In [1]:
```python
import findspark
findspark.init()
```

In [2]:
```python
from pyspark.sql import SparkSession
spark=SparkSession.builder.appName("BayesTheoremApp").getOrCreate()
```

In [3]:
```python
spark
```

Out[3]:

SparkSession - in-memory

SparkContext

Spark UI

Version	`v3.2.1`
Master	`local[*]`
AppName	`BayesTheoremApp`

In [4]:
```python
df=spark.read.option("header","true").csv('D:\Toddler Autism dataset July 2018.csv',inferSchema=True)
df.show()
```

```
+-------+---+---+---+---+---+---+---+---+---+---+--------+--------------+---+-------------+--------+------------------+---------------------+-------------
--+
|Case_No| A1| A2| A3| A4| A5| A6| A7| A8| A9|A10|Age_Mons|Qchat-10-Score|Sex|     Ethnicity|Jaundice|Family_mem_with_ASD|Who completed the test|Class/ASD Trai
ts|
+-------+---+---+---+---+---+---+---+---+---+---+--------+--------------+---+-------------+--------+------------------+---------------------+-------------
--+
|      1|  0|  0|  0|  0|  0|  0|  1|  1|  0|  1|      28|             3|  f|middle eastern|     yes|                 no|         family member|
No|
|      2|  1|  1|  0|  0|  0|  1|  1|  0|  0|  0|      36|             4|  m|White European|     yes|                 no|         family member|             Y
es|
|      3|  1|  0|  0|  0|  0|  0|  1|  1|  0|  1|      36|             4|  m|middle eastern|     yes|                 no|         family member|             Y
es|
|      4|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|      24|            10|  m|      Hispanic|      no|                 no|         family member|             Y
es|
|      5|  1|  1|  0|  1|  1|  1|  1|  1|  1|  1|      20|             9|  f|White European|      no|                yes|         family member|             Y
es|
|      6|  1|  1|  0|  0|  1|  1|  1|  1|  1|  1|      21|             8|  m|         black|      no|                 no|         family member|             Y
es|
|      7|  1|  0|  0|  1|  1|  1|  0|  0|  1|  0|      33|             5|  m|         asian|     yes|                 no|         family member|             Y
es|
|      8|  0|  1|  0|  0|  1|  0|  1|  1|  1|  1|      33|             6|  m|         asian|     yes|                 no|         family member|             Y
es|
|      9|  0|  0|  0|  0|  0|  0|  1|  0|  0|  1|      36|             2|  m|         asian|      no|                 no|         family member|
No|
|     10|  1|  1|  1|  0|  1|  1|  0|  1|  1|  1|      22|             8|  m|   south asian|      no|                 no|  Health Care Profe...|             Y
es|
|     11|  1|  0|  0|  1|  0|  1|  1|  0|  1|  1|      36|             6|  m|      Hispanic|     yes|                yes|         family member|             Y
es|
|     12|  1|  1|  1|  1|  0|  1|  1|  1|  0|  1|      17|             8|  m|middle eastern|     yes|                 no|         family member|             Y
es|
|     13|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|      25|             0|  f|middle eastern|     yes|                 no|         family member|
No|
|     14|  1|  1|  1|  1|  0|  0|  1|  0|  1|  1|      15|             7|  f|middle eastern|     yes|                 no|         family member|             Y
es|
|     15|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|      18|             0|  m|middle eastern|      no|                 no|         family member|
No|
|     16|  1|  1|  1|  0|  1|  0|  1|  1|  0|  1|      12|             7|  m|         black|      no|                 no|         family member|             Y
es|
|     17|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|      36|             0|  m|middle eastern|      no|                yes|         family member|
No|
|     18|  1|  1|  1|  0|  1|  1|  1|  1|  0|  1|      12|             8|  f|middle eastern|     yes|                 no|         family member|             Y
es|
|     19|  1|  0|  0|  0|  1|  0|  0|  0|  0|  1|      29|             3|  f|middle eastern|      no|                 no|         family member|
No|
|     20|  1|  1|  1|  0|  1|  0|  1|  1|  0|  1|      12|             7|  f|         black|      no|                 no|         family member|             Y
es|
+-------+---+---+---+---+---+---+---+---+---+---+--------+--------------+---+-------------+--------+------------------+---------------------+-------------
--+
only showing top 20 rows
```

In [5]:
```python
df.printSchema()
```

```
root
 |-- Case_No: integer (nullable = true)
 |-- A1: integer (nullable = true)
 |-- A2: integer (nullable = true)
 |-- A3: integer (nullable = true)
 |-- A4: integer (nullable = true)
 |-- A5: integer (nullable = true)
 |-- A6: integer (nullable = true)
 |-- A7: integer (nullable = true)
 |-- A8: integer (nullable = true)
 |-- A9: integer (nullable = true)
 |-- A10: integer (nullable = true)
 |-- Age_Mons: integer (nullable = true)
 |-- Qchat-10-Score: integer (nullable = true)
 |-- Sex: string (nullable = true)
 |-- Ethnicity: string (nullable = true)
 |-- Jaundice: string (nullable = true)
 |-- Family_mem_with_ASD: string (nullable = true)
 |-- Who completed the test: string (nullable = true)
 |-- Class/ASD Traits: string (nullable = true)
```

In [6]:
```python
df.groupBy("Class/ASD Traits").count().show()
```

```
+----------------+-----+
|Class/ASD Traits|count|
+----------------+-----+
|              No|  326|
|             Yes|  728|
+----------------+-----+
```

In [7]:
```python
input_columns=df.columns
input_columns=input_columns[1:-1]
dependent_var='Class/ASD Traits'

print(input_columns)
print(dependent_var)
```

```
['A1', 'A2', 'A3', 'A4', 'A5', 'A6', 'A7', 'A8', 'A9', 'A10', 'Age_Mons', 'Qchat-10-Score', 'Sex', 'Ethnicity', 'Jaundice', 'Family_mem_with_ASD', 'Who comple
ted the test']
Class/ASD Traits
```

In [27]:
```python
from pyspark.ml.feature import VectorAssembler
from pyspark.sql.types import *
from pyspark.sql.functions import *
from pyspark.ml.feature import StringIndexer
from pyspark.ml.feature import MinMaxScaler
from pyspark.ml.classification import NaiveBayes
from pyspark.ml.evaluation import BinaryClassificationEvaluator
```

In [10]:
```python
renamed=df.withColumn("label_str",df[dependent_var].cast(StringType()))
indexer=StringIndexer(inputCol="label_str",outputCol="label")
indexed=indexer.fit(renamed).transform(renamed)
indexed.show(5)
```

```
+-------+---+---+---+---+---+---+---+---+---+---+--------+--------------+---+-------------+--------+------------------+---------------------+-------------
--+---------+-----+
|Case_No| A1| A2| A3| A4| A5| A6| A7| A8| A9|A10|Age_Mons|Qchat-10-Score|Sex|     Ethnicity|Jaundice|Family_mem_with_ASD|Who completed the test|Class/ASD Trai
ts|label_str|label|
+-------+---+---+---+---+---+---+---+---+---+---+--------+--------------+---+-------------+--------+------------------+---------------------+-------------
--+---------+-----+
|      1|  0|  0|  0|  0|  0|  0|  1|  1|  0|  1|      28|             3|  f|middle eastern|     yes|                 no|         family member|
No|       No|  1.0|
|      2|  1|  1|  0|  0|  0|  1|  1|  0|  0|  0|      36|             4|  m|White European|     yes|                 no|         family member|             Y
es|      Yes|  0.0|
|      3|  1|  0|  0|  0|  0|  0|  1|  1|  0|  1|      36|             4|  m|middle eastern|     yes|                 no|         family member|             Y
es|      Yes|  0.0|
|      4|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|      24|            10|  m|      Hispanic|      no|                 no|         family member|             Y
es|      Yes|  0.0|
|      5|  1|  1|  0|  1|  1|  1|  1|  1|  1|  1|      20|             9|  f|White European|      no|                yes|         family member|             Y
es|      Yes|  0.0|
+-------+---+---+---+---+---+---+---+---+---+---+--------+--------------+---+-------------+--------+------------------+---------------------+-------------
--+---------+-----+
only showing top 5 rows
```

