SVM IN BIG DATA

which can be used for classification, regression, or other tasks. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data points of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier. LinearSVC in Spark ML supports binary classification with linear SVM.

1-get Data

In [9]:
```
import findspark
import pyspark
findspark.init()
```

In [10]:
```
from pyspark.sql import SparkSession
from pyspark import SparkContext, SparkConf

spark = SparkSession.builder.appName('svm').getOrCreate()
sc = spark.sparkContext
```

In [13]:
```
df= spark.read.format("csv").option("header", "true").load("data/bill_authentica
df.show(5)
```

```
+--------+--------+--------+--------+-----+
|Variance|Skewness|Curtosis| Entropy|Class|
+--------+--------+--------+--------+-----+
|  3.6216|  8.6661| -2.8073|-0.44699|    0|
|  4.5459|  8.1674| -2.4586| -1.4621|    0|
|   3.866| -2.6383|  1.9242| 0.10645|    0|
|  3.4566|  9.5228| -4.0112| -3.5944|    0|
| 0.32924| -4.4552|  4.5718| -0.9888|    0|
+--------+--------+--------+--------+-----+
only showing top 5 rows
```

In [21]:
```
df.printSchema()
```

```
root
 |-- Variance: string (nullable = true)
 |-- Skewness: string (nullable = true)
 |-- Curtosis: string (nullable = true)
 |-- Entropy: string (nullable = true)
 |-- Class: string (nullable = true)
```

In [30]:
```
from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import VectorAssembler
from pyspark.sql.types import FloatType,DoubleType

assembler = VectorAssembler(
    inputCols=["Variance", "Skewness", "Curtosis", "Entropy"],
    outputCol="features")

newdf=df.select(df["Variance"].cast(FloatType()),df["Skewness"].cast(FloatType()

print(newdf.printSchema())
newdf.show(5)
#outputdf = assembler.transform(newdf)
#newdf.show()
#outputdf = assembler.transform(newdf)
#outputdf2=outputdf.withColumnRenamed('Class','label').select('features','label',
#outputdf2.show(5)
```

```
root
 |-- Variance: float (nullable = true)
 |-- Skewness: float (nullable = true)
 |-- Curtosis: float (nullable = true)
 |-- Entropy: float (nullable = true)
 |-- Class: double (nullable = true)

None
+--------+--------+--------+--------+-----+
|Variance|Skewness|Curtosis| Entropy|Class|
+--------+--------+--------+--------+-----+
|  3.6216|  8.6661| -2.8073|-0.44699|  0.0|
|  4.5459|  8.1674| -2.4586| -1.4621|  0.0|
|   3.866| -2.6383|  1.9242| 0.10645|  0.0|
|  3.4566|  9.5228| -4.0112| -3.5944|  0.0|
| 0.32924| -4.4552|  4.5718| -0.9888|  0.0|
+--------+--------+--------+--------+-----+
only showing top 5 rows
```

In [32]:
```
outputdf = assembler.transform(newdf)
newdf.show(5)
outputdf = assembler.transform(newdf)
outputdf2=outputdf.withColumnRenamed('Class','label').select('features','label')
outputdf2.show(5)
```

```
+--------+--------+--------+--------+-----+
|Variance|Skewness|Curtosis| Entropy|Class|
+--------+--------+--------+--------+-----+
|  3.6216|  8.6661| -2.8073|-0.44699|  0.0|
|  4.5459|  8.1674| -2.4586| -1.4621|  0.0|
|   3.866| -2.6383|  1.9242| 0.10645|  0.0|
|  3.4566|  9.5228| -4.0112| -3.5944|  0.0|
| 0.32924| -4.4552|  4.5718| -0.9888|  0.0|
+--------+--------+--------+--------+-----+
only showing top 5 rows

+--------------------+-----+
|            features|label|
+--------------------+-----+
|[3.62159991264343...|  0.0|
|[4.54589986801147...|  0.0|
|[3.86599993705749...|  0.0|
|[3.45659995079040...|  0.0|
|[0.32923999428749...|  0.0|
+--------------------+-----+
only showing top 5 rows
```

Split Data set to train and test

In [ ]:
```

```

In [47]:
```
from pyspark.ml.classification import LinearSVC

# Load training data

lsvc = LinearSVC(maxIter=10, regParam=0.1)

# Fit the model
lsvcModel = lsvc.fit(outputdf2)

prediction=lsvcModel.transform(outputdf2).select("features", "label", "predictio
prediction.show(5)
#print("Coefficients: " + str(lsvcModel.coefficients))
#print("Intercept: " + str(lsvcModel.intercept))
```

```
+--------------------+-----+----------+
|            features|label|prediction|
+--------------------+-----+----------+
|[3.62159991264343...|  0.0|       0.0|
|[4.54589986801147...|  0.0|       0.0|
|[3.86599993705749...|  0.0|       0.0|
|[3.45659995079040...|  0.0|       0.0|
|[0.32923999428749...|  0.0|       1.0|
+--------------------+-----+----------+
only showing top 5 rows
```

In [56]:
```

```

```
+-----+
|label|
+-----+
|  0.0|
|  1.0|
+-----+
```

In [58]:
```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

evaluator = MulticlassClassificationEvaluator()
evaluation = evaluator.evaluate(prediction)
print("evalue=",evaluation)
```

```
evalue= 0.9752594564697722
```

In [ ]:
```

```