211.55.2
[[hadoop@node01 ~]$
```

启动 Spark 服务，查看 spark 进程

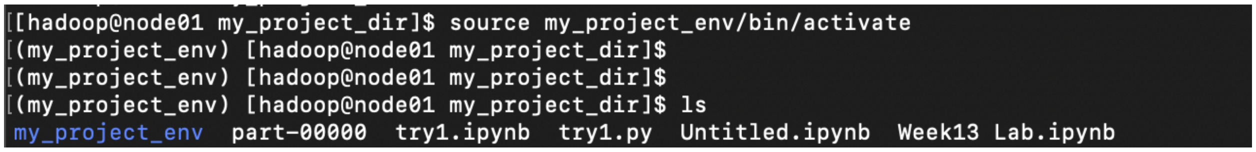
$ $SPARK_HOME/sbin/start-all.sh
$ jps



```
[[hadoop@node01 ~]$
[[hadoop@node01 ~]$ $SPARK_HOME/sbin/start-all.sh
starting org.apache.spark.deploy.master.Master, logging to /home/hadoop/spark-3.
ut
node01: starting org.apache.spark.deploy.worker.Worker, logging to /home/hadoop/
node01.out
[[hadoop@node01 ~]$
[[hadoop@node01 ~]$ jps
2914 Worker
2965 Jps
2829 Master
```

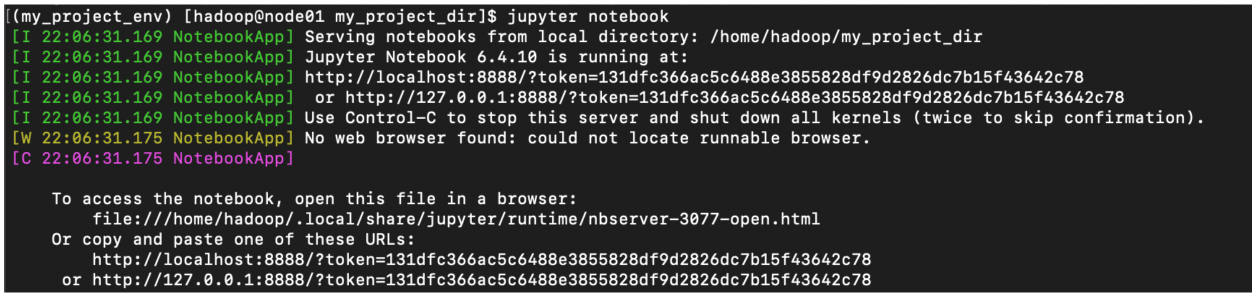
来到项目目录下，激活 Python 虚拟环境

$ virtualenv my_project_env
$ source my_project_env/bin/activate



```
[[hadoop@node01 my_project_dir]$ source my_project_env/bin/activate
[(my_project_env) [hadoop@node01 my_project_dir]$
[(my_project_env) [hadoop@node01 my_project_dir]$
[(my_project_env) [hadoop@node01 my_project_dir]$ ls
my_project_env  part-00000  try1.ipynb  try1.py  Untitled.ipynb  Week13 Lab.ipynb
```

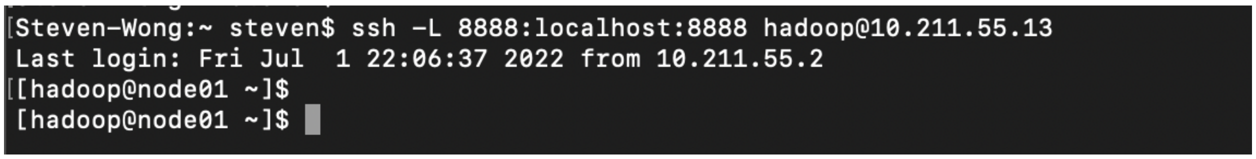
此时，输入 jupyter notebook，便可打开 Notebook。



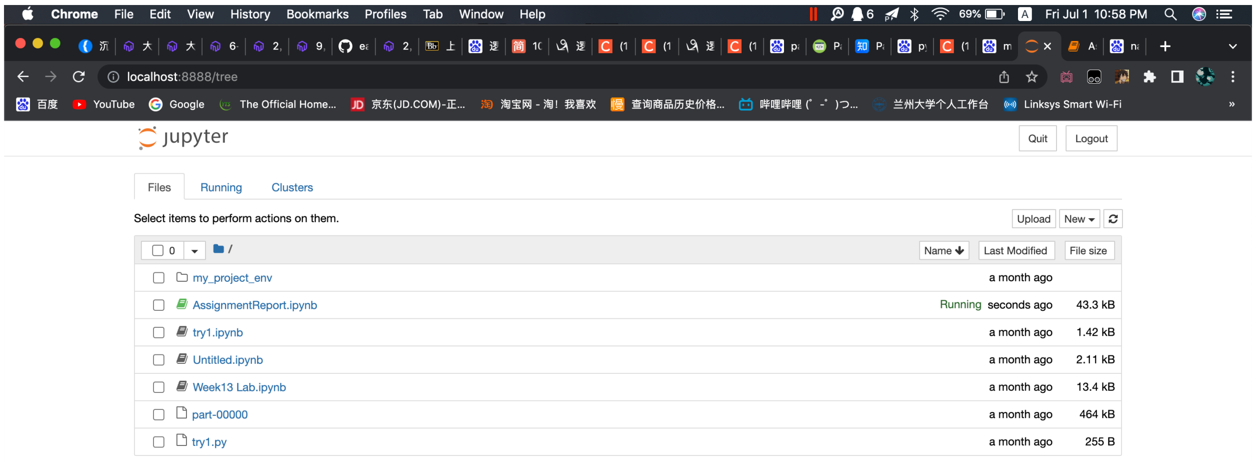
```
[(my_project_env) [hadoop@node01 my_project_dir]$ jupyter notebook
[I 22:06:31.169 NotebookApp] Serving notebooks from local directory: /home/hadoop/my_project_dir
[I 22:06:31.169 NotebookApp] Jupyter Notebook 6.4.10 is running at:
[I 22:06:31.169 NotebookApp] http://localhost:8888/?token=131dfc366ac5c6488e3855828df9d2826dc7b15f43642c78
[I 22:06:31.169 NotebookApp]  or http://127.0.0.1:8888/?token=131dfc366ac5c6488e3855828df9d2826dc7b15f43642c78
[I 22:06:31.169 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[W 22:06:31.175 NotebookApp] No web browser found: could not locate runnable browser.
[C 22:06:31.175 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///home/hadoop/.local/share/jupyter/runtime/nbserver-3077-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=131dfc366ac5c6488e3855828df9d2826dc7b15f43642c78
     or http://127.0.0.1:8888/?token=131dfc366ac5c6488e3855828df9d2826dc7b15f43642c78
```

此时，Mac 端通过 ssh 远程连接 Linux node01 的 8888 端口，使得 Mac 上的 Google Chrome 游览器能够访问已经开启的 Notebook。



```
[Steven-Wong:~ steven$ ssh -L 8888:localhost:8888 hadoop@10.211.55.13
 Last login: Fri Jul  1 22:06:37 2022 from 10.211.55.2
[[hadoop@node01 ~]$
 [hadoop@node01 ~]$ 
```



Select items to perform actions on them.

```
my_project_env                          a month ago
AssignmentReport.ipynb      Running     seconds ago   43.3 kB
try1.ipynb                              a month ago   1.42 kB
Untitled.ipynb                          a month ago   2.11 kB
Week13 Lab.ipynb                        a month ago   13.4 kB
part-00000                              a month ago   464 kB
try1.py                                 a month ago   255 B
```

实验数据集介绍：

实验数据集是一个 Spark DataFrame，由 15 个 observation 组成，其中前 10 个作为训练集，后 5 个作为测试集。DataFrame 由三个列（column），其中第一列为 id 号，开始为 1 号，结束为 15 号；第二列为文本内容（text）；第三列为分类结果（class），其中 0.0 代表在文本中 “spark” 单字没有出现，1.0 代表 “spark” 单字在文本中出现了。

```
dftrain = spark.createDataFrame([(0,"a b c d e spark",1.0),
                                 (1,"a c f g",0.0),
                                 (2,"learn spark and say hello world",1.0),
                                 (3,"hadoop mapreduce big data analysis",0.0),
                                 (4,"I love spark and spark is fast", 1.0),
                                 (5,"big data needs ",0.0),
                                 (6,"spark is much faster than mapreduce",1.0),
                                 (7,"spark is based on hadoop",1.0),
                                 (8,"spark fast ",1.0),
                                 (9,"fast does not mean reliable",0.0),
                                 (10,"hello spark and hadoop",0.0)],["id","text","label"])
dftrain.show()
```

