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

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

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

Out[3]:

SparkSession - in-memory

SparkContext

Spark UI

Version	v3.2.1
Master	local[*]
AppName	BayesTheoremApp

In [46]:
```
df=spark.read.option("header","true").csv('D:\\titanic.csv',inferSchema=True)
df.show()
```

```
+--------+------+--------------------+------+----+-----+-----+----------------+-----------------+-------+--------+
|Survived|Pclass|                Name|   Sex| Age|SibSp|Parch|Parents Children Aboard|Fare|
+--------+------+--------------------+------+----+-----+-----+----------------+-----------------+-------+--------+
|       0|     3|Mr. Owen Harris B...|  male|22.0|    1|    0|               1|   0|  7.25|
|       1|     1|Mrs. John Bradley...|female|38.0|    1|    0|               1|   0|71.2833|
|       1|     3|Miss. Laina Heikk...|female|26.0|    0|    0|               0|   0|  7.925|
|       1|     1|Mrs. Jacques Heat...|female|35.0|    1|    0|               1|   0|   53.1|
|       0|     3|Mr. William Henry...|  male|35.0|    0|    0|               0|   0|   8.05|
|       0|     3|     Mr. James Moran|  male|27.0|    0|    0|               0|   0| 8.4583|
|       0|     1|Mr. Timothy J McC...|  male|54.0|    0|    0|               0|   0|51.8625|
|       1|     3|Master. Gosta Leo...|  male| 2.0|    3|    1|               1|   1| 21.075|
|       1|     3|Mrs. Oscar W (Eli...|female|27.0|    0|    2|               2|   1|11.1333|
|       1|     2|Mrs. Nicholas (Ad...|female|14.0|    1|    0|               1|   0|30.0708|
|       1|     3|Miss. Marguerite ...|female| 4.0|    1|    1|               1|   1|   16.7|
|       1|     1|Miss. Elizabeth B...|female|58.0|    0|    0|               0|   0|  26.55|
|       0|     3|Mr. William Henry...|  male|20.0|    0|    0|               0|   0|   8.05|
|       0|     3|Mr. Anders Johan ...|  male|39.0|    1|    5|               1|   5| 31.275|
|       0|     3|Miss. Hulda Amand...|female|14.0|    0|    0|               0|   0| 7.8542|
|       1|     2|Mrs. (Mary D King...|female|55.0|    0|    0|               0|   0|   16.0|
|       0|     3| Master. Eugene Rice|  male| 2.0|    4|    1|               1|   1| 29.125|
|       1|     2|Mr. Charles Eugen...|  male|23.0|    0|    0|               0|   0|   13.0|
|       0|     3|Mrs. Julius (Emel...|female|31.0|    1|    0|               1|   0|   18.0|
|       1|     3|Mrs. Fatima Masse...|female|22.0|    0|    0|               0|   0|  7.225|
+--------+------+--------------------+------+----+-----+-----+----------------+-----------------+-------+--------+
only showing top 20 rows
```

In [47]:
```
df.printSchema()
```

```
root
 |-- Survived: integer (nullable = true)
 |-- Pclass: integer (nullable = true)
 |-- Name: string (nullable = true)
 |-- Sex: string (nullable = true)
 |-- Age: double (nullable = true)
 |-- SibSp: integer (nullable = true)
 |-- Parch: integer (nullable = true)
 |-- Parents Children Aboard: integer (nullable = true)
 |-- Fare: double (nullable = true)
```

In [48]:
```
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 [49]:
```
df.groupby("Survived").count().show()
```

```
+--------+-----+
|Survived|count|
+--------+-----+
|       1|  342|
|       0|  545|
+--------+-----+
```

In [50]:
```
input_columns=df.columns
input_columns=input_columns[4:6]
dependent_var='Survived'

print(input_columns)
print(dependent_var)
```

```
['Age', 'SiblingsSpouses Aboard']
Survived
```

In [51]:
```
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()
```

```
+--------+------+--------------------+------+----+-----+-----+----------------+-----------------+-------+---------+-----+
|Survived|Pclass|                Name|   Sex| Age|SibSp|Parch|Parents Children Aboard|Fare|label_str|label|
+--------+------+--------------------+------+----+-----+-----+----------------+-----------------+-------+---------+-----+
|       0|     3|Mr. Owen Harris B...|  male|22.0|    1|    0|               1|   0|  7.25|        0|  0.0|
|       1|     1|Mrs. John Bradley...|female|38.0|    1|    0|               1|   0|71.2833|        1|  1.0|
|       1|     3|Miss. Laina Heikk...|female|26.0|    0|    0|               0|   0|  7.925|        1|  1.0|
|       1|     1|Mrs. Jacques Heat...|female|35.0|    1|    0|               1|   0|   53.1|        1|  1.0|
|       0|     3|Mr. William Henry...|  male|35.0|    0|    0|               0|   0|   8.05|        0|  0.0|
|       0|     3|     Mr. James Moran|  male|27.0|    0|    0|               0|   0| 8.4583|        0|  0.0|
|       0|     1|Mr. Timothy J McC...|  male|54.0|    0|    0|               0|   0|51.8625|        0|  0.0|
|       0|     3|Master. Gosta Leo...|  male| 2.0|    3|    1|               1|   1| 21.075|        0|  0.0|
|       1|     3|Mrs. Oscar W (Eli...|female|27.0|    0|    2|               2|   1|11.1333|        1|  1.0|
|       1|     2|Mrs. Nicholas (Ad...|female|14.0|    1|    0|               1|   0|30.0708|        1|  1.0|
|       1|     3|Miss. Marguerite ...|female| 4.0|    1|    1|               1|   1|   16.7|        1|  1.0|
|       1|     1|Miss. Elizabeth B...|female|58.0|    0|    0|               0|   0|  26.55|        1|  1.0|
|       0|     3|Mr. William Henry...|  male|20.0|    0|    0|               0|   0|   8.05|        0|  0.0|
|       0|     3|Mr. Anders Johan ...|  male|39.0|    1|    5|               1|   5| 31.275|        0|  0.0|
|       0|     3|Miss. Hulda Amand...|female|14.0|    0|    0|               0|   0| 7.8542|        0|  0.0|
|       1|     2|Mrs. (Mary D King...|female|55.0|    0|    0|               0|   0|   16.0|        1|  1.0|
|       0|     3| Master. Eugene Rice|  male| 2.0|    4|    1|               1|   1| 29.125|        0|  0.0|
|       1|     2|Mr. Charles Eugen...|  male|23.0|    0|    0|               0|   0|   13.0|        1|  1.0|
|       0|     3|Mrs. Julius (Emel...|female|31.0|    1|    0|               1|   0|   18.0|        0|  0.0|
|       1|     3|Mrs. Fatima Masse...|female|22.0|    0|    0|               0|   0|  7.225|        1|  1.0|
+--------+------+--------------------+------+----+-----+-----+----------------+-----------------+-------+---------+-----+
only showing top 20 rows
```

In [52]:
```
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 ['Age', 'SiblingsSpouses Aboard']
string_inputs []
```

In [53]:
```
# 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()
    skewness=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)
```

```
SiblingsSpouses Aboardpositive(right) skewness.(skew>) 3.6805221729276023 )
3.6805221729276023
```

In [54]:
```
# 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 [55]:
```
# 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|
+-------------------------+-----+
|[22.0,0.6931471805599453]|0.0  |
|[38.0,0.6931471805599453]|1.0  |
|[26.0,0.0]               |1.0  |
|[35.0,0.6931471805599453]|1.0  |
|[35.0,0.0]               |0.0  |
+-------------------------+-----+
only showing top 5 rows
```

In [56]:
```
# 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|
+-----+--------------------+
|  0.0|[271.173661724051...|
|  1.0|[472.229203317416...|
|  1.0|[321.437547122392...|
|  1.0|[434.531289268660...|
|  0.0|[434.531289268660...|
|  0.0|[334.003518471977...|
|  0.0|[673.284744910781...|
|  0.0|[19.8542347323448...|
|  1.0|[334.003518471977...|
|  1.0|[170.645809927368...|
|  1.0|[44.9861774315154...|
|  1.0|[723.548630309122...|
|  0.0|[246.841719024880...|
|  0.0|[484.795174667001...|
|  0.0|[170.645809927368...|
|  1.0|[685.850716269367...|
|  0.0|[19.8542347323448...|
|  1.0|[283.739633073636...|
|  0.0|[384.267403870319...|
|  1.0|[271.173661724051...|
+-----+--------------------+
only showing top 20 rows
```

In [57]:
```
# min max Scalar
scalar=MinMaxScaler(inputCol="features",outputCol="scaler")
```

In [58]:
```
# split
train,test=final_data.randomSplit([0.70,0.30])
```

In [59]:
```
nbclassifier=NaiveBayes()
```

In [60]:
```
nbModel=nbclassifier.fit(train)
```

In [61]:
```
predictions=nbModel.transform(test)
predictions.select('label','rawPrediction','probability','prediction').show()
predictions.printSchema()
```

```
+-----+--------------------+--------------------+----------+
|label|       rawPrediction|         probability|prediction|
+-----+--------------------+--------------------+----------+
|  0.0|[-992.4785299171...|[3.55896850459963...|       1.0|
|  0.0|[-859.0117883884...|[3.74738959926080...|       1.0|
|  0.0|[-996.2436913152...|[5.34392229619734...|       1.0|
|  0.0|[-862.7769497784...|[5.62684350989519...|       1.0|
|  0.0|[-1003.774014311...|[1.20484759321544...|       1.0|
|  0.0|[-870.3072721746...|[1.26861557389421...|       1.0|
|  0.0|[-1011.304336957...|[2.71644648707285...|       1.0|
|  0.0|[-451.5563695891...|[2.52794817070870...|       1.0|
|  0.0|[-881.6027568489...|[4.29481821749465...|       1.0|
|  0.0|[-1134.7264501392...|[6.62687113676345...|       1.0|
|  0.0|[-36.5706279770569...|[0.98695150838391...|       0.0|
|  0.0|[-51.6312736909803...|[0.99740586571205...|       0.0|
|  0.0|[-1041.4256282292...|[7.01927185936643...|       1.0|
|  0.0|[-1402.9107890707...|[5.19776001902935...|       1.0|
|  0.0|[-59.1615964129439...|[0.99684774499192...|       0.0|
|  0.0|[-485.44282227738...|[9.80826236515007...|       1.0|
|  0.0|[-62.9265753839228...|[0.99923232209696...|       0.0|
|  0.0|[-62.9265753839228...|[0.99923232209696...|       0.0|
|  0.0|[-62.9265753839228...|[0.99923232209696...|       0.0|
|  0.0|[-62.9265753839228...|[0.99923232209696...|       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 [62]:
```
evaluator=BinaryClassificationEvaluator();
accuracy=evaluator.evaluate(predictions)
print("Accuracy of Model :",accuracy)
print("test Error of Model :",(1-accuracy))
```

```
Accuracy of Model : 0.4785948039898425
test Error of Model : 0.5214051960195017
```

In [ ]: