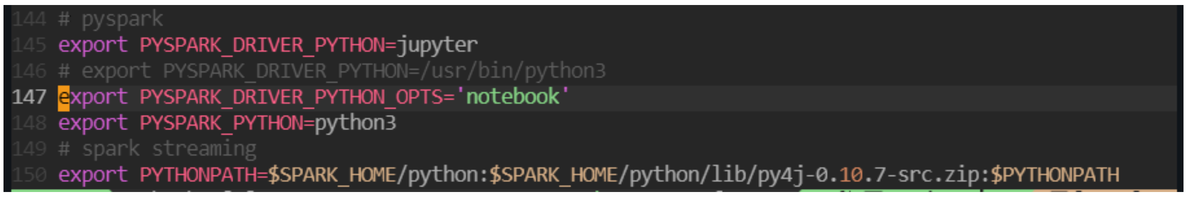
SparkMLlib

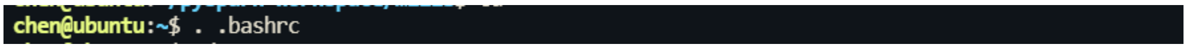
实验目的：　　熟练掌握sparkmllib的使用

准备工作：



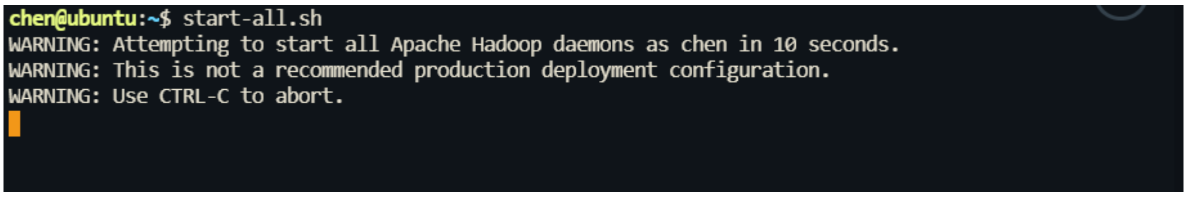
```
# pyspark
export PYSPARK_DRIVER_PYTHON=jupyter
# export PYSPARK_DRIVER_PYTHON=/usr/bin/python3
export PYSPARK_DRIVER_PYTHON_OPTS='notebook'
export PYSPARK_PYTHON=python3
# spark streaming
export PYTHONPATH=$SPARK_HOME/python:$SPARK_HOME/python/lib/py4j-0.10.7-src.zip:$PYTHONPATH
```

激活环境变量



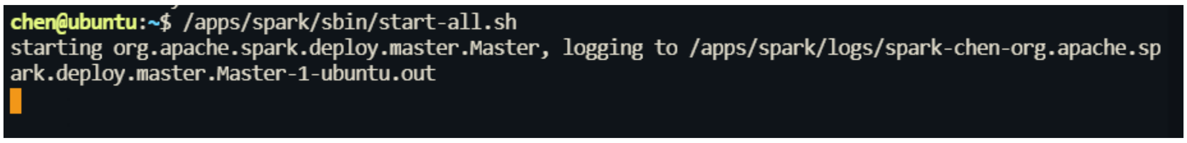
```
chen@ubuntu:~$ . .bashrc
```

启动hadoop



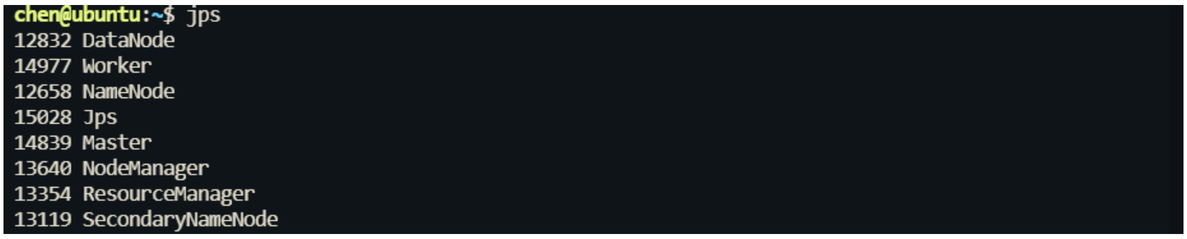
```
chen@ubuntu:~$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as chen in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
```

启动spark



```
chen@ubuntu:~$ /apps/spark/sbin/start-all.sh
starting org.apache.spark.deploy.master.Master, logging to /apps/spark/logs/spark-chen-org.apache.sp
ark.deploy.master.Master-1-ubuntu.out
```

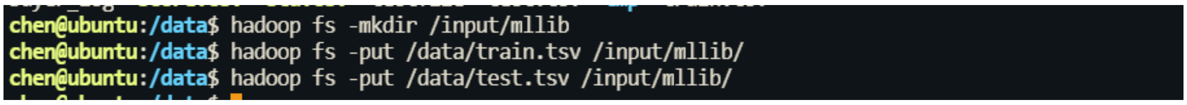
检查



```
chen@ubuntu:~$ jps
12832 DataNode
14977 Worker
12658 NameNode
15028 Jps
14839 Master
13640 NodeManager
13354 ResourceManager
13119 SecondaryNameNode
```

上传数据

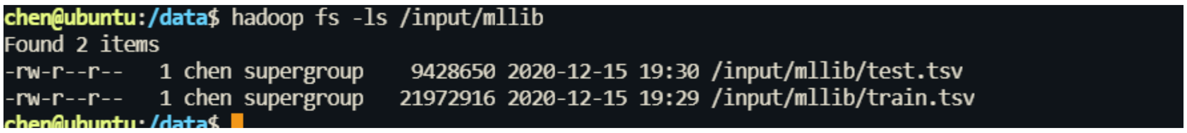
```
hadoop fs -mkdir /input/mllib
hadoop fs -put /data/train.tsv /input/mllib/
hadoop fs -put /data/test.tsv /input/mllib/
```



```
chen@ubuntu:/data$ hadoop fs -mkdir /input/mllib
chen@ubuntu:/data$ hadoop fs -put /data/train.tsv /input/mllib/
chen@ubuntu:/data$ hadoop fs -put /data/test.tsv /input/mllib/
```

检查

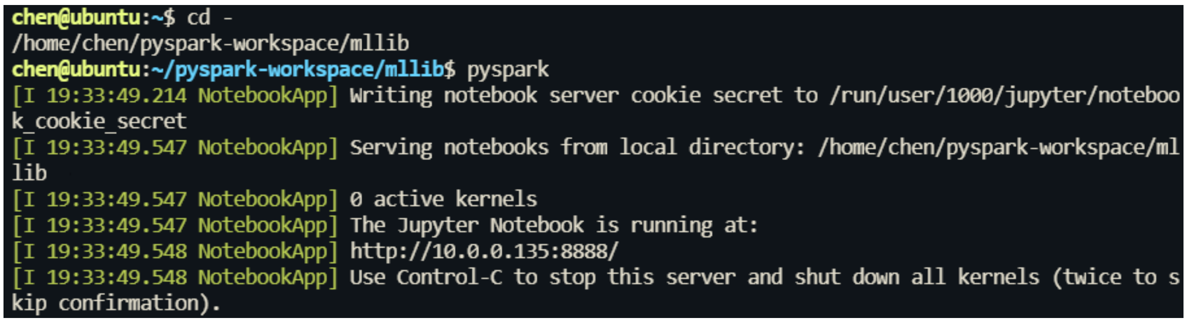
```
hadoop fs -ls /input/mllib
```



```
chen@ubuntu:/data$ hadoop fs -ls /input/mllib
Found 2 items
-rw-r--r--   1 chen supergroup    9428650 2020-12-15 19:30 /input/mllib/test.tsv
-rw-r--r--   1 chen supergroup   21972916 2020-12-15 19:29 /input/mllib/train.tsv
```

启动pyspark

创建目录~/pyspark-workspace/mllib/，在该目录中启动 Pyspark，创建一个 Notebook。



```
chen@ubuntu:~$ cd -
/home/chen/pyspark-workspace/mllib
chen@ubuntu:~/pyspark-workspace/mllib$ pyspark
[I 19:33:49.214 NotebookApp] Writing notebook server cookie secret to /run/user/1000/jupyter/noteboo
k_cookie_secret
[I 19:33:49.547 NotebookApp] Serving notebooks from local directory: /home/chen/pyspark-workspace/ml
lib
[I 19:33:49.547 NotebookApp] 0 active kernels
[I 19:33:49.547 NotebookApp] The Jupyter Notebook is running at:
[I 19:33:49.548 NotebookApp] http://10.0.0.135:8888/
[I 19:33:49.548 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to s
kip confirmation).
```

读取数据

读取hdfs上的train.tsv文件，并查看数据项

In [1]:
```
from pyspark import SparkContext,SparkConf
from pyspark.sql import SQLContext
conf = SparkConf()
conf.setAppName('Streaming').set('spark.io.compression.codec','snappy')
conf.setMaster('local[2]')
sc = SparkContext(conf = conf)
sqlContext = SQLContext(sc)
```

In [2]:
```
row_df = sqlContext.read.format("csv")\
.option("header","true")\
.option("delimiter","\t")\
.load("/input/mllib/train.tsv")
print(row_df.count())
```

```
7395
```

In [3]:
```
row_df.printSchema()
```

```
root
 |-- url: string (nullable = true)
 |-- urlid: string (nullable = true)
 |-- boilerplate: string (nullable = true)
 |-- alchemy_category: string (nullable = true)
 |-- alchemy_category_score: string (nullable = true)
 |-- avglinksize: string (nullable = true)
 |-- commonlinkratio_1: string (nullable = true)
 |-- commonlinkratio_2: string (nullable = true)
 |-- commonlinkratio_3: string (nullable = true)
 |-- commonlinkratio_4: string (nullable = true)
 |-- compression_ratio: string (nullable = true)
 |-- embed_ratio: string (nullable = true)
 |-- framebased: string (nullable = true)
 |-- frameTagRatio: string (nullable = true)
 |-- hasDomainLink: string (nullable = true)
 |-- html_ratio: string (nullable = true)
 |-- image_ratio: string (nullable = true)
 |-- is_news: string (nullable = true)
 |-- lengthyLinkDomain: string (nullable = true)
 |-- linkwordscore: string (nullable = true)
 |-- news_front_page: string (nullable = true)
 |-- non_markup_alphanum_characters: string (nullable = true)
 |-- numberOfLinks: string (nullable = true)
 |-- numwords_in_url: string (nullable = true)
 |-- parametrizedLinkRatio: string (nullable = true)
 |-- spelling_errors_ratio: string (nullable = true)
 |-- label: string (nullable = true)
```

In [4]:
```
row_df.show(1)
```

```
+--------------------+-----+--------------------+----------------+----------------------+-----------+-----------------+------
-----------+-----------------+-----------------+-----------------+-----------+----------+-------------+-------------+--------
```

```
---+-----------+-------+-----------------+-------------+---------------+------------------------------+-------------+--------
-------+---------------------+---------------------+-----+
|                 url|urlid|         boilerplate|alchemy_category|alchemy_category_score|avglinksize|commonlinkrati
o_1|commonlinkratio_2|commonlinkratio_3|commonlinkratio_4|compression_ratio|embed_ratio|framebased|
frameTagRatio|hasDomainLink| html_ratio|image_ratio|is_news|lengthyLinkDomain|linkwordscore|news_fr
ont_page|non_markup_alphanum_characters|numberOfLinks|numwords_in_url|parametrizedLinkRatio|spelli
ng_errors_ratio|label|
+--------------------+-----+--------------------+----------------+----------------------+-----------+-----------------+------
-----------+-----------------+-----------------+-----------------+-----------+----------+-------------+-------------+--------
---+-----------+-------+-----------------+-------------+---------------+------------------------------+-------------+--------
-------+---------------------+---------------------+-----+
|http://www.bloomb...| 4042|"{""title"":""IBM...|        business|              0.789131|2.055555556|      0.6764705
88|      0.205882353|      0.047058824|      0.023529412|      0.443783175|          0|         0|   0.09077381|
 0|0.245831182|0.003883495|      1|                1|           24|              0|                          5424|          170|
8|           0.152941176|           0.079129575|    0|
+--------------------+-----+--------------------+----------------+----------------------+-----------+-----------------+------
-----------+-----------------+-----------------+-----------------+-----------+----------+-------------+-------------+--------
---+-----------+-------+-----------------+-------------+---------------+------------------------------+-------------+--------
-------+---------------------+---------------------+-----+
only showing top 1 row
```

In [5]:
```
row_df.select('url','urlid','boilerplate','alchemy_category','alchemy_category_score','is_news','label').show(10
)
```

```
+--------------------+-----+--------------------+------------------+----------------------+-------+-----+
|                 url|urlid|         boilerplate|  alchemy_category|alchemy_category_score|is_news|label|
+--------------------+-----+--------------------+------------------+----------------------+-------+-----+
|http://www.bloomb...| 4042|"{""title"":""IBM...|          business|              0.789131|      1|    0|
|http://www.popsci...| 8471|"{""title"":""The...|        recreation|              0.574147|      1|    1|
|http://www.menshe...| 1164|"{""title"":""Fru...|            health|              0.996526|      1|    1|
|http://www.dumbli...| 6684|"{""title"":""10 ...|            health|              0.801248|      1|    1|
|http://bleacherre...| 9006|"{""title"":""The...|            sports|              0.719157|      1|    0|
|http://www.conven...| 7018|"{""url"":""conve...|                 ?|                     ?|      ?|    0|
|http://gofashionl...| 8685|"{""title"":""fas...|arts_entertainment|               0.22111|      1|    1|
|http://www.inside...| 3402|"{""url"":""insid...|                 ?|                     ?|      ?|    0|
|http://www.valetm...|  477|"{""title"":""Val...|                 ?|                     ?|      1|    1|
|http://www.howswe...| 6731|"{""url"":""howsw...|                 ?|                     ?|      ?|    1|
+--------------------+-----+--------------------+------------------+----------------------+-------+-----+
only showing top 10 rows
```

编写 DataFrames UDF 用户自定义函数，将数据中的？转换为 0

In [6]:
```python
from pyspark.sql.functions import udf
def replace_question(x):
    return ("0" if x=="?" else x)
replace_question=udf(replace_question)
```

导入 col 模块及 pyspark.sql.types 模块，后续可以使用 col 模块读取字段数据，使用 pyspark.sql.types 模块转换数据类型

In [7]:
```python
from pyspark.sql.functions import col
import pyspark.sql.types
```

使用 replace_question UDF 用户自定义函数，将 row_df DataFrame 第 4 个字段至最后一个字 段转换为 double。其中，最后一个字段为 label，其余是 feature。

In [8]:
```
df = row_df.select(
    ['url','alchemy_category']+
    [replace_question(col(column)).cast("double").alias(column) for column in row_df.columns[4:]]
)
```

说明: - 用 row_df.select 选取字段

- 选取字段 [‘url’,‘alchemy_category’], 不需要转换
- for column in row_df.columns[4:] 读取第 4 个字段至最后一个字段
- col(column) 读取字段数据并调用 replace_question 自定义函数删除问号 “？”
- .cast(“double”) 转换为 double
- .alias(column) 把别名设置为原来的字段名

查看使用 replace_question UDF 转换后的字段

In [9]:
```
df.printSchema()
```

```
root
 |-- url: string (nullable = true)
 |-- alchemy_category: string (nullable = true)
 |-- alchemy_category_score: double (nullable = true)
 |-- avglinksize: double (nullable = true)
 |-- commonlinkratio_1: double (nullable = true)
 |-- commonlinkratio_2: double (nullable = true)
 |-- commonlinkratio_3: double (nullable = true)
 |-- commonlinkratio_4: double (nullable = true)
 |-- compression_ratio: double (nullable = true)
 |-- embed_ratio: double (nullable = true)
 |-- framebased: double (nullable = true)
 |-- frameTagRatio: double (nullable = true)
 |-- hasDomainLink: double (nullable = true)
 |-- html_ratio: double (nullable = true)
 |-- image_ratio: double (nullable = true)
 |-- is_news: double (nullable = true)
 |-- lengthyLinkDomain: double (nullable = true)
 |-- linkwordscore: double (nullable = true)
 |-- news_front_page: double (nullable = true)
 |-- non_markup_alphanum_characters: double (nullable = true)
 |-- numberOfLinks: double (nullable = true)
 |-- numwords_in_url: double (nullable = true)
 |-- parametrizedLinkRatio: double (nullable = true)
 |-- spelling_errors_ratio: double (nullable = true)
 |-- label: double (nullable = true)
```

In [10]:
```
df.select('url','alchemy_category','alchemy_category_score','is_news','label').show(10)
```

```
+--------------------+------------------+----------------------+-------+-----+
|                 url|  alchemy_category|alchemy_category_score|is_news|label|
+--------------------+------------------+----------------------+-------+-----+
|http://www.bloomb...|          business|              0.789131|    1.0|  0.0|
|http://www.popsci...|        recreation|              0.574147|    1.0|  1.0|
|http://www.menshe...|            health|              0.996526|    1.0|  1.0|
|http://www.dumbli...|            health|              0.801248|    1.0|  1.0|
|http://bleacherre...|            sports|              0.719157|    1.0|  0.0|
|http://www.conven...|                ?|                0.0|    0.0|  0.0|
|http://gofashionl...|arts_entertainment|             0.22111|    1.0|  1.0|
|http://www.inside...|                ?|                0.0|    0.0|  0.0|
|http://www.valetm...|                ?|                0.0|    1.0|  1.0|
|http://www.howswe...|                ?|                0.0|    0.0|  1.0|
+--------------------+------------------+----------------------+-------+-----+
only showing top 10 rows
```

使用 randomSplit 将数据按照 7:3 的比例分成 train_df（训练数据）与 test_df（测试数据），并且.cache() 暂存在内存中，加快后续程序运行的速度。

In [11]:
```
train_df,test_df = df.randomSplit([0.7,0.3])
train_df.cache()
test_df.cache()
```

Out[11]:
```
DataFrame[url: string, alchemy_category: string, alchemy_category_score: double, avglinksize: double, com
monlinkratio_1: double, commonlinkratio_2: double, commonlinkratio_3: double, commonlinkratio_4: doub
le, compression_ratio: double, embed_ratio: double, framebased: double, frameTagRatio: double, hasDomai
nLink: double, html_ratio: double, image_ratio: double, is_news: double, lengthyLinkDomain: double, linkw
ordscore: double, news_front_page: double, non_markup_alphanum_characters: double, numberOfLinks: do
uble, numwords_in_url: double, parametrizedLinkRatio: double, spelling_errors_ratio: double, label: double]
```

1.StringIndexer

In [12]:
```
from pyspark.ml.feature import StringIndexer
```

创建StringIndexer

In [13]:
```
categoryIndexer = StringIndexer(
                inputCol='alchemy_category',
                outputCol='alchemy_category_Index'
)
```

StringINdexer使用fit方法生成"Transformer"

In [14]:
```
categoryTransformer=categoryIndexer.fit(df)
```

查看 categoryTransformer 的内容，categoryTransformer 的 label 属性其实就是网页分类的字 典

In [15]:
```
for i in range(0,len(categoryTransformer.labels)):
    print(str(i)+':'+categoryTransformer.labels[i])
```

```
0:?
1:recreation
2:arts_entertainment
3:business
4:health
5:sports
6:culture_politics
7:computer_internet
8:science_technology
9:gaming
10:religion
11:law_crime
```

```
12:unknown
13:weather
```

使用 categpryTransformer 将所有 df 转换为 df1

In [16]:
```
df1 = categoryTransformer.transform(df)
```

查看转换后的 df1 字段

In [17]:
```
df1.columns
```

Out[17]:
```
['url',
 'alchemy_category',
 'alchemy_category_score',
 'avglinksize',
 'commonlinkratio_1',
 'commonlinkratio_2',
 'commonlinkratio_3',
 'commonlinkratio_4',
 'compression_ratio',
 'embed_ratio',
 'framebased',
 'frameTagRatio',
 'hasDomainLink',
 'html_ratio',
 'image_ratio',
 'is_news',
 'lengthyLinkDomain',
 'linkwordscore',
 'news_front_page',
 'non_markup_alphanum_characters',
 'numberOfLinks',
 'numwords_in_url',
 'parametrizedLinkRatio',
 'spelling_errors_ratio',
 'label',
 'alchemy_category_Index']
```

查看转换后的结果

In [18]:
```
df1.select('alchemy_category','alchemy_category_Index').show(10)
```

```
+------------------+----------------------+
|  alchemy_category|alchemy_category_Index|
+------------------+----------------------+
|          business|                   3.0|
|        recreation|                   1.0|
|            health|                   4.0|
|            health|                   4.0|
|            sports|                   5.0|
|                 ?|                   0.0|
|arts_entertainment|                   2.0|
|                 ?|                   0.0|
|                 ?|                   0.0|
|                 ?|                   0.0|
+------------------+----------------------+
only showing top 10 rows
```