```
# Model Testing
dftest = spark.createDataFrame([(11,"spark job",1.0),
                                (12,"hello world and hadoop",0.0),
                                (13,"a b c d e",0.0),
                                (14,"fast blast cast mapreduce",0.0),
                                (15,"spark is easy to use.",1.0)]).toDF("id","text","label")
dftest.show()

dfresult = model.transform(dftest)
dfresult.selectExpr("text","features","probability","prediction").show()
```

3.2 定义模型和训练模型

模型创建代码如下：

```
# Model Construction
tokenizer = Tokenizer().setInputCol("text").setOutputCol("words")
print(type(tokenizer))

hashingTF = HashingTF().setNumFeatures(100) \
   .setInputCol(tokenizer.getOutputCol()) \
   .setOutputCol("features")
print(type(hashingTF))

lr = LogisticRegression().setLabelCol("label")
# print(lr.explainParams)
lr.setFeaturesCol("features").setMaxIter(10).setRegParam(0.2)
print(type(lr))

pipe = Pipeline().setStages([tokenizer,hashingTF,lr])
print(type(pipe))
```

```
<class 'pyspark.ml.feature.Tokenizer'>
<class 'pyspark.ml.feature.HashingTF'>
<class 'pyspark.ml.classification.LogisticRegression'>
<class 'pyspark.ml.pipeline.Pipeline'>
```

测试模型以及获得的预测结果如下：

```
# Model Training
model = pipe.fit(dftrain)
print(type(model))
```

```
<class 'pyspark.ml.pipeline.PipelineModel'>
```

```
+--------------------+--------------------+--------------------+----------+
|                text|            features|         probability|prediction|
+--------------------+--------------------+--------------------+----------+
|           spark job|(100,[57,86],[1.0...|[0.37840207538908...|       1.0|
|hello world and h...|(100,[60,85,89,91...|[0.65410393426222...|       0.0|
|           a b c d e|(100,[50,65,67,68...|[0.21379785526741...|       1.0|
|fast blast cast m...|(100,[16,32,50,63...|[0.37796022540695...|       1.0|
|spark is easy to ...|(100,[9,21,60,86,...|[0.17295405224269...|       1.0|
+--------------------+--------------------+--------------------+----------+
```

3.3 使用模型和结果评估

使用内置评估函数 printSchema()来获得模型评估总结：

```
# Model Evaluation
dfresult.printSchema()
```

```
root
 |-- id: long (nullable = true)
 |-- text: string (nullable = true)
 |-- label: double (nullable = true)
 |-- words: array (nullable = true)
 |    |-- element: string (containsNull = true)
 |-- features: vector (nullable = true)
 |-- rawPrediction: vector (nullable = true)
 |-- probability: vector (nullable = true)
 |-- prediction: double (nullable = false)
```

项目采用准确率 (Accuracy) 来评估最终的模型预测结果：

```
# Evaluation via Accuracy Index
evaluator = MulticlassClassificationEvaluator().setMetricName("f1") \
    .setPredictionCol("prediction").setLabelCol("label")

#print(evaluator.explainParams())
accuracy  = evaluator.evaluate(dfresult)
print("\n accuracy = {}".format(accuracy))
```

```
 accuracy = 0.5666666666666667
```

4. 项目反思和后续优化

4.1 逻辑回归模型的优缺点

LR 优点：

- 直接对分类的可能性建模，无需事先假设数据分布，避免假设分布不准确带来的问题

- 不仅预测出类别，还可得到近似概率预测
- 对率函数是任意阶可导凸函数，有很好得数学性质，很多数值优化算法可直接用于求取最优解
- 容易使用和解释，计算代价低
- LR 对时间和内存需求上相当高效
- 可应用于分布式数据，并且还有在线算法实现，用较小资源处理较大数据
- 对数据中小噪声鲁棒性很好，并且不会受到轻微多重共线性影响
- 因为结果是概率，可用作排序模型

LR 缺点：

- 容易欠拟合，分类精度不高
- 数据特征有缺失或特征空间很大时效果不好

4.2 结果反思与后续优化

可以看到，模型最终的准确率只有 56.7%，还有很大的提升空间。使得模型准确率低的一个关键原因是数据集太小，可供训练的训练样本太少，因此模型没有能够充分准确地把握数据集的关键特征。因此，后面的改进中，我会寻找更大样本的数据集重新进行建模。与此同时，在模型参数的设置上，可以进一步尝试优化，以获得更好的准确率。

5. 参考资料

[1] https://blog.csdn.net/weixin_42331985/article/details/124126019

[2] https://zhuanlan.zhihu.com/p/397770615

[3] https://blog.csdn.net/iqdutao/article/details/109478633

[4] https://www.heywhale.com/mw/project/5fe6b1d15e24ed0030230d8b

6. 详细代码

Cloud Computing and Big Data, Spring 2022

Assignment Report

Set Date: Friday, June 10, 2022
Due Date: 11:59pm, Friday, July 01, 2022

This assignment counts for 30% of the final grade, along with 40% lab practical, and 30% final exam
Submit your completed Jupyter notebook file through the course website in the Blackboard Learn system

---

Assignment Title: Word Prediction Using Logistic Regression

Student Name: Runyi Wang

Student ID: 320190940491

Assignment Requirements

This assignment examines the level of knowledge the students have learned from the course. The assignment must be done on a cloud computing platform, for example, the Apache Spark platform.

Abstract

(Briefly summarize the cloud-based assignment including the problem, data sets, models, evaluation, and final findings.)

本项目将采用机器学习中的逻辑回归模型实现对于句子的单字预测，查看所需要预测的句子中是否包含“Spark”这个单字。项目数据集较小， 利用 ML 中的 创建相关函数，采用手动输入方式进行原始数据集输入，使用 准确率(A)作为算法评价指标，来判断输出结果的可靠性。

1. Introduction

(Introduce the assignment, describe the objectives, and present the results. This section will provide an overview of the entire assignment including the description of the data sets and the cloud-based data analytics methods and techniques the team used for analyzing the data to address the problem. Highlight the key findings.)

本项目将采用机器学习中的逻辑回归模型实现对于句子的单字预测，查看所需要预测的句子中是否包含“Spark”这个单字。项目数据集较小， 利用 ML 中的 创建相关函数，采用手动输入方式进行原始数据集输入，使用 准确率(A)作为算法评价指标，来判断输出结果的可靠性。

2. Problem Definition

(Define the problem that will be solved in this cloud computing assignment, what are the challenges of this problem?)

问题定义：MLlib 下逻辑回归模型(LR) 实现句子的单字预测

3. Data

(Describe the origin, format, and charateristics of the data. For example, where the data was obtained from, what data is contained in the set. The size of the dataset in terms of instances and features. What are the classes of the dataset)

Data Description: A small dataset that contains

In [1]:
```
# Import Relevant Modules and Libraries

# Modules for DataFrame Structure
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
```

In [7]:
```
# Modelues for pySpark Machine Learning Tool (Mlib)
from pyspark.ml.feature import Tokenizer,HashingTF
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import MulticlassClassificationEvaluator,Bina
from pyspark.ml import Pipeline,PipelineModel
from pyspark.ml.linalg import Vector
from pyspark.sql import Row

# Modeules for Distuibuted Computing
import pyspark
import findspark
from pyspark.sql import SparkSession
from pyspark.storagelevel import StorageLevel

spark = SparkSession.builder \
        .appName("dbscan") \
        .config("master","local[4]") \
        .enableHiveSupport() \
        .getOrCreate()

sc = spark.sparkContext
```

```
/home/hadoop/my_project_dir/my_project_env/lib/python3.6/site-packages
/pyspark/context.py:238: FutureWarning: Python 3.6 support is deprecat
ed in Spark 3.2.
  FutureWarning
```

In [2]:
```
data_train_raw = [(0,"a b c d e spark",1.0),
                  (1,"a c f g",0.0),
                  (2,"learn spark and say hello world",1.0),
                  (3,"hadoop mapreduce big data analysis",0.0),
                  (4,"I love spark and spark is fast", 1.0),
                  (5,"big data needs ",0.0),
                  (6,"spark is much faster than mapreduce",1.0),
                  (7,"spark is based on hadoop",1.0),
                  (8,"spark fast ",1.0),
                  (9,"fast does not mean reliable",0.0),
                  (10,"hello spark and hadoop",0.0)]

data_test_raw = [(11,"spark job",1.0),
                 (12,"hello world and hadoop",0.0),
                 (13,"a b c d e",0.0),
                 (14,"fast blast cast mapreduce",0.0),
                 (15,"spark is easy to use.",1.0)]

df_train = pd.DataFrame(data_train_raw, columns=["id","text","label"])
df_test = pd.DataFrame(data_test_raw, columns=["id","text","label"])
```

In [3]:
```
# Check df_train
df_train
```

Out[3]:

	id	text	label
0	0	a b c d e spark	1.0
1	1	a c f g	0.0
2	2	learn spark and say hello world	1.0
3	3	hadoop mapreduce big data analysis	0.0
4	4	I love spark and spark is fast	1.0
5	5	big data needs	0.0
6	6	spark is much faster than mapreduce	1.0
7	7	spark is based on hadoop	1.0
8	8	spark fast	1.0
9	9	fast does not mean reliable	0.0
10	10	hello spark and hadoop	0.0

In [8]:
```
dftrain = spark.createDataFrame([(0,"a b c d e spark",1.0),
                                 (1,"a c f g",0.0),
                                 (2,"learn spark and say hello world",1.
                                 (3,"hadoop mapreduce big data analysis"
                                 (4,"I love spark and spark is fast", 1.
                                 (5,"big data needs ",0.0),
                                 (6,"spark is much faster than mapreduce
                                 (7,"spark is based on hadoop",1.0),
                                 (8,"spark fast ",1.0),
                                 (9,"fast does not mean reliable",0.0),
                                 (10,"hello spark and hadoop",0.0)],["id
dftrain.show()
```