In [11]:
```python
indexed.printSchema()
```

```
root
 |-- Case_No: integer (nullable = true)
 |-- A1: integer (nullable = true)
 |-- A2: integer (nullable = true)
 |-- A3: integer (nullable = true)
 |-- A4: integer (nullable = true)
 |-- A5: integer (nullable = true)
 |-- A6: integer (nullable = true)
 |-- A7: integer (nullable = true)
 |-- A8: integer (nullable = true)
 |-- A9: integer (nullable = true)
 |-- A10: integer (nullable = true)
 |-- Age_Mons: integer (nullable = true)
 |-- Qchat-10-Score: integer (nullable = true)
 |-- Sex: string (nullable = true)
 |-- Ethnicity: string (nullable = true)
 |-- Jaundice: string (nullable = true)
 |-- Family_mem_with_ASD: string (nullable = true)
 |-- Who completed the test: string (nullable = true)
 |-- Class/ASD Traits: string (nullable = true)
 |-- label_str: string (nullable = true)
 |-- label: double (nullable = false)
```

In [12]:
```python
numeric_inputs=[]
string_inputs=[]
for column in input_columns:
    if str(indexed.schema[column].dataType)=='StringType':
        indexer=StringIndexer(inputCol=column, outputCol=column+"_num")
        indexed=indexer.fit(indexed).transform(indexed)
        new_col_name=column+"_num"
        string_inputs.append(new_col_name)
    else:
        numeric_inputs.append(column)
print('numeric_inputs',numeric_inputs)
print('string_inputs',string_inputs)
```

```
numeric_inputs ['A1', 'A2', 'A3', 'A4', 'A5', 'A6', 'A7', 'A8', 'A9', 'A10', 'Age_Mons', 'Qchat-10-Score']
string_inputs ['Sex_num', 'Ethnicity_num', 'Jaundice_num', 'Family_mem_with_ASD_num', 'Who completed the test_num']
```

In [15]:
```python
# skewness
d={}
for col in numeric_inputs:
    d[col]=indexed.approxQuantile(col,[0.01,0.99],0.25)
for col in numeric_inputs:
    skew=indexed.agg(skewness(indexed[col])).collect()
    skew=skew[0][0]
    if skew>1 :
        indexed=indexed.withColumn(col, \
        log(when(df[col]<d[col][0],d[col][0])\
        .when(indexed[col]>d[col][1],d[col][1])\
        .otherwise(indexed[col]) +1).alias(col))
        print(col+"positive(right) skewness.(skew=)",skew,")")
    if skew<-1 :
        indexed=indexed.withColumn(col, \
        log(when(df[col]<d[col][0],d[col][0])\
        .when(indexed[col]>d[col][1],d[col][1])\
        .otherwise(indexed[col]) +1).alias(col))
        print(col+"negative(left) skewness.(skew=)",skew,")")
print(skew)
```

```
-0.08006349388828135
```

In [23]:
```python
# Negative value
minimums=df.select([min(c).alias(c) for c in df.columns if c in numeric_inputs])
min_array=minimums.select(array(numeric_inputs).alias("mins"))
df_minimum=min_array.select(array_min(min_array.mins)).collect()
df_minimum=df_minimum[0][0]
if df_minimum<0:
    print("WARNING: The Naive Bayes Classifier will not be able to process your dataframe as it contain negative value.")
else:
    print("No Negative values were found in your dataframe.")
```

```
No Negative values were found in your dataframe.
```

In [24]:
```python
# Vector Assembler
features_list=numeric_inputs+string_inputs
assembler=VectorAssembler(inputCols=features_list,outputCol='features')
output=assembler.transform(indexed).select('features','label')
output.show(5,False)
```

```
+-----------------------------------------------------------------------+-----+
|features                                                               |label|
+-----------------------------------------------------------------------+-----+
|(17,[6,7,9,10,11,12,13,14],[1.0,1.0,1.0,28.0,3.0,1.0,2.0,1.0])         |1.0  |
|(17,[0,1,5,6,10,11,14],[1.0,1.0,1.0,1.0,36.0,4.0,1.0])                 |0.0  |
|(17,[0,6,7,9,10,11,13,14],[1.0,1.0,1.0,1.0,36.0,4.0,2.0,1.0])          |0.0  |
|[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,24.0,10.0,0.0,5.0,0.0,0.0,0.0]|0.0  |
|[1.0,1.0,0.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,20.0,9.0,1.0,0.0,0.0,1.0,0.0] |0.0  |
+-----------------------------------------------------------------------+-----+
only showing top 5 rows
```