2.OneHotEncoder

导入onehotencoder模块

In [19]:
```
from pyspark.ml.feature import OneHotEncoder
```

创建 OneHotEncoder

In [20]:
```
encoder = OneHotEncoder(dropLast=False,
                        inputCol='alchemy_category_Index',
                        outputCol="alchemy_category_IndexVec")
```

OneHotEncoder 使用 transform 转换，结果是 df2，我们可以使用下列指令查看字段

In [21]:
```
df2=encoder.transform(df1)
df2.columns
```

Out[21]:
```
['url',
 'alchemy_category',
 'alchemy_category_score',
 'avglinksize',
 'commonlinkratio_1',
 'commonlinkratio_2',
 'commonlinkratio_3',
 'commonlinkratio_4',
 'compression_ratio',
 'embed_ratio',
 'framebased',
 'frameTagRatio',
 'hasDomainLink',
 'html_ratio',
 'image_ratio',
 'is_news',
 'lengthyLinkDomain',
 'linkwordscore',
 'news_front_page',
 'non_markup_alphanum_characters',
 'numberOfLinks',
 'numwords_in_url',
 'parametrizedLinkRatio',
 'spelling_errors_ratio',
 'label',
 'alchemy_category_Index',
 'alchemy_category_IndexVec']
```

结果显示新增了 alchemy_category_IndexVec 字段

查看转换后新增的字段

In [22]:
```
df2.select("alchemy_category","alchemy_category_Index",
           "alchemy_category_IndexVec").show(10)
```

```
+----------------+----------------------+-------------------------+
|alchemy_category|alchemy_category_Index|alchemy_category_IndexVec|
+----------------+----------------------+-------------------------+
|        business|                   3.0|           (14,[3],[1.0])|
|       recreation|                1.0|         (14,[1],[1.0])|
|           health|                4.0|         (14,[4],[1.0])|
|           health|                4.0|         (14,[4],[1.0])|
|           sports|                5.0|         (14,[5],[1.0])|
|                ?|                0.0|         (14,[0],[1.0])|
|arts_entertainment|               2.0|         (14,[2],[1.0])|
|                ?|                0.0|         (14,[0],[1.0])|
|                ?|                0.0|         (14,[0],[1.0])|
|                ?|                0.0|         (14,[0],[1.0])|
+------------------+--------------------+-------------------------+
only showing top 10 rows
```

3.VectorAssembler

VectorAssembler可以将多个特征字段整合成一个特征的 Vector

In [23]:
```
from pyspark.ml.feature import VectorAssembler
```

创建全部特征字段 List

In [24]:
```
assemblerInputs = ['alchemy_category_IndexVec']+row_df.columns[4:-1]
assemblerInputs
```

Out[24]:
```
['alchemy_category_IndexVec',
 'alchemy_category_score',
 'avglinksize',
 'commonlinkratio_1',
 'commonlinkratio_2',
 'commonlinkratio_3',
 'commonlinkratio_4',
 'compression_ratio',
 'embed_ratio',
 'framebased',
 'frameTagRatio',
 'hasDomainLink',
 'html_ratio',
 'image_ratio',
 'is_news',
 'lengthyLinkDomain',
 'linkwordscore',
 'news_front_page',
 'non_markup_alphanum_characters',
 'numberOfLinks',
 'numwords_in_url',
 'parametrizedLinkRatio',
 'spelling_errors_ratio']
```

创建 VectorAssembler

In [25]:
```
assembler = VectorAssembler(inputCols=assemblerInputs,
                            outputCol="features")
```

运行 VectorAssembler 转换

In [26]:
```
df3=assembler.transform(df2)
```

查看整合后的新增字段

In [27]:
```
df3.columns
```

Out[27]:
```
['url',
 'alchemy_category',
 'alchemy_category_score',
 'avglinksize',
 'commonlinkratio_1',
 'commonlinkratio_2',
 'commonlinkratio_3',
 'commonlinkratio_4',
 'compression_ratio',
 'embed_ratio',
 'framebased',
 'frameTagRatio',
 'hasDomainLink',
 'html_ratio',
 'image_ratio',
 'is_news',
 'lengthyLinkDomain',
 'linkwordscore',
 'news_front_page',
 'non_markup_alphanum_characters',
 'numberOfLinks',
 'numwords_in_url',
 'parametrizedLinkRatio',
 'spelling_errors_ratio',
 'label',
 'alchemy_category_Index',
 'alchemy_category_IndexVec',
 'features']
```

In [28]:
```
df3.select('features').take(1)
```

Out[28]:
```
[Row(features=SparseVector(36, {3: 1.0, 14: 0.7891, 15: 2.0556, 16: 0.6765, 17: 0.2059, 18: 0.0471, 19: 0.0
235, 20: 0.4438, 23: 0.0908, 25: 0.2458, 26: 0.0039, 27: 1.0, 28: 1.0, 29: 24.0, 31: 5424.0, 32: 170.0, 33: 8.0,
 34: 0.1529, 35: 0.0791}))]
```

4.DecisionTreeClassifier

In [29]:
```
from pyspark.ml.classification import DecisionTreeClassifier
```

运行 DesionTreeClassifier

In [30]:
```
dt = DecisionTreeClassifier(labelCol="label",featuresCol="features",
                            impurity="gini",maxDepth=10,maxBins=14)
```