```
+---+--------------------+-----+
| id|                text|label|
+---+--------------------+-----+
|  0|     a b c d e spark|  1.0|
|  1|             a c f g|  0.0|
|  2|learn spark and s...|  1.0|
|  3|hadoop mapreduce ...|  0.0|
|  4|I love spark and ...|  1.0|
|  5|     big data needs |  0.0|
|  6|spark is much fas...|  1.0|
|  7|spark is based on...|  1.0|
|  8|         spark fast |  1.0|
|  9|fast does not mea...|  0.0|
| 10|hello spark and h...|  0.0|
+---+--------------------+-----+
```

In [4]:
```
# Check df_test
df_test
```

Out[4]:

	id	text	label
0	11	spark job	1.0
1	12	hello world and hadoop	0.0
2	13	a b c d e	0.0
3	14	fast blast cast mapreduce	0.0
4	15	spark is easy to use.	1.0

4. Exploratory Data Analysis

(Cleanse, wrangle, transform, and analyze the data. Compute descriptive statistics and visualize the values of features and correlations between different features. Explain the process and the results.)

In [5]:
```
# Print the dataset columns
df_train.columns
```

Out[5]:
```
Index(['id', 'text', 'label'], dtype='object')
```

In [6]:
```
# Display below the first 5 rows of the DataFrame we created
df_train.head(5)
```

Out[6]:

	id	text	label
0	0	a b c d e spark	1.0
1	1	a c f g	0.0
2	2	learn spark and say hello world	1.0
3	3	hadoop mapreduce big data analysis	0.0
4	4	I love spark and spark is fast	1.0

In [7]:
```
# Returns the row and column count of a dataset
df_train.shape
```

Out[7]:
```
(11, 3)
```

In [8]:
```
# Returns statistics about the numerical columns in a dataset
df_train.describe()
```

Out[8]:

	id	label
count	11.000000	11.000000
mean	5.000000	0.545455
std	3.316625	0.522233
min	0.000000	0.000000
25%	2.500000	0.000000
50%	5.000000	1.000000
75%	7.500000	1.000000
max	10.000000	1.000000

In [9]:
```
# Returns the data type of each column
df_train.dtypes
```

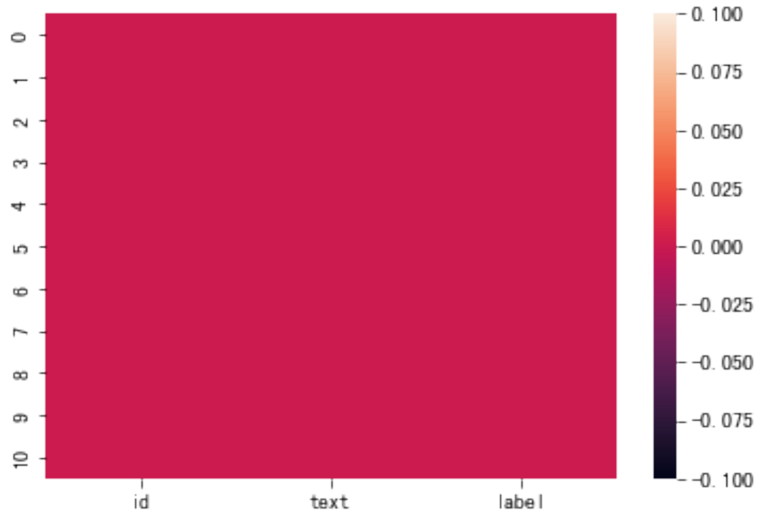
Out[9]:
```
id          int64
text       object
label     float64
dtype: object
```

In [10]:
```
# Checking if there are any missing values in the dataset
df_train.isnull().sum().any()
```

Out[10]:
```
False
```

In [11]:
```
# Visualization to check for missing values
# I use heatmap function in "seaborn" library to identify NaN values!
sns.heatmap(df_train.isna())
```

Out[11]:
```
<AxesSubplot:>
```



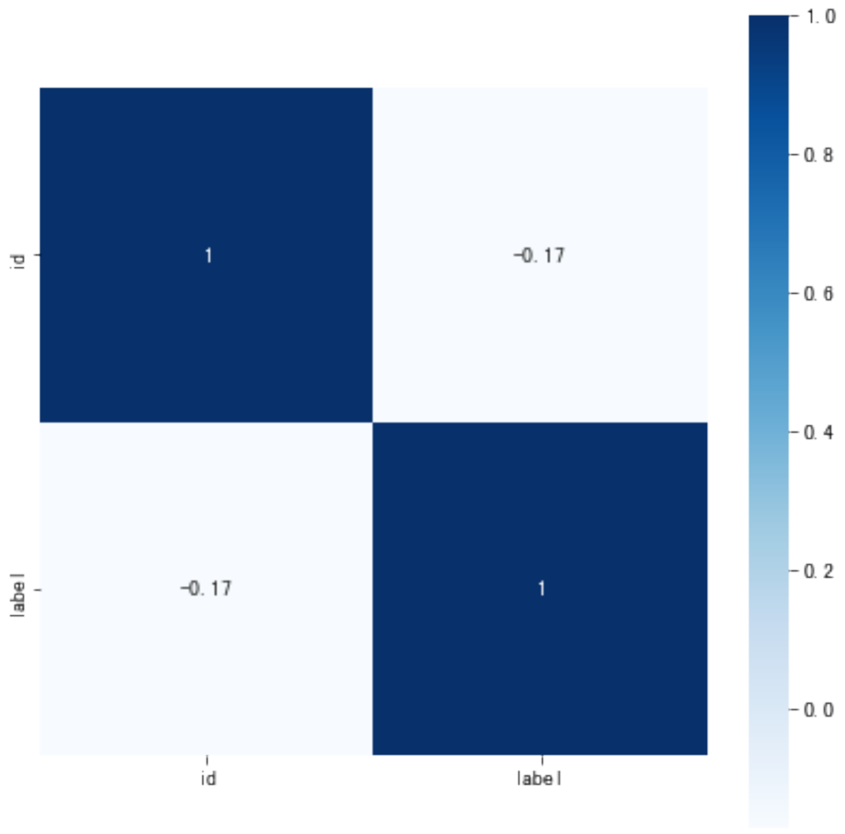
In [12]:
```
# Counts the number of missing values in the whole dataset
df_train.isnull().sum().sum()
```

Out[12]:
```
0
```

In [13]:
```
# Counts null values per column
df_train.isnull().sum()
```

Out[13]:
```
id       0
text     0
label    0
dtype: int64
```

In [14]:
```python
# Heatmap to show Correlations between different features
corr = df_train.corr()
plt.figure(figsize=(7,7))
sns.heatmap(corr, annot=True, cmap='Blues',square=True)
plt.show()
```



5. Modeling and Evaluation

(Build analytic models and evaluate the models on the data. Provide detail on the data analytics process you have undertaken, e.g. data cleansing and preparation, feature selection, data modelling techniques and your rationale for selecting these.The analytics should be guided by the problem defined earlier. Explain the process and results along with code snippets. For the results, application of different methods and metrics for model evaluation. Visualization of results.)

In [9]:
```
# Model Construction
tokenizer = Tokenizer().setInputCol("text").setOutputCol("words")
print(type(tokenizer))