In [31]:
```python
# final Data
scaler=MinMaxScaler(inputCol="features",outputCol="scaledFeatures",min=0,max=1000)
print("Features scaled to range:[%f, %f]" %(scaler.getMin(), scaler.getMax()))
scalerModel=scaler.fit(output)
scaled_data=scalerModel.transform(output)
final_data=scaled_data.select('label','scaledFeatures')
final_data=final_data.withColumnRenamed('scaledFeatures','features')
final_data.show()
```

```
Features scaled to range:[0.000000, 1000.000000]
+-----+--------------------+
|label|            features|
+-----+--------------------+
|  1.0|(17,[6,7,9,10,11,...|
|  0.0|(17,[0,1,5,6,10,1...|
|  0.0|(17,[0,6,7,9,10,1...|
|  0.0|[1000.0,1000.0,10...|
|  0.0|[1000.0,1000.0,0....|
|  0.0|[1000.0,1000.0,0....|
|  0.0|(17,[0,3,4,5,8,10...|
|  0.0|(17,[1,4,6,7,8,9,...|
|  1.0|(17,[6,9,10,11,13...|
|  0.0|[1000.0,1000.0,10...|
|  0.0|[1000.0,0.0,0.0,1...|
|  0.0|[1000.0,1000.0,10...|
|  1.0|(17,[10,12,13,14]...|
|  0.0|[1000.0,1000.0,10...|
|  1.0|(17,[10,13],[250....|
|  0.0|(17,[0,1,2,4,6,7,...|
|  1.0|(17,[10,13,15],[1...|
|  0.0|[1000.0,1000.0,10...|
|  1.0|(17,[0,4,9,10,11,...|
|  0.0|(17,[0,1,2,4,6,7,...|
+-----+--------------------+
only showing top 20 rows
```

In [32]:
```python
# min max Scalar
scalar=MinMaxScaler(inputCol="features",outputCol="scaler")
```

In [33]:
```python
# split
train,test=final_data.randomSplit([0.70,0.30])
```

In [34]:
```python
nbclassifier=NaiveBayes()
```

In [35]:
```python
nbModel=nbclassifier.fit(train)
```

In [36]:
```python
predictions=nbModel.transform(test)
predictions.select('label','rawPrediction','probability','prediction').show()
predictions.printSchema()
```

```
+-----+--------------------+--------------------+----------+
|label|       rawPrediction|         probability|prediction|
+-----+--------------------+--------------------+----------+
|  0.0|[-25681.783221863...|           [1.0,0.0]|       0.0|
|  0.0|[-22952.650519772...|           [1.0,0.0]|       0.0|
|  0.0|[-23562.526189011...|           [1.0,0.0]|       0.0|
|  0.0|[-24023.320797739...|           [1.0,0.0]|       0.0|
|  0.0|[-26145.401361572...|           [1.0,0.0]|       0.0|
|  0.0|[-25629.034496377...|[1.0,4.9386273184...|       0.0|
|  0.0|[-21256.654119682...|           [1.0,0.0]|       0.0|
|  0.0|[-21910.313581580...|           [1.0,0.0]|       0.0|
|  0.0|[-19781.005356366...|           [1.0,0.0]|       0.0|
|  0.0|[-23837.879111670...|           [1.0,0.0]|       0.0|
|  0.0|[-23311.790976083...|           [1.0,0.0]|       0.0|
|  0.0|[-19850.367295550...|           [1.0,0.0]|       0.0|
|  0.0|[-24460.093157884...|           [1.0,0.0]|       0.0|
|  0.0|[-22692.163766251...|           [1.0,0.0]|       0.0|
|  0.0|[-23088.823765583...|           [1.0,0.0]|       0.0|
|  0.0|[-23345.223382709...|           [1.0,0.0]|       0.0|
|  0.0|[-23219.626034859...|           [1.0,0.0]|       0.0|
|  0.0|[-20607.582160967...|           [1.0,0.0]|       0.0|
|  0.0|[-25890.055489223...|           [1.0,0.0]|       0.0|
|  0.0|[-18641.892081209...|           [1.0,0.0]|       0.0|
+-----+--------------------+--------------------+----------+
only showing top 20 rows

root
 |-- label: double (nullable = false)
 |-- features: vector (nullable = true)
 |-- rawPrediction: vector (nullable = true)
 |-- probability: vector (nullable = true)
 |-- prediction: double (nullable = false)
```

In [38]:
```python
evaluator=BinaryClassificationEvaluator();
accuracy=evaluator.evaluate(predictions)
print("Accuracy of Model :",accuracy)
print("test Error of Model :",(1-accuracy))
```

```
Accuracy of Model : 0.9941620879120878
test Error of Model : 0.005837912087912178
```

In [ ]:

In [ ]: