Statlog DATASET

already preprocessed the dataset.

In [13]:
```
from pyspark.ml.feature import HashingTF, IDF, Tokenizer
from pyspark.sql import SparkSession
from pyspark.sql.functions import monotonically_increasing_id
from pyspark.sql import Row
import csv
spark = SparkSession.builder.getOrCreate()
f = open("pima/australian.dat")
reader = csv.reader(f,delimiter=' ')
ww = []
for w in reader:
    ww.append(w)

data = map(lambda p: Row(label=int(p[0]),
a_1=float(p[1]),a_2=float(p[2]),
                               label_2=int(p[3]),label_3=int(p[4]),label
_4=int(p[5]),
                               a_3=float(p[6]),label_5 =int(p[7]),label_
6=int(p[8]),
                               a_4=float(p[9]),label_7=int(p[10]),label_
8=int(p[11]),

a_5=float(p[12]),a_6=float(p[13]),label_9=int(p[14]))
                            ,ww)
data = spark.createDataFrame(data)
f.close()
```

In [14]:
```
data.show()
```

```
+-----+-----+-----+----+-----+------+-----+-------+-------+-------+----
---+-------+-------+-------+-------+
|  a_1|  a_2|  a_3| a_4|  a_5|   a_6|label|label_2|label_3|label_4|labe
l_5|label_6|label_7|label_8|label_9|
+-----+-----+-----+----+-----+------+-----+-------+-------+-------+----
---+-------+-------+-------+-------+
|22.08|11.46|1.585| 0.0|100.0|1213.0|    1|      2|      4|      4|
  0|      0|      1|      2|      0|
|22.67|  7.0|0.165| 0.0|160.0|   1.0|    0|      2|      8|      4|
  0|      0|      0|      2|      0|
|29.58| 1.75| 1.25| 0.0|280.0|   1.0|    0|      1|      4|      4|
  0|      0|      1|      2|      0|
|21.67| 11.5|  0.0|11.0|  0.0|   1.0|    0|      1|      5|      3|
  1|      1|      1|      2|      1|
|20.17| 8.17| 1.96|14.0| 60.0| 159.0|    1|      2|      6|      4|
  1|      1|      0|      2|      1|
|15.83|0.585|  1.5| 2.0|100.0|   1.0|    0|      2|      8|      8|
  1|      1|      0|      2|      1|
|17.42|  6.5|0.125| 0.0| 60.0| 101.0|    1|      2|      3|      4|
  0|      0|      0|      2|      0|
|58.67| 4.46| 3.04| 6.0| 43.0| 561.0|    0|      2|     11|      8|
  1|      1|      0|      2|      1|
|27.83|  1.0|  3.0| 0.0|176.0| 538.0|    1|      1|      2|      8|
  0|      0|      0|      2|      0|
|55.75| 7.08| 6.75| 3.0|100.0|  51.0|    0|      2|      4|      8|
  1|      1|      1|      2|      0|
| 33.5| 1.75|  4.5| 4.0|253.0| 858.0|    1|      2|     14|      8|
  1|      1|      1|      2|      1|
|41.42|  5.0|  5.0| 6.0|470.0|   1.0|    1|      2|     11|      8|
  1|      1|      1|      2|      1|
|20.67| 1.25|1.375| 3.0|140.0| 211.0|    1|      1|      8|      8|
  1|      1|      1|      2|      0|
|34.92|  5.0|  7.5| 6.0|  0.0|1001.0|    1|      2|     14|      8|
  1|      1|      1|      2|      1|
|58.58| 2.71|2.415| 0.0|320.0|   1.0|    1|      2|      8|      4|
  0|      0|      1|      2|      0|
|48.08| 6.04| 0.04| 0.0|  0.0|2691.0|    1|      2|      4|      4|
  0|      0|      0|      2|      1|
|29.58|  4.5|  7.5| 2.0|330.0|   1.0|    1|      2|      9|      4|
  1|      1|      1|      2|      1|
|18.92|  9.0| 0.75| 2.0| 88.0| 592.0|    0|      2|      6|      4|
  1|      1|      0|      2|      1|
| 20.0| 1.25|0.125| 0.0|140.0|   5.0|    1|      1|      4|      4|
  0|      0|      0|      2|      0|
|22.42|5.665|2.585| 7.0|129.0|3258.0|    0|      2|     11|      4|
  1|      1|      0|      2|      1|
+-----+-----+-----+----+-----+------+-----+-------+-------+-------+----
---+-------+-------+-------+-------+
only showing top 20 rows
```

Vertorize the Data into feature

In [15]:
```
from pyspark.ml.feature import VectorAssembler
label = ["label"]
assembler = VectorAssembler(
    inputCols=[x for x in data.columns if x not in label],
    outputCol='features')
data = assembler.transform(data)
data.show()
```

```
+-----+-----+-----+----+-----+------+-----+-------+-------+-------+----
---+-------+-------+-------+-------+--------------------+
|  a_1|  a_2|  a_3| a_4|  a_5|   a_6|label|label_2|label_3|label_4|labe
l_5|label_6|label_7|label_8|label_9|            features|
+-----+-----+-----+----+-----+------+-----+-------+-------+-------+----
---+-------+-------+-------+-------+--------------------+
|22.08|11.46|1.585| 0.0|100.0|1213.0|    1|      2|      4|      4|
  0|      0|      1|      2|      0|[22.08,11.46,1.58...|
|22.67|  7.0|0.165| 0.0|160.0|   1.0|    0|      2|      8|      4|
  0|      0|      0|      2|      0|[22.67,7.0,0.165,...|
|29.58| 1.75| 1.25| 0.0|280.0|   1.0|    0|      1|      4|      4|
  0|      0|      1|      2|      0|[29.58,1.75,1.25,...|
|21.67| 11.5|  0.0|11.0|  0.0|   1.0|    0|      1|      5|      3|
  1|      1|      1|      2|      1|[21.67,11.5,0.0,1...|
|20.17| 8.17| 1.96|14.0| 60.0| 159.0|    1|      2|      6|      4|
  1|      1|      0|      2|      1|[20.17,8.17,1.96,...|
|15.83|0.585|  1.5| 2.0|100.0|   1.0|    0|      2|      8|      8|
  1|      1|      0|      2|      1|[15.83,0.585,1.5,...|
|17.42|  6.5|0.125| 0.0| 60.0| 101.0|    1|      2|      3|      4|
  0|      0|      0|      2|      0|[17.42,6.5,0.125,...|
|58.67| 4.46| 3.04| 6.0| 43.0| 561.0|    0|      2|     11|      8|
  1|      1|      0|      2|      1|[58.67,4.46,3.04,...|
|27.83|  1.0|  3.0| 0.0|176.0| 538.0|    1|      1|      2|      8|
  0|      0|      0|      2|      0|[27.83,1.0,3.0,0....|
|55.75| 7.08| 6.75| 3.0|100.0|  51.0|    0|      2|      4|      8|
  1|      1|      1|      2|      0|[55.75,7.08,6.75,...|
| 33.5| 1.75|  4.5| 4.0|253.0| 858.0|    1|      2|     14|      8|
  1|      1|      1|      2|      1|[33.5,1.75,4.5,4....|
|41.42|  5.0|  5.0| 6.0|470.0|   1.0|    1|      2|     11|      8|
  1|      1|      1|      2|      1|[41.42,5.0,5.0,6....|
|20.67| 1.25|1.375| 3.0|140.0| 211.0|    1|      1|      8|      8|
  1|      1|      1|      2|      0|[20.67,1.25,1.375...|
|34.92|  5.0|  7.5| 6.0|  0.0|1001.0|    1|      2|     14|      8|
  1|      1|      1|      2|      1|[34.92,5.0,7.5,6....|
|58.58| 2.71|2.415| 0.0|320.0|   1.0|    1|      2|      8|      4|
  0|      0|      1|      2|      0|[58.58,2.71,2.415...|
|48.08| 6.04| 0.04| 0.0|  0.0|2691.0|    1|      2|      4|      4|
  0|      0|      0|      2|      1|[48.08,6.04,0.04,...|
|29.58|  4.5|  7.5| 2.0|330.0|   1.0|    1|      2|      9|      4|
  1|      1|      1|      2|      1|[29.58,4.5,7.5,2....|
|18.92|  9.0| 0.75| 2.0| 88.0| 592.0|    0|      2|      6|      4|
  1|      1|      0|      2|      1|[18.92,9.0,0.75,2...|
| 20.0| 1.25|0.125| 0.0|140.0|   5.0|    1|      1|      4|      4|
  0|      0|      0|      2|      0|[20.0,1.25,0.125,...|
|22.42|5.665|2.585| 7.0|129.0|3258.0|    0|      2|     11|      4|
  1|      1|      0|      2|      1|[22.42,5.665,2.58...|
+-----+-----+-----+----+-----+------+-----+-------+-------+-------+----
---+-------+-------+-------+-------+--------------------+
only showing top 20 rows
```

Split the Data

```
In [17]: splits = data.select("label", "features").randomSplit([0.8, 0.2], 1234)
         train = splits[1]
         test = splits[0]
```

Use NaiveBayes method to build a model

```
In [18]: from pyspark.ml.classification import NaiveBayes
         from pyspark.ml.evaluation import MulticlassClassificationEvaluator

         nb = NaiveBayes()
         model = nb.fit(train)
         predictions = model.transform(test)

         evaluator = MulticlassClassificationEvaluator(labelCol="label", predicti
         onCol="prediction",
                                                       metricName="accuracy")
         accuracy = evaluator.evaluate(predictions)
         print("Test set accuracy = " + str(accuracy))

         Test set accuracy = 0.388791593695
```

Use DecisionTree method to build a model

```
In [19]: from pyspark.ml.classification import DecisionTreeClassifier
         from pyspark.ml.evaluation import MulticlassClassificationEvaluator

         dt = DecisionTreeClassifier()

         model = dt.fit(train)

         predictions = model.transform(test)

         evaluator = MulticlassClassificationEvaluator(labelCol="label", predicti
         onCol="prediction",metricName="accuracy")
         accuracy = evaluator.evaluate(predictions)
         print("Test set accuracy = " + str(accuracy))

         Test set accuracy = 0.647985989492
```

Use RandomForest method to build a model

In [20]:
```
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

rf = RandomForestClassifier()

model = rf.fit(train)

predictions = model.transform(test)

evaluator = MulticlassClassificationEvaluator(labelCol="label",predictio
nCol="prediction",metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print("Test set accuracy of RandomForest= " + str(accuracy))
```

```
Test set accuracy of RandomForest= 0.647985989492
```

Summary

Naive Bayes model fails when the input data is very independent. And the classifiers used in this experiment give out poor performance on continuous data.