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 [4]:

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

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

In [5]:

```
df.printSchema()
```

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

In [6]:

```
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 [7]:

```
# Q1.Calculate the conditional probability that a person survives given their sex and passenger-class.
df.groupBy("Survived").count().show()
```

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

In [9]:

```
input_columns=df.columns
input_columns=input_columns[1:4:2]
dependent_var='Survived'

print(input_columns)
print(dependent_var)
```

```
['Pclass', 'Sex']
Survived
```

In [10]:

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

In [11]:

```
indexed.printSchema()
```

```
root
 |-- Survived: integer (nullable = true)
 |-- Pclass: integer (nullable = true)
 |-- Name: string (nullable = true)
 |-- Sex: string (nullable = true)
 |-- Age: double (nullable = true)
 |-- Siblings/Spouses Aboard: integer (nullable = true)
 |-- Parents/Children Aboard: integer (nullable = true)
 |-- Fare: double (nullable = true)
 |-- label_str: string (nullable = true)
 |-- label: double (nullable = false)
```

In [12]:

```
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 ['Pclass']
string_inputs ['Sex_num']
```

In [13]:

```
# 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)
```

```
-0.6223541098616062
```

In [14]:

```
# 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 [15]:

```
# 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|
+---------+-----+
|[3.0,0.0]|0.0  |
|[1.0,1.0]|1.0  |
|[3.0,1.0]|1.0  |
|[1.0,1.0]|1.0  |
|[3.0,0.0]|0.0  |
+---------+-----+
only showing top 5 rows
```

In [16]:

```
# 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|   [1000.0,0.0]|
|  1.0|   [0.0,1000.0]|
|  1.0|[1000.0,1000.0]|
|  1.0|   [0.0,1000.0]|
|  0.0|   [1000.0,0.0]|
|  0.0|   [1000.0,0.0]|
|  0.0|      (2,[],[])|
|  0.0|   [1000.0,0.0]|
|  1.0|[1000.0,1000.0]|
|  1.0| [500.0,1000.0]|
|  1.0|[1000.0,1000.0]|
|  1.0|   [0.0,1000.0]|
|  0.0|   [1000.0,0.0]|
|  0.0|   [1000.0,0.0]|
|  0.0|[1000.0,1000.0]|
|  1.0| [500.0,1000.0]|
|  0.0|   [1000.0,0.0]|
|  1.0|    [500.0,0.0]|
|  0.0|[1000.0,1000.0]|
|  1.0|[1000.0,1000.0]|
+-----+---------------+
only showing top 20 rows
```

In [17]:

```
# min max Scalar
scalar=MinMaxScaler(inputCol="features",outputCol="scaler")
```

In [18]:

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

In [19]:

```
nbclassifier=NaiveBayes()
```

In [20]:

```
nbModel=nbclassifier.fit(train)
```

In [21]:

```
predictions=nbModel.transform(test)
predictions.select('label','rawPrediction','probability','prediction').show()
predictions.printSchema()
```

```
+-----+--------------------+--------------------+----------+
|label|       rawPrediction|         probability|prediction|
+-----+--------------------+--------------------+----------+
|  0.0|[-0.5024573740954...|[0.60504201680672...|       0.0|
|  0.0|[-0.5024573740954...|[0.60504201680672...|       0.0|
|  0.0|[-0.5024573740954...|[0.60504201680672...|       0.0|
|  0.0|[-0.5024573740954...|[0.60504201680672...|       0.0|
|  0.0|[-0.5024573740954...|[0.60504201680672...|       0.0|
|  0.0|[-0.5024573740954...|[0.60504201680672...|       0.0|
|  0.0|[-0.5024573740954...|[0.60504201680672...|       0.0|
|  0.0|[-0.5024573740954...|[0.60504201680672...|       0.0|
|  0.0|[-0.5024573740954...|[0.60504201680672...|       0.0|
|  0.0|[-0.5024573740954...|[0.60504201680672...|       0.0|
|  0.0|[-0.5024573740954...|[0.60504201680672...|       0.0|
|  0.0|[-0.5024573740954...|[0.60504201680672...|       0.0|
|  0.0|[-0.5024573740954...|[0.60504201680672...|       0.0|
|  0.0|[-0.5024573740954...|[0.60504201680672...|       0.0|
|  0.0|[-0.5024573740954...|[0.60504201680672...|       0.0|
|  0.0|[-0.5024573740954...|[0.60504201680672...|       0.0|
|  0.0|[-0.5024573740954...|[0.60504201680672...|       0.0|
|  0.0|[-0.5024573740954...|[0.60504201680672...|       0.0|
|  0.0|[-0.5024573740954...|[0.60504201680672...|       0.0|
|  0.0|[-0.5024573740954...|[0.60504201680672...|       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 [22]:

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

```
Accuracy of Model : 0.4483024691358024
test Error of Model : 0.5516975308641976
```

In [ ]:

In [ ]: