PIMA DATASET

already preprocessed the dataset.

In [3]:
```
from pyspark.ml.feature import HashingTF, IDF, Tokenizer
from pyspark.sql import SparkSession
from pyspark.sql.functions import monotonically_increasing_id
spark = SparkSession.builder.getOrCreate()

data = spark.read.format("csv").option("header",True).option("inferSchem
a",True).\
    load("pima/pima-indians-diabetes.data")
```

Vertorize the Data into feature

In [4]:
```
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)
```

Split the Data

In [5]:
```
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 [6]:
```
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 = 0.61648177496
```

Use DecisionTree method to build a model

In [7]:
```
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 = 0.698890649762
```

Use RandomForest method to build a model

In [10]:
```
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= 0.765451664025
```