进行训练，之前创建的 dt 决策树分类，我们可以使用.fit() 方法进行训练，训练结果产生 dt_model 模型，之后可以使用 print 查看产生的模型

In [31]:
```
dt_model=dt.fit(df3)
print(dt_model)
```

```
DecisionTreeClassificationModel (uid=DecisionTreeClassifier_2a7f3eec7817) of depth 10 with 653 nodes
```

进行预测，建立模型后就可以使用.trainform() 进行转换了，转换后会产生预测结果 df4/

In [32]:
```
df4=dt_model.transform(df3)
```

5 建立机器学习 Pipeline 流程

In [33]:
```
from pyspark.ml import Pipeline
from pyspark.ml.feature import StringIndexer, OneHotEncoder,VectorAssembler
from pyspark.ml.classification import DecisionTreeClassifier
```

建立 pipeline

In [34]:
```
stringIndexer = StringIndexer(inputCol='alchemy_category',
outputCol="alchemy_category_Index")
encoder = OneHotEncoder(dropLast=False,
inputCol='alchemy_category_Index',
outputCol="alchemy_category_IndexVec")
assemblerInputs =['alchemy_category_IndexVec'] + row_df.columns[4:-1]
assembler = VectorAssembler(inputCols=assemblerInputs, outputCol="features")
dt = DecisionTreeClassifier(labelCol="label",featuresCol="features",impurity="gini",
                    maxDepth=10, maxBins=14)

pipeline = Pipeline(stages=[stringIndexer,encoder ,assembler,dt ])
```

In [35]:
```
pipeline.getStages()
```

Out[35]:
```
[StringIndexer_bd31abc032f4,
 OneHotEncoder_97883791e25c,
 VectorAssembler_96032e56ba91,
 DecisionTreeClassifier_07bfaed33184]
```

6.使用 pipeline 进行数据处理与训练

In [36]:
```
pipelineModel = pipeline.fit(train_df)
```

看训练完成后的决策树模型

In [37]:
```
pipelineModel.stages[3]
```

Out[37]:
```
DecisionTreeClassificationModel (uid=DecisionTreeClassifier_07bfaed33184) of depth 10 with 615 nodes
```

查看训练完后的决策树模型规则

In [38]:
```
print (pipelineModel.stages[3].toDebugString[:1000])
```

```
DecisionTreeClassificationModel (uid=DecisionTreeClassifier_07bfaed33184) of depth 10 with 615 nodes
  If (feature 31 <= 1765.0)
   If (feature 2 in {1.0})
    If (feature 26 <= 0.201427045)
     If (feature 20 <= 0.484175714)
      If (feature 29 <= 21.5)
       Predict: 0.0
     Else (feature 29 > 21.5)
      If (feature 34 <= 0.049220706)
       If (feature 15 <= 3.723642426)
        If (feature 15 <= 1.0132735585)
         Predict: 1.0
        Else (feature 15 > 1.0132735585)
         If (feature 20 <= 0.4488383145)
          Predict: 0.0
         Else (feature 20 > 0.4488383145)
          Predict: 1.0
       Else (feature 15 > 3.723642426)
        Predict: 1.0
      Else (feature 34 > 0.049220706)
       If (feature 14 <= 0.8503704999999999)
        If (feature 29 <= 65.5)
         Predict: 1.0
        Else (feature 29 > 65.5)
         Predict: 0.0
       Else (feature 14 > 0.8503704999999999)
        Predict: 0.0
    Else (feature 20 > 0.484175714)
     If (
```

7.使用 pipelineModel 进行预测

使用 pipelineModel 的 transform 方法，传入 test_df 测试数据进行预测

In [39]:
```
predicted=pipelineModel.transform(test_df)
```

查看预测后的 Schema，发现新增了 3 个字段

In [40]:
```
predicted.columns
```

Out[40]:
```
['url',
 'alchemy_category',
 'alchemy_category_score',
 'avglinksize',
 'commonlinkratio_1',
 'commonlinkratio_2',
 'commonlinkratio_3',
 'commonlinkratio_4',
 'compression_ratio',
 'embed_ratio',
 'framebased',
 'frameTagRatio',
 'hasDomainLink',
 'html_ratio',
 'image_ratio',
 'is_news',
 'lengthyLinkDomain',
 'linkwordscore',
 'news_front_page',
 'non_markup_alphanum_characters',
 'numberOfLinks',
 'numwords_in_url',
 'parametrizedLinkRatio',
 'spelling_errors_ratio',
 'label',
 'alchemy_category_Index',
 'alchemy_category_IndexVec',
 'features',
 'rawPrediction',
 'probability',
 'prediction']
```

看预测结果 DataFrame

In [41]:
```
predicted.select('url','features','rawprediction','probability','label','prediction').show(10)
```

```
+--------------------+--------------------+-------------+--------------------+-----+----------+
|                 url|            features|rawprediction|         probability|label|prediction|
+--------------------+--------------------+-------------+--------------------+-----+----------+
|http://1000awesom...|(36,[0,15,16,17,1...| [34.0,163.0]|[0.17258883248730...|  1.0|       1.0|
|http://100miledie...|(36,[0,15,20,23,2...|  [1.0,13.0]|[0.07142857142857...|  0.0|       1.0|
|http://13gb.com/v...|(36,[0,15,16,20,2...|  [56.0,30.0]|[0.65116279069767...|  0.0|       0.0|
|http://3kidsandus...|(36,[0,15,16,17,1...| [34.0,163.0]|[0.17258883248730...|  0.0|       1.0|
|http://3kidsandus...|(36,[0,15,16,17,1...| [34.0,163.0]|[0.17258883248730...|  1.0|       1.0|
|http://3kidsandus...|(36,[0,15,16,17,1...|  [55.0,29.0]|[0.65476190476190...|  0.0|       0.0|
|http://8tracks.co...|(36,[2,14,15,16,1...|    [0.0,3.0]|           [0.0,1.0]|  0.0|       1.0|
|http://9gag.com/g...|(36,[0,15,16,17,1...|   [2.0,11.0]|[0.15384615384615...|  0.0|       1.0|
|http://abeautiful...|(36,[2,14,15,16,1...|   [29.0,3.0]|  [0.90625,0.09375]|  1.0|       0.0|
|http://addapinch....|(36,[1,14,15,16,1...|  [3.0,105.0]|[0.02777777777777...|  0.0|       1.0|
+--------------------+--------------------+-------------+--------------------+-----+----------+
only showing top 10 rows
```

查看预测结果与概率

In [42]:
```
predicted.select('probability','prediction') .take(10)
```

Out[42]:
```
[Row(probability=DenseVector([0.1726, 0.8274]), prediction=1.0),
 Row(probability=DenseVector([0.0714, 0.9286]), prediction=1.0),
 Row(probability=DenseVector([0.6512, 0.3488]), prediction=0.0),
 Row(probability=DenseVector([0.1726, 0.8274]), prediction=1.0),
 Row(probability=DenseVector([0.1726, 0.8274]), prediction=1.0),
 Row(probability=DenseVector([0.6548, 0.3452]), prediction=0.0),
 Row(probability=DenseVector([0.0, 1.0]), prediction=1.0),
 Row(probability=DenseVector([0.1538, 0.8462]), prediction=1.0),
 Row(probability=DenseVector([0.9062, 0.0938]), prediction=0.0),
 Row(probability=DenseVector([0.0278, 0.9722]), prediction=1.0)]
```

8.评估模型的准确率

首先从 pyspark.ml.evaluation 导入 BinaryClassificationEvaluator 模块

In [43]:
```
from pyspark.ml.evaluation import BinaryClassificationEvaluator
```

创建 BinaryClassificationEvaluator，传入下列参数：- rawPredictionCol="rawPrediction" 之 前预测后产生的字段

- labelCol="label" 标签字段

- metricName="areaUnderROC" 也就是 AUC

In [44]:
```
evaluator = BinaryClassificationEvaluator(rawPredictionCol="rawPrediction",
                                          labelCol="label",
                                          metricName="areaUnderROC" )
```

计算AUC

In [45]:
```
predictions =pipelineModel.transform(test_df)
auc= evaluator.evaluate(predictions)
auc
```

Out[45]:
```
0.6324079757381753
```

使用 TrainValidation 进行训练验证找出最佳模型

从 pyspark.ml.tuning 导入 ParamGridBuilder 与 TrainValidationSplit 模块

In [46]:
```
from pyspark.ml.tuning import ParamGridBuilder,TrainValidationSplit
```

设置训练验证的参数，我们使用 ParamGridBuilder 设置 impurity 两个参数值、maxDepth 三个 参数值与 maxBins 三个参数值，后续执行训练验证时会执行 233=18 次。

In [47]:
```
paramGrid = ParamGridBuilder()\
.addGrid(dt.impurity, [ "gini","entropy"])\
.addGrid(dt.maxDepth, [ 5,10,15])\
.addGrid(dt.maxBins, [10, 15,20])\
.build()
```

创建 TrainValidationSplit，传入下列参数，执行后创建 tvs 变量：

- estimator=dt，之前创建的 DecisionTreeClassifier
- evaluator=evaluator，之前创建的 BinaryClassificationEvaluator
- estimatorParamMaps=paramGrid，之前创建的 ParamGridBuilder
- trainRatio=0.8，训练验证前会先将数据按照 8:2 的比例分成训练数据与验证数据

In [48]:
```
tvs = TrainValidationSplit(estimator=dt,evaluator=evaluator,
                           estimatorParamMaps=paramGrid,trainRatio=0.8)
```

建立 tvs_pipeline/

In [49]:
```
tvs_pipeline = Pipeline(stages=[stringIndexer,encoder ,assembler, tvs])
```

使用 tvs_pipeline 流程进行训练验证

In [50]:
```
tvs_pipelineModel =tvs_pipeline.fit(train_df)
```

查看训练完成的最佳模型

In [51]:
```
bestModel=tvs_pipelineModel.stages[3].bestModel
bestModel
```

Out[51]:

```
DecisionTreeClassificationModel (uid=DecisionTreeClassifier_07bfaed33184) of depth 15 with 1255 nodes
```

看训练验证完成的最佳模型规则，[: 500] 表示只显示前 500 文字

In [52]:
```
print (bestModel.toDebugString[:500])
```

```
DecisionTreeClassificationModel (uid=DecisionTreeClassifier_07bfaed33184) of depth 15 with 1255 nodes
  If (feature 31 <= 1548.0)
   If (feature 2 in {1.0})
    If (feature 33 <= 11.5)
     If (feature 33 <= 5.5)
      If (feature 35 <= 0.128495221)
       If (feature 28 <= 0.5)
        If (feature 16 <= 0.5052088985000001)
         If (feature 17 <= 0.103194726)
          If (feature 26 <= 0.6825757575)
           If (feature 34 <= 0.179858142)
            If (feature 29 <= 22.5)
             I
```

评估最佳模型 AUC

In [53]:
```
predictions = tvs_pipelineModel.transform(test_df)
auc= evaluator.evaluate(predictions)
auc
```

Out[53]:
```
0.6440101246327793
```

9.使用 crossValidation 交叉验证找出最佳模型

In [54]:
```
from pyspark.ml.tuning import CrossValidator
```

建立交叉验证的 CrossValidator

In [55]:
```
cv = CrossValidator(estimator=dt, evaluator=evaluator,
                    estimatorParamMaps=paramGrid, numFolds=3)
```

建立交叉验证的 cv_pipeline

In [56]:
```
 cv_pipeline = Pipeline(stages=[stringIndexer,encoder ,assembler, cv])
```

用 cv_pipeline 流程进行交叉验证

In [57]:
```
cv_pipelineModel = cv_pipeline.fit(train_df)
```

查看交叉验证完成的最佳模型

In [58]:
```
bestModel=cv_pipelineModel.stages[3].bestModel
bestModel
```

Out[58]:
```
DecisionTreeClassificationModel (uid=DecisionTreeClassifier_07bfaed33184) of depth 15 with 1575 nodes
```

评估最佳模型 AUC

In [59]:
```
predictions = cv_pipelineModel.transform(test_df)
auc= evaluator.evaluate(predictions)
auc
```

Out[59]:
```
0.6608278234025085
```

使用随机森林 **RandomForestClassifier** 分类器

创建 RandomForestClassifier 变量 rf，传入参数与决策树类似，只是多了 numTrees 参数（设置 决策森林中有多少决策树，这里设为 10）

In [60]:
```
from pyspark.ml.classification import RandomForestClassifier
rf =RandomForestClassifier(labelCol="label",featuresCol="features",numTrees=10)
rfpipeline = Pipeline(stages=[stringIndexer,encoder ,assembler,rf ])
```

评估 RandomForestClassifier 的准确度

rfpipeline.fit 传入 train_df 进行训练，再用 rftvs_pipelineModel.transform 传入 test_df 进行评 估，我们可以看到 AUC 约为 0.72，比之前使用决策树的准确度明显增加

In [61]:
```
rfpipelineModel = rfpipeline.fit(train_df)
rfpredicted=rfpipelineModel.transform(test_df)
evaluator.evaluate(rfpredicted)
```

Out[61]:
```
0.7371290346808376
```

使用 RandomForestClassifier TrainValidation 找出最佳模型

In [62]:
```
from pyspark.ml.tuning import ParamGridBuilder, TrainValidationSplit
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.ml.classification import RandomForestClassifier
paramGrid = ParamGridBuilder()\
.addGrid(rf.impurity, [ "gini","entropy"])\
.addGrid(rf.maxDepth, [ 5,10,15])\
.addGrid(rf.maxBins, [10, 15,20])\
.addGrid(rf.numTrees, [10, 20,30])\
.build()
rftvs = TrainValidationSplit(estimator=rf, evaluator=evaluator,
estimatorParamMaps=paramGrid, trainRatio=0.8)
rftvs_pipeline = Pipeline(stages=[stringIndexer,encoder ,assembler, rftvs])
rftvs_pipelineModel =rftvs_pipeline.fit(train_df)
rftvspredictions = rftvs_pipelineModel.transform(test_df)
auc= evaluator.evaluate(rftvspredictions)
auc
```

Out[62]:
```
0.7653783831147928
```

使用 crossValidation 找出最佳模型

In [63]:
```
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
from pyspark.ml import Pipeline
rfcv = CrossValidator(estimator=rf, evaluator=evaluator,
estimatorParamMaps=paramGrid, numFolds=3)
```

```
rfcv_pipeline = Pipeline(stages=[stringIndexer,encoder ,assembler, rfcv])
rfcv_pipelineModel = rfcv_pipeline.fit(train_df)
```

使用最佳模型进行预测

In [64]:
```
rfcvpredictions = rfcv_pipelineModel.transform(test_df)
```

显示使用最佳模型进行预测结果

In [67]:
```
DescDict = {
0: "暂时性网页 (ephemeral)",
1: "长青网页 (evergreen)"
}
for data in rfcvpredictions .select('url','prediction').take(5):
    print(" 网址：  " +str(data[0])+"\n" +\
    " ==> 预测:"+ str(data[1])+ \
    " 说明:"+DescDict[data[1]] +"\n")
```

```
 网址：  http://1000awesomethings.com/2008/07/07/989-blowing-your-nose-in-the-shower/
==> 预测:1.0 说明:长青网页 (evergreen)

 网址：  http://100milediet.org/
==> 预测:0.0 说明:暂时性网页 (ephemeral)

 网址：  http://13gb.com/videos/2298/
==> 预测:0.0 说明:暂时性网页 (ephemeral)

 网址：  http://3kidsandus.com/2010/patriotic-checkerboard-cake-for-the-4th-of-july/
==> 预测:1.0 说明:长青网页 (evergreen)

 网址：  http://3kidsandus.com/2011/cook-for-the-cure-strawberries-n-cream-layer-cake/
==> 预测:1.0 说明:长青网页 (evergreen)
```

计算最佳模型 AUC

In [68]:
```
auc= evaluator.evaluate(rfcvpredictions)
auc
```

Out[68]:
```
0.7632262080194218
```