In [1]:
```python
from pyspark.sql import SparkSession
import pyspark.sql.functions as F
from pyspark.sql.types import *

spark = SparkSession\
    .builder\
    .appName("chapter-25-ML-preprocessing")\
    .getOrCreate()

import os
SPARK_BOOK_DATA_PATH = os.environ['SPARK_BOOK_DATA_PATH']
```

In [41]:
```python
from IPython.display import display
```

In [2]:
```python
sales = spark.read.format("csv")\
  .option("header", "true")\
  .option("inferSchema", "true")\
  .load(SPARK_BOOK_DATA_PATH + "/data/retail-data/by-day/*.csv")\
  .coalesce(5)\
  .where("Description IS NOT NULL")
```

In [3]:
```python
sales.show(3, False)
```

```
+---------+---------+-------------------------------+--------+-------------------+---------+-------
---+--------------+
|InvoiceNo|StockCode|Description                    |Quantity|InvoiceDate        |UnitPrice|Custome
rID|Country       |
+---------+---------+-------------------------------+--------+-------------------+---------+-------
---+--------------+
|580538   |23084    |RABBIT NIGHT LIGHT             |48      |2011-12-05 08:38:00|1.79     |14075.0
|United Kingdom|
|580538   |23077    |DOUGHNUT LIP GLOSS             |20      |2011-12-05 08:38:00|1.25     |14075.0
|United Kingdom|
|580538   |22906    |12 MESSAGE CARDS WITH ENVELOPES|24      |2011-12-05 08:38:00|1.65     |14075.0
|United Kingdom|
+---------+---------+-------------------------------+--------+-------------------+---------+-------
---+--------------+
only showing top 3 rows
```

In [4]:
```
sales.count()
```
Out[4]:
```
540455
```
In [5]:
```
fakeIntDF = spark.read.parquet(SPARK_BOOK_DATA_PATH + "/data/simple-ml-integers")
```
In [6]:
```
fakeIntDF.show(5, False)
```
```
+----+----+----+
|int1|int2|int3|
+----+----+----+
|4   |5   |6   |
|1   |2   |3   |
|7   |8   |9   |
+----+----+----+
```
In [7]:
```
simpleDF = spark.read.json(SPARK_BOOK_DATA_PATH + "/data/simple-ml")
```
In [8]:
```
simpleDF.show(5, False)
```
```
+-----+----+------+------------------+
|color|lab |value1|value2            |
+-----+----+------+------------------+
|green|good|1     |14.386294994851129|
|blue |bad |8     |14.386294994851129|
|blue |bad |12    |14.386294994851129|
|green|good|15    |38.97187133755819 |
|green|good|12    |14.386294994851129|
+-----+----+------+------------------+
only showing top 5 rows
```
In [9]:
```
scaleDF = spark.read.parquet(SPARK_BOOK_DATA_PATH + "/data/simple-ml-scaling")
```

In [10]:
```
scaleDF.show(5, False)
```

```
+---+--------------+
|id |features      |
+---+--------------+
|0  |[1.0,0.1,-1.0]|
|1  |[2.0,1.1,1.0] |
|0  |[1.0,0.1,-1.0]|
|1  |[2.0,1.1,1.0] |
|1  |[3.0,10.1,3.0]|
+---+--------------+
```

In [11]:
```
# COMMAND ----------

from pyspark.ml.feature import RFormula

supervised = RFormula(formula="lab ~ . + color:value1 + color:value2")
supervised.fit(simpleDF).transform(simpleDF).show(5, False)
```

```
+-----+----+------+------------------+--------------------------------------------------------------------
---------+-----+
|color|lab |value1|value2            |features
|label|
+-----+----+------+------------------+--------------------------------------------------------------------
---------+-----+
|green|good|1     |14.386294994851129|(10,[1,2,3,5,8],[1.0,1.0,14.386294994851129,1.0,14.3862949948
51129])  |1.0  |
|blue |bad |8     |14.386294994851129|(10,[2,3,6,9],[8.0,14.386294994851129,8.0,14.38629499485112
9])        |0.0  |
|blue |bad |12    |14.386294994851129|(10,[2,3,6,9],[12.0,14.386294994851129,12.0,14.38629499485112
9])      |0.0  |
|green|good|15    |38.97187133755819 |(10,[1,2,3,5,8],[1.0,15.0,38.97187133755819,15.0,38.971871337
55819])  |1.0  |
|green|good|12    |14.386294994851129|(10,[1,2,3,5,8],[1.0,12.0,14.386294994851129,12.0,14.38629499
4851129])|1.0  |
+-----+----+------+------------------+--------------------------------------------------------------------
---------+-----+
only showing top 5 rows
```

