```
!pip install pyspark
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting pyspark
  Downloading pyspark-3.3.2.tar.gz (281.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 281.4/281.4 MB 4.3 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting py4j==0.10.9.5
  Downloading py4j-0.10.9.5-py2.py3-none-any.whl (199 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 199.7/199.7 KB 8.5 MB/s eta 0:00:00
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.3.2-py2.py3-none-any.whl size=281824025 sha256=18c99a52cdac1fd9082f619e0e02a9b4f04ce0414
  Stored in directory: /root/.cache/pip/wheels/b1/59/a0/a1a0624b5e865fd389919c1a10f53aec9b12195d6747710baf
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.9.5 pyspark-3.3.2
```

```
from pyspark.ml.classification import DecisionTreeClassifier,NaiveBayes,LinearSVC,RandomForestClassifier
from pyspark.ml.feature import VectorAssembler,StringIndexer,StandardScaler
from pyspark.sql.functions import when,count,col


from pyspark.sql import SparkSession
spark=SparkSession.builder.getOrCreate()


df=spark.read.csv("cancerdata_test.csv",header=True,inferSchema=True)


df.show()
```

```
+--------+---------+-----------+------------+--------------+---------+--------------+----------------+--------------+----------------
|      id|diagnosis|radius_mean|texture_mean|perimeter_mean|area_mean|smoothness_mean|compactness_mean|concavity_mean|concave points_mea
+--------+---------+-----------+------------+--------------+---------+--------------+----------------+--------------+----------------
|  842302|        M|      17.99|       10.38|         122.8|   1001.0|        0.1184|          0.2776|        0.3001|             0.147
|  842517|        M|      20.57|       17.77|         132.9|   1326.0|       0.08474|         0.07864|        0.0869|            0.0701
|84300903|        M|      19.69|       21.25|         130.0|   1203.0|        0.1096|          0.1599|        0.1974|             0.127
|84348301|        M|      11.42|       20.38|         77.58|    386.1|        0.1425|          0.2839|        0.2414|             0.105
|84358402|        M|      20.29|       14.34|         135.1|   1297.0|        0.1003|          0.1328|         0.198|             0.104
|  843786|        M|      12.45|        15.7|         82.57|    477.1|        0.1278|            0.17|        0.1578|            0.0808
|  844359|        M|      18.25|       19.98|         119.6|   1040.0|       0.09463|           0.109|        0.1127|              0.07
|84458202|        M|      13.71|       20.83|          90.2|    577.9|        0.1189|          0.1645|       0.09366|            0.0598
|  844981|        M|       13.0|       21.82|          87.5|    519.8|        0.1273|          0.1932|        0.1859|            0.0935
|84501001|        M|      12.46|       24.04|         83.97|    475.9|        0.1186|          0.2396|        0.2273|            0.0854
|  845636|        M|      16.02|       23.24|         102.7|    797.8|       0.08206|         0.06669|       0.03299|            0.0332
|84610002|        M|      15.78|       17.89|         103.6|    781.0|        0.0971|          0.1292|       0.09954|            0.0666
|  846226|        M|      19.17|        24.8|         132.4|   1123.0|        0.0974|          0.2458|        0.2065|             0.111
|  846381|        M|      15.85|       23.95|         103.7|    782.7|       0.08401|          0.1002|       0.09938|            0.0536
|84667401|        M|      13.73|       22.61|          93.6|    578.3|        0.1131|          0.2293|        0.2128|            0.0802
|84799002|        M|      14.54|       27.54|         96.73|    658.8|        0.1139|          0.1595|        0.1639|            0.0736