hashingTF = HashingTF().setNumFeatures(100) \
   .setInputCol(tokenizer.getOutputCol()) \
   .setOutputCol("features")
print(type(hashingTF))

lr = LogisticRegression().setLabelCol("label")
#print(lr.explainParams)
lr.setFeaturesCol("features").setMaxIter(10).setRegParam(0.2)
print(type(lr))

pipe = Pipeline().setStages([tokenizer,hashingTF,lr])
print(type(pipe))
```

```
<class 'pyspark.ml.feature.Tokenizer'>
<class 'pyspark.ml.feature.HashingTF'>
<class 'pyspark.ml.classification.LogisticRegression'>
<class 'pyspark.ml.pipeline.Pipeline'>
```

In [10]:
```
# Model Training
model = pipe.fit(dftrain)
print(type(model))
```

```
<class 'pyspark.ml.pipeline.PipelineModel'>
```

In [11]:
```
# Model Testing
dftest = spark.createDataFrame([(11,"spark job",1.0),
                                (12,"hello world and hadoop",0.0),
                                (13,"a b c d e",0.0),
                                (14,"fast blast cast mapreduce",0.0),
                                (15,"spark is easy to use.",1.0)]).toDF(
dftest.show()

dfresult = model.transform(dftest)
dfresult.selectExpr("text","features","probability","prediction").show()
```

```
+---+--------------------+-----+
| id|                text|label|
+---+--------------------+-----+
| 11|           spark job|  1.0|
| 12|hello world and h...|  0.0|
| 13|           a b c d e|  0.0|
| 14|fast blast cast m...|  0.0|
| 15|spark is easy to ...|  1.0|
+---+--------------------+-----+

+--------------------+--------------------+--------------------+------
----+
|                text|            features|         probability|predic
tion|
+--------------------+--------------------+--------------------+------
----+
|           spark job|(100,[57,86],[1.0...|[0.37840207538908...|
1.0|
|hello world and h...|(100,[60,85,89,91...|[0.65410393426222...|
0.0|
|           a b c d e|(100,[50,65,67,68...|[0.21379785526741...|
1.0|
|fast blast cast m...|(100,[16,32,50,63...|[0.37796022540695...|
1.0|
|spark is easy to ...|(100,[9,21,60,86,...|[0.17295405224269...|
1.0|
+--------------------+--------------------+--------------------+------
----+
In [12]: # Model Evaluation
         dfresult.printSchema()
```

```
root
 |-- id: long (nullable = true)
 |-- text: string (nullable = true)
 |-- label: double (nullable = true)
 |-- words: array (nullable = true)
 |    |-- element: string (containsNull = true)
 |-- features: vector (nullable = true)
 |-- rawPrediction: vector (nullable = true)
 |-- probability: vector (nullable = true)
 |-- prediction: double (nullable = false)
```

```
In [13]: # Evaluation via Accuracy Index
         evaluator = MulticlassClassificationEvaluator().setMetricName("f1") \
             .setPredictionCol("prediction").setLabelCol("label")

         #print(evaluator.explainParams())
         accuracy  = evaluator.evaluate(dfresult)
         print("\n accuracy = {}".format(accuracy))
```

```

 accuracy = 0.5666666666666667
```

6. Discussions and Conclusions

*(Provide a summary of the various results that you have obtained from the analysis and what these tell us. Briefly describe what you have done using cloud computing techniques and what you discovered. Discuss any shortcomings of the process and results. Propose future work. **Finally, discuss the lessons learned from doing the assignment**.)*

可以看到，模型最终的准确率只有 56.7%，还有很大的提升空间。使得模型准确率低的一 个关键原因是数据集太小，可供训练的训练样本太少，因此模型没有能够充分准确地把握 数据集的关键特征。因此，后面的改进中，我会寻找更大样本的数据集重新进行建模。与 此同时，在模型参数的设置上，可以进一步尝试优化，以获得更好的准确率。

7. References

[1] htps://blog.csdn.net/weixin_42331985/article/details/124126019
[2] https://zhuanlan.zhihu.com/p/397770615
[3] https://blog.csdn.net/iqdutao/article/details/109478633
[4] https://www.heywhale.com/mw/project/5fe6b1d15e24ed0030230d8b

In [ ]:

```

```