In [12]:
```python
# SQLTransformer - like spark.sql()

from pyspark.ml.feature import SQLTransformer

basicTransformation = SQLTransformer()\
  .setStatement("""
    SELECT sum(Quantity), count(*), CustomerID
    FROM __THIS__
    GROUP BY CustomerID
  """)

basicTransformation.transform(sales).show(5, False)
```

```
+-------------+--------+----------+
|sum(Quantity)|count(1)|CustomerID|
+-------------+--------+----------+
|119          |62      |14452.0   |
|440          |143     |16916.0   |
|630          |72      |17633.0   |
|34           |6       |14768.0   |
|1542         |30      |13094.0   |
+-------------+--------+----------+
only showing top 5 rows
```

In [13]:
```python
sales.createOrReplaceTempView("sales")
```

In [14]:
```
spark.sql("select sum(Quantity), count(*), CustomerID from sales group by CustomerID").show(5,Fa
```

```
+-------------+--------+----------+
|sum(Quantity)|count(1)|CustomerID|
+-------------+--------+----------+
|119          |62      |14452.0   |
|440          |143     |16916.0   |
|630          |72      |17633.0   |
|34           |6       |14768.0   |
|1542         |30      |13094.0   |
+-------------+--------+----------+
only showing top 5 rows
```

In [15]:
```
# VectorAssembler - transformer to assemble columns into vector

from pyspark.ml.feature import VectorAssembler
va = VectorAssembler().setInputCols(["int1", "int2", "int3"])
va.transform(fakeIntDF).show(5, False)
```

```
+----+----+----+------------------------------------+
|int1|int2|int3|VectorAssembler_91f1042ca8cb__output|
+----+----+----+------------------------------------+
|4   |5   |6   |[4.0,5.0,6.0]                       |
|1   |2   |3   |[1.0,2.0,3.0]                       |
|7   |8   |9   |[7.0,8.0,9.0]                       |
+----+----+----+------------------------------------+
```

In [ ]:
```

```

In [16]:
```python
# COMMAND ----------

contDF = spark.range(20).selectExpr("cast(id as double)")
contDF.show(5, False)
```

```
+---+
|id |
+---+
|0.0|
|1.0|
|2.0|
|3.0|
|4.0|
+---+
only showing top 5 rows
```

In [17]:
```python
# Bucketizer - transformer to split data into buckets

from pyspark.ml.feature import Bucketizer
bucketBorders = [-1.0, 5.0, 10.0, 250.0, 600.0]
bucketer = Bucketizer().setSplits(bucketBorders).setInputCol("id")
bucketer.transform(contDF).show(truncate=False)
```

```
+----+-----------------------------+
|id  |Bucketizer_bffc8656d38c__output|
+----+-----------------------------+
|0.0 |0.0                          |
|1.0 |0.0                          |
|2.0 |0.0                          |
|3.0 |0.0                          |
|4.0 |0.0                          |
|5.0 |1.0                          |
|6.0 |1.0                          |
|7.0 |1.0                          |
|8.0 |1.0                          |
|9.0 |1.0                          |
|10.0|2.0                          |
|11.0|2.0                          |
|12.0|2.0                          |
|13.0|2.0                          |
|14.0|2.0                          |
|15.0|2.0                          |
|16.0|2.0                          |
|17.0|2.0                          |
|18.0|2.0                          |
|19.0|2.0                          |
+----+-----------------------------+
```

In [18]:
```python
# QuantileDiscretizer - transformer to partition data by Percentile
from pyspark.ml.feature import QuantileDiscretizer
bucketer = QuantileDiscretizer().setInputCol("id").setNumBuckets(5)
```

In [19]:
```
bucketer.fit(contDF).transform(contDF).show(truncate=False)
```

```
+----+---------------------------------------+
|id  |QuantileDiscretizer_4f6bd0bcc5c0__output|
+----+---------------------------------------+
|0.0 |0.0                                    |
|1.0 |0.0                                    |
|2.0 |0.0                                    |
|3.0 |1.0                                    |
|4.0 |1.0                                    |
|5.0 |1.0                                    |
|6.0 |1.0                                    |
|7.0 |2.0                                    |
|8.0 |2.0                                    |
|9.0 |2.0                                    |
|10.0|2.0                                    |
|11.0|3.0                                    |
|12.0|3.0                                    |
|13.0|3.0                                    |
|14.0|3.0                                    |
|15.0|4.0                                    |
|16.0|4.0                                    |
|17.0|4.0                                    |
|18.0|4.0                                    |
|19.0|4.0                                    |
+----+---------------------------------------+
```

In [20]:

```python
# StandardScaler - transformer to normalize data

from pyspark.ml.feature import StandardScaler
sScaler = StandardScaler().setInputCol("features")
sScaler.fit(scaleDF).transform(scaleDF).show(5, False)
```

```
+---+--------------+------------------------------------------------------------+
|id |features      |StandardScaler_3325a6ec77c0__output                         |
+---+--------------+------------------------------------------------------------+
|0  |[1.0,0.1,-1.0]|[1.1952286093343936,0.02337622911060922,-0.5976143046671968]|
|1  |[2.0,1.1,1.0] |[2.390457218668787,0.2571385202167014,0.5976143046671968]   |
|0  |[1.0,0.1,-1.0]|[1.1952286093343936,0.02337622911060922,-0.5976143046671968]|
|1  |[2.0,1.1,1.0] |[2.390457218668787,0.2571385202167014,0.5976143046671968]   |
|1  |[3.0,10.1,3.0]|[3.5856858280031805,2.3609991401715313,1.7928429140015902]  |
+---+--------------+------------------------------------------------------------+
```

In [21]:

```python
# COMMAND ----------

from pyspark.ml.feature import MinMaxScaler
minMax = MinMaxScaler().setMin(5).setMax(10).setInputCol("features")
minMax.fit(scaleDF).transform(scaleDF).show(5, False)
```

```
+---+--------------+-------------------------------+
|id |features      |MinMaxScaler_e07d77613bde__output|
+---+--------------+-------------------------------+
|0  |[1.0,0.1,-1.0]|[5.0,5.0,5.0]                  |
|1  |[2.0,1.1,1.0] |[7.5,5.5,7.5]                  |
|0  |[1.0,0.1,-1.0]|[5.0,5.0,5.0]                  |
|1  |[2.0,1.1,1.0] |[7.5,5.5,7.5]                  |
|1  |[3.0,10.1,3.0]|[10.0,10.0,10.0]               |
+---+--------------+-------------------------------+
```

In [22]:

```python
# COMMAND ----------

from pyspark.ml.feature import MaxAbsScaler
maScaler = MaxAbsScaler().setInputCol("features")
maScaler.fit(scaleDF).transform(scaleDF).show(5, False)
```

```
+---+--------------+-------------------------------------------------------------+
|id |features      |MaxAbsScaler_585c91198f51__output                            |
+---+--------------+-------------------------------------------------------------+
|0  |[1.0,0.1,-1.0]|[0.3333333333333333,0.009900990099009903,-0.3333333333333333]|
|1  |[2.0,1.1,1.0] |[0.6666666666666666,0.10891089108910892,0.3333333333333333]  |
|0  |[1.0,0.1,-1.0]|[0.3333333333333333,0.009900990099009903,-0.3333333333333333]|
|1  |[2.0,1.1,1.0] |[0.6666666666666666,0.10891089108910892,0.3333333333333333]  |
|1  |[3.0,10.1,3.0]|[1.0,1.0,1.0]                                                |
+---+--------------+-------------------------------------------------------------+
```

In [23]:

```python
# COMMAND ----------

from pyspark.ml.feature import ElementwiseProduct
from pyspark.ml.linalg import Vectors
scaleUpVec = Vectors.dense(10.0, 15.0, 20.0)
scalingUp = ElementwiseProduct()\
  .setScalingVec(scaleUpVec)\
  .setInputCol("features")
scalingUp.transform(scaleDF).show(5, False)
```

```
+---+--------------+-----------------------------------------+
|id |features      |ElementwiseProduct_8d94b9364b23__output|
+---+--------------+-----------------------------------------+
|0  |[1.0,0.1,-1.0]|[10.0,1.5,-20.0]                         |
|1  |[2.0,1.1,1.0] |[20.0,16.5,20.0]                         |
|0  |[1.0,0.1,-1.0]|[10.0,1.5,-20.0]                         |
|1  |[2.0,1.1,1.0] |[20.0,16.5,20.0]                         |
|1  |[3.0,10.1,3.0]|[30.0,151.5,60.0]                        |
+---+--------------+-----------------------------------------+
```

In [24]:

```python
# COMMAND ----------

from pyspark.ml.feature import Normalizer
manhattanDistance = Normalizer().setP(1).setInputCol("features")
manhattanDistance.transform(scaleDF).show(5, False)
```

```
+---+--------------+------------------------------------------------------------------+
|id |features      |Normalizer_5f82750e3449__output                                   |
+---+--------------+------------------------------------------------------------------+
|0  |[1.0,0.1,-1.0]|[0.47619047619047616,0.047619047619047616,-0.47619047619047616]|
|1  |[2.0,1.1,1.0] |[0.48780487804878053,0.26829268292682934,0.24390243902439027]  |
|0  |[1.0,0.1,-1.0]|[0.47619047619047616,0.047619047619047616,-0.47619047619047616]|
|1  |[2.0,1.1,1.0] |[0.48780487804878053,0.26829268292682934,0.24390243902439027]  |
|1  |[3.0,10.1,3.0]|[0.18633540372670807,0.6273291925465838,0.18633540372670807]   |
+---+--------------+------------------------------------------------------------------+
```

In [26]:

```python
# StringIndexer - transformer to convert categorical data into number

from pyspark.ml.feature import StringIndexer
lblIndxr = StringIndexer().setInputCol("lab").setOutputCol("labelInd")
idxRes = lblIndxr.fit(simpleDF).transform(simpleDF)
idxRes.show(5, False)
```

```
+-----+----+------+------------------+--------+
|color|lab |value1|value2            |labelInd|
+-----+----+------+------------------+--------+
|green|good|1     |14.386294994851129|1.0     |
|blue |bad |8     |14.386294994851129|0.0     |
|blue |bad |12    |14.386294994851129|0.0     |
|green|good|15    |38.97187133755819 |1.0     |
|green|good|12    |14.386294994851129|1.0     |
+-----+----+------+------------------+--------+
only showing top 5 rows
```

In [27]:
```python
# IndexToString - transformer to convert number back to category

from pyspark.ml.feature import IndexToString
labelReverse = IndexToString().setInputCol("labelInd")
labelReverse.transform(idxRes).show(5, False)
```

```
+-----+----+------+-----------------+--------+--------------------------------+
|color|lab |value1|value2           |labelInd|IndexToString_1e66c232baf4__output|
+-----+----+------+-----------------+--------+--------------------------------+
|green|good|1     |14.386294994851129|1.0     |good                            |
|blue |bad |8     |14.386294994851129|0.0     |bad                             |
|blue |bad |12    |14.386294994851129|0.0     |bad                             |
|green|good|15    |38.97187133755819 |1.0     |good                            |
|green|good|12    |14.386294994851129|1.0     |good                            |
+-----+----+------+-----------------+--------+--------------------------------+
only showing top 5 rows
```

In [28]:
```python
# COMMAND ----------

valIndexer = StringIndexer().setInputCol("value1").setOutputCol("value1_Ind")
valIndexer.fit(simpleDF).transform(simpleDF).show(5, False)
```

```
+-----+----+------+-----------------+----------+
|color|lab |value1|value2           |value1_Ind|
+-----+----+------+-----------------+----------+
|green|good|1     |14.386294994851129|0.0       |
|blue |bad |8     |14.386294994851129|7.0       |
|blue |bad |12    |14.386294994851129|1.0       |
|green|good|15    |38.97187133755819 |3.0       |
|green|good|12    |14.386294994851129|1.0       |
+-----+----+------+-----------------+----------+
only showing top 5 rows
```

In [29]:

```python
# COMMAND ----------

from pyspark.ml.feature import VectorIndexer
from pyspark.ml.linalg import Vectors
idxIn = spark.createDataFrame([
  (Vectors.dense(1, 2, 3),1),
  (Vectors.dense(2, 5, 6),2),
  (Vectors.dense(1, 8, 9),3)
]).toDF("features", "label")
indxr = VectorIndexer()\
  .setInputCol("features")\
  .setOutputCol("idxed")\
  .setMaxCategories(2)
indxr.fit(idxIn).transform(idxIn).show(5, False)
```

```
+-------------+-----+-------------+
|features     |label|idxed        |
+-------------+-----+-------------+
|[1.0,2.0,3.0]|1    |[0.0,2.0,3.0]|
|[2.0,5.0,6.0]|2    |[1.0,5.0,6.0]|
|[1.0,8.0,9.0]|3    |[0.0,8.0,9.0]|
+-------------+-----+-------------+
```

In [30]:
```python
# COMMAND ----------

from pyspark.ml.feature import OneHotEncoder, StringIndexer
lblIndxr = StringIndexer().setInputCol("color").setOutputCol("colorInd")
colorLab = lblIndxr.fit(simpleDF).transform(simpleDF.select("color"))
ohe = OneHotEncoder().setInputCol("colorInd")
ohe.transform(colorLab).show(5, False)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-30-64f90509c5ec> in <module>
      5 colorLab = lblIndxr.fit(simpleDF).transform(simpleDF.select("color"))
      6 ohe = OneHotEncoder().setInputCol("colorInd")
----> 7 ohe.transform(colorLab).show(5, False)

AttributeError: 'OneHotEncoder' object has no attribute 'transform'
```

In [31]:
```python
# Tokenizer - transformer to turn sentence into word array

from pyspark.ml.feature import Tokenizer
tkn = Tokenizer().setInputCol("Description").setOutputCol("DescOut")
tokenized = tkn.transform(sales.select("Description"))
tokenized.show(20, False)
```

```
+-----------------------------------+-------------------------------------------+
|Description                        |DescOut                                    |
+-----------------------------------+-------------------------------------------+
|RABBIT NIGHT LIGHT                 |[rabbit, night, light]                     |
|DOUGHNUT LIP GLOSS                 |[doughnut, lip, gloss]                     |
|12 MESSAGE CARDS WITH ENVELOPES    |[12, message, cards, with, envelopes]      |
|BLUE HARMONICA IN BOX              |[blue, harmonica, in, box]                 |
|GUMBALL COAT RACK                  |[gumball, coat, rack]                      |
|SKULLS  WATER TRANSFER TATTOOS     |[skulls, , water, transfer, tattoos]       |
|FELTCRAFT GIRL AMELIE KIT          |[feltcraft, girl, amelie, kit]             |
|CAMOUFLAGE LED TORCH               |[camouflage, led, torch]                   |
|WHITE SKULL HOT WATER BOTTLE       |[white, skull, hot, water, bottle]         |
|ENGLISH ROSE HOT WATER BOTTLE      |[english, rose, hot, water, bottle]        |
|HOT WATER BOTTLE KEEP CALM         |[hot, water, bottle, keep, calm]           |
|SCOTTIE DOG HOT WATER BOTTLE       |[scottie, dog, hot, water, bottle]         |
|ROSE CARAVAN DOORSTOP              |[rose, caravan, doorstop]                  |
|GINGHAM HEART  DOORSTOP RED        |[gingham, heart, , doorstop, red]          |
|STORAGE TIN VINTAGE LEAF           |[storage, tin, vintage, leaf]              |
|SET OF 4 KNICK KNACK TINS POPPIES  |[set, of, 4, knick, knack, tins, poppies]  |
|POPCORN HOLDER                     |[popcorn, holder]                          |
|GROW A FLYTRAP OR SUNFLOWER IN TIN |[grow, a, flytrap, or, sunflower, in, tin]|
|AIRLINE BAG VINTAGE WORLD CHAMPION |[airline, bag, vintage, world, champion]   |
|AIRLINE BAG VINTAGE JET SET BROWN  |[airline, bag, vintage, jet, set, brown]   |
+-----------------------------------+-------------------------------------------+
only showing top 20 rows
```

In [33]:
```python
# RegexTokenizer - Tokenizer with RegEx

from pyspark.ml.feature import RegexTokenizer
rt = RegexTokenizer()\
  .setInputCol("Description")\
  .setOutputCol("DescOut")\
  .setPattern(" ")\
  .setToLowercase(True)
rt.transform(sales.select("Description")).show(10, False)
```

```
+--------------------------------+--------------------------------------+
|Description                     |DescOut                               |
+--------------------------------+--------------------------------------+
|RABBIT NIGHT LIGHT              |[rabbit, night, light]                |
|DOUGHNUT LIP GLOSS              |[doughnut, lip, gloss]                |
|12 MESSAGE CARDS WITH ENVELOPES|[12, message, cards, with, envelopes]|
|BLUE HARMONICA IN BOX           |[blue, harmonica, in, box]            |
|GUMBALL COAT RACK               |[gumball, coat, rack]                 |
|SKULLS  WATER TRANSFER TATTOOS  |[skulls, water, transfer, tattoos]    |
|FELTCRAFT GIRL AMELIE KIT       |[feltcraft, girl, amelie, kit]        |
|CAMOUFLAGE LED TORCH            |[camouflage, led, torch]              |
|WHITE SKULL HOT WATER BOTTLE    |[white, skull, hot, water, bottle]    |
|ENGLISH ROSE HOT WATER BOTTLE   |[english, rose, hot, water, bottle]   |
+--------------------------------+--------------------------------------+
only showing top 10 rows
```

In [34]:

```python
# COMMAND ----------

from pyspark.ml.feature import RegexTokenizer
rt = RegexTokenizer()\
  .setInputCol("Description")\
  .setOutputCol("DescOut")\
  .setPattern(" ")\
  .setGaps(False)\
  .setToLowercase(True)
rt.transform(sales.select("Description")).show(10, False)
```

```
+-------------------------------+---------------+
|Description                    |DescOut        |
+-------------------------------+---------------+
|RABBIT NIGHT LIGHT             |[ ,  ]         |
|DOUGHNUT LIP GLOSS             |[ ,  ,  ]      |
|12 MESSAGE CARDS WITH ENVELOPES|[ ,  ,  ,  ]   |
|BLUE HARMONICA IN BOX          |[ ,  ,  ,  ]   |
|GUMBALL COAT RACK              |[ ,  ]         |
|SKULLS  WATER TRANSFER TATTOOS |[ ,  ,  ,  ,  ]|
|FELTCRAFT GIRL AMELIE KIT      |[ ,  ,  ]      |
|CAMOUFLAGE LED TORCH           |[ ,  ]         |
|WHITE SKULL HOT WATER BOTTLE   |[ ,  ,  ,  ,  ]|
|ENGLISH ROSE HOT WATER BOTTLE  |[ ,  ,  ,  ]   |
+-------------------------------+---------------+
only showing top 10 rows
```

In [35]:

```python
# COMMAND ----------

from pyspark.ml.feature import StopWordsRemover
englishStopWords = StopWordsRemover.loadDefaultStopWords("english")
stops = StopWordsRemover()\
  .setStopWords(englishStopWords)\
  .setInputCol("DescOut")
stops.transform(tokenized).show(5, False)
```

```
+-------------------------------+-------------------------------------+-----------------------------
---------+
|Description                    |DescOut                              |StopWordsRemover_352087335d7
d__output|
+-------------------------------+-------------------------------------+-----------------------------
---------+
|RABBIT NIGHT LIGHT             |[rabbit, night, light]               |[rabbit, night, light]
|
|DOUGHNUT LIP GLOSS             |[doughnut, lip, gloss]               |[doughnut, lip, gloss]
|
|12 MESSAGE CARDS WITH ENVELOPES|[12, message, cards, with, envelopes]|[12, message, cards, envelop
es]      |
|BLUE HARMONICA IN BOX          |[blue, harmonica, in, box]           |[blue, harmonica, box]
|
|GUMBALL COAT RACK              |[gumball, coat, rack]                |[gumball, coat, rack]
|
+-------------------------------+-------------------------------------+-----------------------------
---------+
only showing top 5 rows
```

In [36]:
```python
# COMMAND ----------

from pyspark.ml.feature import NGram
unigram = NGram().setInputCol("DescOut").setN(1)
bigram = NGram().setInputCol("DescOut").setN(2)
unigram.transform(tokenized.select("DescOut")).show(5, False)
bigram.transform(tokenized.select("DescOut")).show(5, False)
```

```
+--------------------------------------+--------------------------------------+
|DescOut                               |NGram_89b73e4d33d9__output            |
+--------------------------------------+--------------------------------------+
|[rabbit, night, light]                |[rabbit, night, light]                |
|[doughnut, lip, gloss]                |[doughnut, lip, gloss]                |
|[12, message, cards, with, envelopes]|[12, message, cards, with, envelopes]|
|[blue, harmonica, in, box]            |[blue, harmonica, in, box]            |
|[gumball, coat, rack]                 |[gumball, coat, rack]                 |
+--------------------------------------+--------------------------------------+
only showing top 5 rows

+--------------------------------------+------------------------------------------------------+
|DescOut                               |NGram_09d12ca8503c__output                            |
+--------------------------------------+------------------------------------------------------+
|[rabbit, night, light]                |[rabbit night, night light]                           |
|[doughnut, lip, gloss]                |[doughnut lip, lip gloss]                             |
|[12, message, cards, with, envelopes]|[12 message, message cards, cards with, with envelopes]|
|[blue, harmonica, in, box]            |[blue harmonica, harmonica in, in box]                |
|[gumball, coat, rack]                 |[gumball coat, coat rack]                             |
+--------------------------------------+------------------------------------------------------+
only showing top 5 rows
```

In [37]:
```python
# COMMAND ----------

from pyspark.ml.feature import CountVectorizer
cv = CountVectorizer()\
  .setInputCol("DescOut")\
  .setOutputCol("countVec")\
  .setVocabSize(500)\
  .setMinTF(1)\
  .setMinDF(2)
fittedCV = cv.fit(tokenized)
fittedCV.transform(tokenized).show(5, False)
```

```
+--------------------------------+--------------------------------------+----------------------------
----------+
|Description                     |DescOut                               |countVec
|
+--------------------------------+--------------------------------------+----------------------------
----------+
|RABBIT NIGHT LIGHT              |[rabbit, night, light]                |(500,[150,185,212],[1.0,1.0,
1.0])     |
|DOUGHNUT LIP GLOSS              |[doughnut, lip, gloss]                |(500,[462,463,491],[1.0,1.0,
1.0])     |
|12 MESSAGE CARDS WITH ENVELOPES|[12, message, cards, with, envelopes]|(500,[35,41,166],[1.0,1.0,1.
0])       |
|BLUE HARMONICA IN BOX           |[blue, harmonica, in, box]            |(500,[10,16,36,352],[1.0,1.
0,1.0,1.0])|
|GUMBALL COAT RACK               |[gumball, coat, rack]                 |(500,[228,281,407],[1.0,1.0,
1.0])     |
+--------------------------------+--------------------------------------+----------------------------
----------+
only showing top 5 rows
```

In [38]:
```python
# COMMAND ----------

tfIdfIn = tokenized\
  .where("array_contains(DescOut, 'red')")\
  .select("DescOut")\
  .limit(10)
tfIdfIn.show(10, False)
```

```
+---------------------------------------+
|DescOut                                |
+---------------------------------------+
|[gingham, heart, , doorstop, red]      |
|[red, floral, feltcraft, shoulder, bag]|
|[alarm, clock, bakelike, red]          |
|[pin, cushion, babushka, red]          |
|[red, retrospot, mini, cases]          |
|[red, kitchen, scales]                 |
|[gingham, heart, , doorstop, red]      |
|[large, red, babushka, notebook]       |
|[red, retrospot, oven, glove]          |
|[red, retrospot, plate]                |
+---------------------------------------+
```

In [39]:
```python
# COMMAND ----------

from pyspark.ml.feature import HashingTF, IDF
tf = HashingTF()\
  .setInputCol("DescOut")\
  .setOutputCol("TFOut")\
  .setNumFeatures(10000)
idf = IDF()\
  .setInputCol("TFOut")\
  .setOutputCol("IDFOut")\
  .setMinDocFreq(2)
```

```
In [40]:
# COMMAND ----------

idf.fit(tf.transform(tfIdfIn)).transform(tf.transform(tfIdfIn)).show(10, False)
```

```
+----------------------------------------+-----------------------------------------------------+--
-----------------------------------------------------------------------------------------------
--------------+
|DescOut                                 |TFOut                                                |ID
FOut
|
+----------------------------------------+-----------------------------------------------------+--
-----------------------------------------------------------------------------------------------
--------------+
|[gingham, heart, , doorstop, red]      |(10000,[52,804,3372,6594,9808],[1.0,1.0,1.0,1.0,1.0])|(1
0000,[52,804,3372,6594,9808],[0.0,1.2992829841302609,1.2992829841302609,1.2992829841302609,1.2992
829841302609])|
|[red, floral, feltcraft, shoulder, bag]|(10000,[50,52,415,6756,8005],[1.0,1.0,1.0,1.0,1.0])  |(1
0000,[50,52,415,6756,8005],[0.0,0.0,0.0,0.0,0.0])
|
|[alarm, clock, bakelike, red]          |(10000,[52,4995,8737,9001],[1.0,1.0,1.0,1.0])        |(1
0000,[52,4995,8737,9001],[0.0,0.0,0.0,0.0])
|
|[pin, cushion, babushka, red]          |(10000,[52,610,2490,7153],[1.0,1.0,1.0,1.0])         |(1
0000,[52,610,2490,7153],[0.0,0.0,0.0,1.2992829841302609])
|
|[red, retrospot, mini, cases]          |(10000,[52,547,6703,8448],[1.0,1.0,1.0,1.0])         |(1
0000,[52,547,6703,8448],[0.0,0.0,0.0,1.0116009116784799])
|
|[red, kitchen, scales]                 |(10000,[52,756,6452],[1.0,1.0,1.0])                  |(1
0000,[52,756,6452],[0.0,0.0,0.0])
|
|[gingham, heart, , doorstop, red]      |(10000,[52,804,3372,6594,9808],[1.0,1.0,1.0,1.0,1.0])|(1
0000,[52,804,3372,6594,9808],[0.0,1.2992829841302609,1.2992829841302609,1.2992829841302609,1.2992
829841302609])|
|[large, red, babushka, notebook]       |(10000,[52,2787,7022,7153],[1.0,1.0,1.0,1.0])        |(1
0000,[52,2787,7022,7153],[0.0,0.0,0.0,1.2992829841302609])
|
|[red, retrospot, oven, glove]          |(10000,[52,8242,8448,8667],[1.0,1.0,1.0,1.0])        |(1
0000,[52,8242,8448,8667],[0.0,0.0,1.0116009116784799,0.0])
|
|[red, retrospot, plate]                |(10000,[52,4925,8448],[1.0,1.0,1.0])                 |(1
0000,[52,4925,8448],[0.0,0.0,1.0116009116784799])
```

```
|
+------------------------------------+------------------------------------------------+--
---------------------------------------------------------------------------------------
-------------+
```

In [42]:
```
display(idf.fit(tf.transform(tfIdfIn)).transform(tf.transform(tfIdfIn)).limit(10).toPandas())
```

	DescOut	TFOut	IDFOut
0	[gingham, heart, , doorstop, red]	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
1	[red, floral, feltcraft, shoulder, bag]	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
2	[alarm, clock, bakelike, red]	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
3	[pin, cushion, babushka, red]	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
4	[red, retrospot, mini, cases]	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
5	[red, kitchen, scales]	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
6	[gingham, heart, , doorstop, red]	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
7	[large, red, babushka, notebook]	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
8	[red, retrospot, oven, glove]	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
9	[red, retrospot, plate]	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...

In [43]:
```python
# COMMAND ----------

from pyspark.ml.feature import Word2Vec
# Input data: Each row is a bag of words from a sentence or document.
documentDF = spark.createDataFrame([
    ("Hi I heard about Spark".split(" "), ),
    ("I wish Java could use case classes".split(" "), ),
    ("Logistic regression models are neat".split(" "), )
], ["text"])
# Learn a mapping from words to Vectors.
word2Vec = Word2Vec(vectorSize=3, minCount=0, inputCol="text",
  outputCol="result")
model = word2Vec.fit(documentDF)
result = model.transform(documentDF)
for row in result.collect():
    text, vector = row
    print("Text: [%s] => \nVector: %s\n" % (", ".join(text), str(vector)))
```

```
Text: [Hi, I, heard, about, Spark] =>
Vector: [0.10983876287937165,-0.03447718722745776,0.000940057821571827]

Text: [I, wish, Java, could, use, case, classes] =>
Vector: [0.0072656965681484765,-0.018058971102748598,0.003386378288269043]

Text: [Logistic, regression, models, are, neat] =>
Vector: [-0.047607143968343736,-0.0320490337908268,0.07224417026154697]
```

In [44]:

```python
# COMMAND ----------

from pyspark.ml.feature import PCA
pca = PCA().setInputCol("features").setK(2)
pca.fit(scaleDF).transform(scaleDF).show(20, False)
```

```
+---+--------------+------------------------------------------+
|id |features      |PCA_6f4ed9b6b39d__output                  |
+---+--------------+------------------------------------------+
|0  |[1.0,0.1,-1.0]|[0.0713719499248417,-0.4526654888147805]  |
|1  |[2.0,1.1,1.0] |[-1.6804946984073723,1.2593401322219198]  |
|0  |[1.0,0.1,-1.0]|[0.0713719499248417,-0.4526654888147805]  |
|1  |[2.0,1.1,1.0] |[-1.6804946984073723,1.2593401322219198]  |
|1  |[3.0,10.1,3.0]|[-10.872398139848944,0.030962697060155975]|
+---+--------------+------------------------------------------+
```

In [45]:
```python
# COMMAND ----------

from pyspark.ml.feature import PolynomialExpansion
pe = PolynomialExpansion().setInputCol("features").setDegree(2)
pe.transform(scaleDF).show(5, False)
```

```
+---+--------------+------------------------------------------------------------------------
----+
|id |features      |PolynomialExpansion_70c2c7232672__output
|
+---+--------------+------------------------------------------------------------------------
----+
|0  |[1.0,0.1,-1.0]|[1.0,1.0,0.1,0.1,0.010000000000000002,-1.0,-1.0,-0.1,1.0]
|
|1  |[2.0,1.1,1.0] |[2.0,4.0,1.1,2.2,1.2100000000000002,1.0,2.0,1.1,1.0]
|
|0  |[1.0,0.1,-1.0]|[1.0,1.0,0.1,0.1,0.010000000000000002,-1.0,-1.0,-0.1,1.0]
|
|1  |[2.0,1.1,1.0] |[2.0,4.0,1.1,2.2,1.2100000000000002,1.0,2.0,1.1,1.0]
|
|1  |[3.0,10.1,3.0]|[3.0,9.0,10.1,30.299999999999997,102.00999999999999,3.0,9.0,30.299999999999997,
9.0]|
+---+--------------+------------------------------------------------------------------------
----+
```

In [46]:

```python
# COMMAND ----------

from pyspark.ml.feature import ChiSqSelector, Tokenizer
tkn = Tokenizer().setInputCol("Description").setOutputCol("DescOut")
tokenized = tkn\
  .transform(sales.select("Description", "CustomerId"))\
  .where("CustomerId IS NOT NULL")
prechi = fittedCV.transform(tokenized)\
  .where("CustomerId IS NOT NULL")
chisq = ChiSqSelector()\
  .setFeaturesCol("countVec")\
  .setLabelCol("CustomerId")\
  .setNumTopFeatures(2)
chisq.fit(prechi).transform(prechi)\
  .drop("customerId", "Description", "DescOut").show(5, False)
```

```
+---------------------------------------+-----------------------------------+
|countVec                               |ChiSqSelector_9631cfd8c94f__output|
+---------------------------------------+-----------------------------------+
|(500,[150,185,212],[1.0,1.0,1.0])      |(2,[],[])                          |
|(500,[462,463,491],[1.0,1.0,1.0])      |(2,[],[])                          |
|(500,[35,41,166],[1.0,1.0,1.0])        |(2,[],[])                          |
|(500,[10,16,36,352],[1.0,1.0,1.0,1.0])|(2,[],[])                          |
|(500,[228,281,407],[1.0,1.0,1.0])      |(2,[],[])                          |
+---------------------------------------+-----------------------------------+
only showing top 5 rows
```

In [47]:

```python
# COMMAND ----------

fittedPCA = pca.fit(scaleDF)
fittedPCA.write().overwrite().save("/tmp/fittedPCA")
```

In [48]:
```python
# COMMAND ----------

from pyspark.ml.feature import PCAModel
loadedPCA = PCAModel.load("/tmp/fittedPCA")
loadedPCA.transform(scaleDF).show(5, False)


# COMMAND ----------
```

```
+---+--------------+------------------------------------------+
|id |features      |PCAModel_5f0cec389764__output             |
+---+--------------+------------------------------------------+
|0  |[1.0,0.1,-1.0]|[0.0713719499248417,-0.4526654888147805]  |
|1  |[2.0,1.1,1.0] |[-1.6804946984073723,1.2593401322219198]  |
|0  |[1.0,0.1,-1.0]|[0.0713719499248417,-0.4526654888147805]  |
|1  |[2.0,1.1,1.0] |[-1.6804946984073723,1.2593401322219198]  |
|1  |[3.0,10.1,3.0]|[-10.872398139848944,0.030962697060155975]|
+---+--------------+------------------------------------------+
```

In [49]:
```python
spark.stop()
```

In [ ]:
```python

```