|  848406|        M|      14.68|       20.13|         94.74|    684.5|       0.09867|           0.072|       0.07395|            0.0525
|84862001|        M|      16.13|       20.68|         108.1|    798.8|         0.117|          0.2022|        0.1722|             0.102
|  849014|        M|      19.81|       22.15|         130.0|   1260.0|       0.09831|          0.1027|        0.1479|            0.0949
| 8510426|        B|      13.54|       14.36|         87.46|    566.3|       0.09779|         0.08129|       0.06664|            0.0478
+--------+---------+-----------+------------+--------------+---------+--------------+----------------+--------------+----------------
only showing top 20 rows
```

```
df.select("id","_c32").show()
#As these two columns do not give information, so we'll drop it
```

```
+--------+----+
|      id|_c32|
+--------+----+
|  842302|null|
|  842517|null|
|84300903|null|
|84348301|null|
|84358402|null|
|  843786|null|
|  844359|null|
|84458202|null|
|  844981|null|
|84501001|null|
|  845636|null|
|84610002|null|
|  846226|null|
```

```
|  846381|null|
|84667401|null|
|84799002|null|
|  848406|null|
|84862001|null|
|  849014|null|
| 8510426|null|
+--------+----+
only showing top 20 rows
```

Droping two columns having null values and give no infromation

```
df=df.drop("id").drop("_c32")
```

```
df.groupBy("diagnosis").count().show()
```

```
+---------+-----+
|diagnosis|count|
+---------+-----+
|     None|    3|
|        B|   32|
|        M|   60|
|     NULL|    4|
+---------+-----+
```

dataset target column contains None and NULL values, so i have replaced with most values in target column i-e "M"

```
cols=[c for c in df.columns if c!='diagnosis']
```

```
df=df.withColumn("diagnosis",when(df.diagnosis=="None","M").when(df.diagnosis=="NULL","M").otherwise(df.diagnosis))
```

```
df.groupBy("diagnosis").count().show()
```

```
+---------+-----+
|diagnosis|count|
+---------+-----+
|        B|   32|
|        M|   67|
+---------+-----+
```

```
df.show()
```

```
+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+--------------------+-------
|diagnosis|radius_mean|texture_mean|perimeter_mean|area_mean|smoothness_mean|compactness_mean|concavity_mean|concave points_mean|symmetr
+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+--------------------+-------
|        M|      17.99|       10.38|         122.8|   1001.0|         0.1184|          0.2776|        0.3001|              0.1471|
|        M|      20.57|       17.77|         132.9|   1326.0|        0.08474|         0.07864|        0.0869|             0.07017|
|        M|      19.69|       21.25|         130.0|   1203.0|         0.1096|          0.1599|        0.1974|              0.1279|
|        M|      11.42|       20.38|         77.58|    386.1|         0.1425|          0.2839|        0.2414|              0.1052|
|        M|      20.29|       14.34|         135.1|   1297.0|         0.1003|          0.1328|         0.198|              0.1043|
|        M|      12.45|        15.7|         82.57|    477.1|         0.1278|            0.17|        0.1578|             0.08089|
|        M|      18.25|       19.98|         119.6|   1040.0|        0.09463|           0.109|        0.1127|               0.074|
|        M|      13.71|       20.83|          90.2|    577.9|         0.1189|          0.1645|       0.09366|             0.05985|
|        M|       13.0|       21.82|          87.5|    519.8|         0.1273|          0.1932|        0.1859|             0.09353|
|        M|      12.46|       24.04|         83.97|    475.9|         0.1186|          0.2396|        0.2273|             0.08543|
|        M|      16.02|       23.24|         102.7|    797.8|        0.08206|         0.06669|       0.03299|             0.03323|
|        M|      15.78|       17.89|         103.6|    781.0|         0.0971|          0.1292|       0.09954|             0.06606|
|        M|      19.17|        24.8|         132.4|   1123.0|         0.0974|          0.2458|        0.2065|              0.1118|
|        M|      15.85|       23.95|         103.7|    782.7|        0.08401|          0.1002|       0.09938|             0.05364|
|        M|      13.73|       22.61|          93.6|    578.3|         0.1131|          0.2293|        0.2128|             0.08025|
|        M|      14.54|       27.54|         96.73|    658.8|         0.1139|          0.1595|        0.1639|             0.07364|
|        M|      14.68|       20.13|         94.74|    684.5|        0.09867|           0.072|       0.07395|             0.05259|
|        M|      16.13|       20.68|         108.1|    798.8|          0.117|          0.2022|        0.1722|              0.1028|
|        M|      19.81|       22.15|         130.0|   1260.0|        0.09831|          0.1027|        0.1479|             0.09498|
|        B|      13.54|       14.36|         87.46|    566.3|        0.09779|         0.08129|       0.06664|             0.04781|
+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+--------------------+-------
only showing top 20 rows
```

```
va=VectorAssembler(inputCols=cols,outputCol='features')
```

```
vaDF=va.transform(df)
```

```
vaDF.show()
```

```
+---------+-----------+------------+--------------+---------+--------------+---------------+-------------+-------------------+-------
|diagnosis|radius_mean|texture_mean|perimeter_mean|area_mean|smoothness_mean|compactness_mean|concavity_mean|concave points_mean|symmetr
+---------+-----------+------------+--------------+---------+--------------+---------------+-------------+-------------------+-------
|        M|      17.99|       10.38|         122.8|   1001.0|         0.1184|         0.2776|        0.3001|             0.1471|
|        M|      20.57|       17.77|         132.9|   1326.0|        0.08474|        0.07864|        0.0869|            0.07017|
|        M|      19.69|       21.25|         130.0|   1203.0|         0.1096|         0.1599|        0.1974|             0.1279|
|        M|      11.42|       20.38|         77.58|    386.1|         0.1425|         0.2839|        0.2414|             0.1052|
|        M|      20.29|       14.34|         135.1|   1297.0|         0.1003|         0.1328|         0.198|             0.1043|
|        M|      12.45|        15.7|         82.57|    477.1|         0.1278|           0.17|        0.1578|            0.08089|
|        M|      18.25|       19.98|         119.6|   1040.0|        0.09463|          0.109|        0.1127|              0.074|
|        M|      13.71|       20.83|          90.2|    577.9|         0.1189|         0.1645|       0.09366|            0.05985|
|        M|       13.0|       21.82|          87.5|    519.8|         0.1273|         0.1932|        0.1859|            0.09353|
|        M|      12.46|       24.04|         83.97|    475.9|         0.1186|         0.2396|        0.2273|            0.08543|
|        M|      16.02|       23.24|         102.7|    797.8|        0.08206|        0.06669|       0.03299|            0.03323|
|        M|      15.78|       17.89|         103.6|    781.0|         0.0971|         0.1292|       0.09954|            0.06606|
|        M|      19.17|        24.8|         132.4|   1123.0|         0.0974|         0.2458|        0.2065|             0.1118|
|        M|      15.85|       23.95|         103.7|    782.7|        0.08401|         0.1002|       0.09938|            0.05364|
|        M|      13.73|       22.61|          93.6|    578.3|         0.1131|         0.2293|        0.2128|            0.08025|
|        M|      14.54|       27.54|         96.73|    658.8|         0.1139|         0.1595|        0.1639|            0.07364|
|        M|      14.68|       20.13|         94.74|    684.5|        0.09867|          0.072|       0.07395|            0.05259|
|        M|      16.13|       20.68|         108.1|    798.8|          0.117|         0.2022|        0.1722|             0.1028|
|        M|      19.81|       22.15|         130.0|   1260.0|        0.09831|         0.1027|        0.1479|            0.09498|
|        B|      13.54|       14.36|         87.46|    566.3|        0.09779|        0.08129|       0.06664|            0.04781|
+---------+-----------+------------+--------------+---------+--------------+---------------+-------------+-------------------+-------
only showing top 20 rows
```

```
si=StringIndexer(inputCol='diagnosis',outputCol='indexedLabel')
```

```
siDF=si.fit(vaDF).transform(vaDF)
```

```
siDF.columns
```

```
['diagnosis',
 'radius_mean',
 'texture_mean',
 'perimeter_mean',
 'area_mean',
 'smoothness_mean',
 'compactness_mean',
 'concavity_mean',
 'concave points_mean',
 'symmetry_mean',
 'fractal_dimension_mean',
 'radius_se',
 'texture_se',
 'perimeter_se',
 'area_se',
 'smoothness_se',
 'compactness_se',
 'concavity_se',
 'concave points_se',
 'symmetry_se',
 'fractal_dimension_se',
 'radius_worst',
 'texture_worst',
 'perimeter_worst',
 'area_worst',
 'smoothness_worst',
 'compactness_worst',
 'concavity_worst',
 'concave points_worst',
 'symmetry_worst',
 'fractal_dimension_worst',
 'features',
 'indexedLabel']
```

```
ss=StandardScaler(inputCol='features',outputCol='scaledFeatures')


ssDF=ss.fit(siDF).transform(siDF)


ssDF=ssDF.select('scaledFeatures','indexedLabel')


ssDF.show()
```

```
+--------------------+------------+
|      scaledFeatures|indexedLabel|
+--------------------+------------+
|[5.34436357265209...|         0.0|
|[6.11081482431648...|         0.0|
|[5.84938959119064...|         0.0|
|[3.39258654806486...|         0.0|
|[6.0276340683219,...|         0.0|
|[3.69857290047351...|         0.0|
|[5.42160284607563...|         0.0|
|[4.07288630244915...|         0.0|
|[3.86196367117716...|         0.0|
|[3.70154364175903...|         0.0|
|[4.75912753940447...|         0.0|
|[4.68782974855197...|         0.0|
|[5.69491104434356...|         0.0|
|[4.70862493755062...|         0.0|
|[4.07882778502019...|         0.0|
|[4.31945782914738...|         0.0|
|[4.36104820714467...|         0.0|
|[4.79180569354520...|         0.0|
|[5.88503848661689...|         0.0|
|[4.02238370059529...|         1.0|
+--------------------+------------+
only showing top 20 rows
```

```
finalDF=ssDF


finalDF.show()
```

```
+--------------------+------------+
|      scaledFeatures|indexedLabel|
+--------------------+------------+
|[5.34436357265209...|         0.0|
|[6.11081482431648...|         0.0|
|[5.84938959119064...|         0.0|
|[3.39258654806486...|         0.0|
|[6.0276340683219,...|         0.0|
|[3.69857290047351...|         0.0|
|[5.42160284607563...|         0.0|
|[4.07288630244915...|         0.0|
|[3.86196367117716...|         0.0|
|[3.70154364175903...|         0.0|
|[4.75912753940447...|         0.0|
|[4.68782974855197...|         0.0|
|[5.69491104434356...|         0.0|
|[4.70862493755062...|         0.0|
|[4.07882778502019...|         0.0|
|[4.31945782914738...|         0.0|
|[4.36104820714467...|         0.0|
|[4.79180569354520...|         0.0|
|[5.88503848661689...|         0.0|
|[4.02238370059529...|         1.0|
+--------------------+------------+
only showing top 20 rows
```

Here user will input for the number of bootstrap

```
NumBootstrap = input ("Please enter the number of bootstrap :")
```

```
Please enter the number of bootstrap :3
```

```
NumBootstrap=int(NumBootstrap)
```

```
BootstrapDataList=[]
```

Creating Bootstrap (with replacement of data)

```
for i in range(0,NumBootstrap):
  BootstrapDataList.append(finalDF.sample(withReplacement=True,fraction=1.0))
```

```
#Records in each bootsrap
for i in range(0,NumBootstrap):
  BootstrapDataList[i].groupBy('indexedLabel').count().show()
```

```
+------------+-----+
|indexedLabel|count|
+------------+-----+
|         0.0|   68|
|         1.0|   25|
+------------+-----+

+------------+-----+
|indexedLabel|count|
+------------+-----+
|         0.0|   70|
|         1.0|   22|
+------------+-----+

+------------+-----+
|indexedLabel|count|
+------------+-----+
|         0.0|   68|
|         1.0|   23|
+------------+-----+
```

```
finalDF.first()
```

```
Row(scaledFeatures=DenseVector([5.3444, 2.7473, 5.2693, 3.1114, 8.963, 4.5196, 3.7875, 3.8414, 7.8097, 9.5875, 4.5302, 1.7497, 4.4108,
4.4778, 2.6337, 2.4153, 1.603, 2.717, 2.8734, 2.2206, 5.2949, 3.0329, 5.451, 3.6811, 7.0349, 3.2331, 2.9889, 3.8686, 5.6906, 4.9676]),
indexedLabel=0.0)
```

```
finalDF.show(truncate=False)
```

```
+--------------------------------------------------------------------------------------------------------------------------------------
|scaledFeatures
+--------------------------------------------------------------------------------------------------------------------------------------
|[5.344363572652093,2.747268060482601,5.269346521655232,3.111378111518874,8.962979755367378,4.519641956714584,3.787478293110162,3.841380
|[6.110814824316485,4.70317470469902,5.702737400064987,4.121565810063963,6.4148893958600635,1.2803481393228922,1.0967406320269015,1.8324
|[5.849389591190646,5.62422411225966,5.578298434977038,3.7392486195376673,8.2968123411171,2.6033528417819234,2.491330273442006,3.3399898
|[3.3925865480648643,5.3939617603695,3.3289568660424504,1.2001029858715657,10.787370060302797,4.622213081812934,3.0466419858606906,2.747
|[6.0276340683219,3.795358765637813,5.7971393735799825,4.031425984655324,7.592794505602601,2.1621341925493396,2.4989027058840794,2.72369
|[3.6985729004735166,4.155309108822431,3.5430777059696457,1.4829555414641906,9.67456767513472,2.767792264558643,1.9915497322651903,2.112
|[5.421602846075637,5.2880940123740245,5.132034560178874,3.2326006353442844,7.163570728466342,1.7746432755111299,1.422355227036039,1.932
|[4.07288630244915,5.51306297686441,3.870480914114837,1.7962691415052519,9.000830176631599,2.6782460442346867,1.18205670420759,1.5629272
|[3.861963671177166,5.775085653153214,3.75462394661916,1.6156786637038065,9.636717253870499,3.1455145030160576,2.346191984968941,2.44244
|[3.701543641759038,6.362651654528105,3.6031516891155526,1.4792256176541778,8.978119923873065,3.9009589799308872,2.8686898234719758,2.23
|[4.759127539404477,6.150916158537153,4.40685576363186,2.4797776796900672,6.2120111378838425,1.0857886242553876,0.4163575771066453,0.867
|[4.687829748551975,4.734935029097662,4.445474752797085,2.4275587463498907,7.350551809511591,2.1035221210645685,1.2562665421399053,1.725
|[5.69491104434356,6.563800375719509,5.681282406084306,3.4905870322034915,7.373262062270123,4.001901991932438,2.6061788321467794,2.91955
|[4.7086249375506215,6.338831411229123,4.449765751593222,2.432842805080742,6.359627780814303,1.6313693229928,1.254247226822019,1.4007588
|[4.0788277850201915,5.984174455444278,4.016374873183467,1.7975124494419226,8.561765289966642,3.7332633309605696,2.685689372788546,2.095
|[4.319457829147384,7.28899444948852,4.15068313550253,2.047728171696937,8.622325963989395,2.5968403893947265,2.068536128759599,1.9230402
|[4.361048207144677,5.327794417872328,4.06529225945942,2.127610706628041,7.4694021322812425,1.172241429695425,0.9333022984854933,1.37333
|[4.791805693545206,5.473362571366107,4.638569698623213,2.4828859495317444,8.85699857582756,3.2920446817279854,2.1732881108749416,2.6845
|[5.885038486616897,5.86242654524948,5.578298434977038,3.9164200005132677,7.4421498289710035,1.67207215041278,1.8666045969709866,2.48031
|[4.022383700595294,3.800652153037587,3.752907547100705,1.7602132113417963,7.402785390856215,1.323493136388071,0.841044829899571,1.24851
+--------------------------------------------------------------------------------------------------------------------------------------
only showing top 20 rows
```

```
finalDF.count()
```

```
99
```

```
train,testoneRowDF=finalDF.randomSplit([0.7,0.3])
```

```
testoneRowDF.show(truncate=False)
testoneRowDF.printSchema()
```

```
+------------------------------------------------------------------------------------------------------------------------------------------------
|scaledFeatures
+------------------------------------------------------------------------------------------------------------------------------------------------
|[2.434819557612927,4.457032190609538,2.2188754774820203,0.627559681034626,6.5102724574458986,0.9675876134277654,0.20041704530019785,0.1
|[2.560184839861909,3.1204518721666537,2.331728745820402,0.6978065794565306,7.382346163373535,0.8583412246325391,0.26011305438520643,0.2
|[2.682282306696818,4.5867201819039956,2.5226781922484616,0.7786215953401378,8.069709813531777,2.300523805777272,3.9502855906147314,1.14
|[3.252961707645382,5.65069104925853,3.0852281344219157,1.1534789382464077,9.288493378239673,1.9830417519014276,1.3176032449206965,1.48
|[3.3925865480648643,5.3939617603695,3.3289568660424504,1.2001029858715657,10.787370060302797,4.622213081812934,3.0466419858606906,2.747
|[3.4935917517725747,5.716858391755702,3.2062343004729557,1.3300286652536724,6.538281769181422,0.8085209638704836,0.2091253426085818,0.2
|[3.5470650949119507,4.827569308593703,3.2487151885547036,1.3601788827179413,6.253646601274485,0.7735165322893006,0.2488806129294648 7,0.
|[3.5648895426250764,4.142075640322997,3.3019235736267927,1.3778960208155013,7.3603929190402875,1.166543033856628,0.523886117784081 5,0.4
|[3.6985729004735166,4.155309108822431,3.5430777059696457,1.4829555414641906,9.67456767513472,2.767792264558643,1.9915497322651903,2.112
|[3.796607362895706,4.364397911113496,3.491585720416012,1.5619055954427912,7.4421498289710035,0.8521543948647021,0.4610349285148758,0.74
|[3.8203732931798733,4.764048659796418,3.5696818985056904,1.5737170208411646,7.5201216967752975,1.5541967622045179,0.4908198294536962,0.
|[3.861963671177166,5.775085653153214,3.75462394661916,1.6156786637038065,9.636717253870499,3.1455145030160576,2.346191984968941,2.44244
|[3.912466273031021,4.938730443988954,3.68940076491789,1.6616810573606293,8.766157564793431,2.0042072221598173,1.547300362330243,1.91677
|[3.9421736858862304,3.9065199010330627,3.636192379845801,1.7148324716533094,5.567796967966813,0.8230111704319963,0.4115617032266658,0.6
|[3.9629688748848766,4.1976562080206214,3.711284858778184,1.6163003176721422,8.160550824565906,2.4991536035867745,1.4753622541305498,1.8
|[4.004559252882169,5.510416283164524,3.7932429357843858,1.7381444954658885,7.691205600889574,2.043281936482998,1.341582614320594,1.4203
|[4.00752999416769,5.9021269507477845,3.729307053721956,1.7437393811809072,6.625337738089129,1.2533214619160256,0.5996104422048111,0.883
|[4.019412959309774,2.8954829076762674,3.772217041683318,1.7381444954658885,9.772978770421693,1.7046344123487642,0.8679269650689299,1.71
|[4.0788277850201915,5.984174455444278,4.016374873183467,1.7975124494419226,8.561765289966642,3.7332633309605696,2.685689372788546,2.095
|[4.233306331867278,5.748618716154344,4.017662172822308,1.9675348097816654,7.436093761568729,1.7876681802855232,1.6646730651823738,1.461
+------------------------------------------------------------------------------------------------------------------------------------------------
only showing top 20 rows

root
 |-- scaledFeatures: vector (nullable = true)
 |-- indexedLabel: double (nullable = false)
```

```
BootstrapModelList=[]
```

```
dt=DecisionTreeClassifier(featuresCol='scaledFeatures',labelCol='indexedLabel')
rf=RandomForestClassifier(featuresCol='scaledFeatures',labelCol='indexedLabel')
nb = NaiveBayes(featuresCol='scaledFeatures',labelCol='indexedLabel')
```

Creating three different models , i-e 1)DecesionTree 2)RandomForest 3)NaiveBsys

```
#Currently i only use three different classifier and apply on three
#different bootstrap but we can increase the classifier
dtModel1=dt.fit(BootstrapDataList[0])
nbModel1=nb.fit(BootstrapDataList[1])
rfModel1=rf.fit(BootstrapDataList[2])
```

```
print("DeceisionTree Result")
dtModel1.transform(testoneRowDF).sort('scaledFeatures').show(5)
print("NaiveBays Result")
nbModel1.transform(testoneRowDF).sort('scaledFeatures').show(5)
print("RandomForest Result")
rfModel1.transform(testoneRowDF).sort('scaledFeatures').show(5)
```

```
DeceisionTree Result
+--------------------+------------+-------------+-----------+----------+
|      scaledFeatures|indexedLabel|rawPrediction|probability|prediction|
+--------------------+------------+-------------+-----------+----------+
|[2.43481955761292...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|
|[2.56018483986190...|         1.0|    [2.0,0.0]|  [1.0,0.0]|       0.0|
|[2.68228230669681...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|
|[3.25296170764538...|         0.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|
|[3.39258654806486...|         0.0|   [56.0,0.0]|  [1.0,0.0]|       0.0|
```

```
+--------------------+------------+-------------+-----------+----------+
only showing top 5 rows

NaiveBays Result
+--------------------+------------+--------------------+--------------------+----------+
|      scaledFeatures|indexedLabel|       rawPrediction|         probability|prediction|
+--------------------+------------+--------------------+--------------------+----------+
|[2.43481955761292...|         1.0|[-204.52882644939...|[0.00949336023834...|       1.0|
|[2.56018483986190...|         1.0|[-208.78427015201...|[0.00886025708841...|       1.0|
|[2.68228230669681...|         1.0|[-385.98314345466...|[0.98890031948828...|       0.0|
|[3.25296170764538...|         0.0|[-283.14963538325...|[0.31269332469421...|       1.0|
|[3.39258654806486...|         0.0|[-427.54886016066...|[0.98760799045147...|       0.0|
+--------------------+------------+--------------------+--------------------+----------+
only showing top 5 rows

RandomForest Result
+--------------------+------------+-------------+-----------+----------+
|      scaledFeatures|indexedLabel|rawPrediction|probability|prediction|
+--------------------+------------+-------------+-----------+----------+
|[2.43481955761292...|         1.0|   [6.0,14.0]|  [0.3,0.7]|       1.0|
|[2.56018483986190...|         1.0|   [8.0,12.0]|  [0.4,0.6]|       1.0|
|[2.68228230669681...|         1.0|   [7.0,13.0]|[0.35,0.65]|       1.0|
|[3.25296170764538...|         0.0|   [16.0,4.0]|  [0.8,0.2]|       0.0|
|[3.39258654806486...|         0.0|   [15.0,5.0]|[0.75,0.25]|       0.0|
+--------------------+------------+-------------+-----------+----------+
only showing top 5 rows
```

```
predictedByDt=dtModel1.transform(testoneRowDF)
predictedByRf=rfModel1.transform(testoneRowDF)
predictedByNb=nbModel1.transform(testoneRowDF)
```

Prediction of each model

```
print("DecisionTree prediction")
predictedByDt.show()
print("Randrom Forest prediction")

predictedByRf.show()
print("Naive Bays prediction")

predictedByNb.show()
```

```
DecisionTree prediction
+--------------------+------------+-------------+-----------+----------+
|      scaledFeatures|indexedLabel|rawPrediction|probability|prediction|
+--------------------+------------+-------------+-----------+----------+
|[2.43481955761292...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|
|[2.56018483986190...|         1.0|    [2.0,0.0]|  [1.0,0.0]|       0.0|
|[2.68228230669681...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|
|[3.25296170764538...|         0.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|
|[3.39258654806486...|         0.0|   [56.0,0.0]|  [1.0,0.0]|       0.0|
|[3.49359175177257...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|
|[3.54706509491195...|         0.0|    [2.0,0.0]|  [1.0,0.0]|       0.0|
|[3.56488954262507...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|
|[3.69857290047351...|         0.0|   [56.0,0.0]|  [1.0,0.0]|       0.0|
|[3.79660736289570...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|
|[3.82037329317987...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|
|[3.86196367117716...|         0.0|   [56.0,0.0]|  [1.0,0.0]|       0.0|
|[3.91246627303102...|         0.0|   [56.0,0.0]|  [1.0,0.0]|       0.0|
|[3.94217368588623...|         0.0|    [7.0,0.0]|  [1.0,0.0]|       0.0|
|[3.96296887488487...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|
|[4.00455925288216...|         0.0|   [56.0,0.0]|  [1.0,0.0]|       0.0|
|[4.00752999416769...|         1.0|    [0.0,1.0]|  [0.0,1.0]|       1.0|
|[4.01941295930977...|         1.0|    [1.0,0.0]|  [1.0,0.0]|       0.0|
|[4.07882778502019...|         0.0|   [56.0,0.0]|  [1.0,0.0]|       0.0|
|[4.23330633186727...|         0.0|    [2.0,0.0]|  [1.0,0.0]|       0.0|
+--------------------+------------+-------------+-----------+----------+
only showing top 20 rows

Randrom Forest prediction
+--------------------+------------+-------------+-----------+----------+
|      scaledFeatures|indexedLabel|rawPrediction|probability|prediction|
+--------------------+------------+-------------+-----------+----------+
|[2.43481955761292...|         1.0|   [6.0,14.0]|  [0.3,0.7]|       1.0|
|[2.56018483986190...|         1.0|   [8.0,12.0]|  [0.4,0.6]|       1.0|
|[2.68228230669681...|         1.0|   [7.0,13.0]|[0.35,0.65]|       1.0|
|[3.25296170764538...|         0.0|   [16.0,4.0]|  [0.8,0.2]|       0.0|
|[3.39258654806486...|         0.0|   [15.0,5.0]|[0.75,0.25]|       0.0|
|[3.49359175177257...|         1.0|   [3.0,17.0]|[0.15,0.85]|       1.0|
```

```
|[3.54706509491195...|         0.0|   [3.0,17.0]|[0.15,0.85]|       1.0|
|[3.56488954262507...|         1.0|   [4.0,16.0]|  [0.2,0.8]|       1.0|
|[3.69857290047351...|         0.0|   [20.0,0.0]|  [1.0,0.0]|       0.0|
|[3.79660736289570...|         1.0|   [3.0,17.0]|[0.15,0.85]|       1.0|
|[3.82037329317987...|         1.0|   [4.0,16.0]|  [0.2,0.8]|       1.0|
|[3.86196367117716...|         0.0|   [20.0,0.0]|  [1.0,0.0]|       0.0|
|[3.91246627303102...|         0.0|   [19.0,1.0]|[0.95,0.05]|       0.0|
|[3.94217368588623...|         0.0|   [13.0,7.0]|[0.65,0.35]|       0.0|
|[3.96296887488487...|         1.0|   [14.0,6.0]|  [0.7,0.3]|       0.0|
|[4.00455925288216...|         0.0|   [20.0,0.0]|  [1.0,0.0]|       0.0|
|[4.00752999416769...|         1.0|   [5.0,15.0]|[0.25,0.75]|       1.0|
|[4.01941295930977...|         1.0|   [2.0,18.0]|  [0.1,0.9]|       1.0|
|[4.07882778502019...|         0.0|   [20.0,0.0]|  [1.0,0.0]|       0.0|
|[4.23330633186727...|         0.0|   [18.0,2.0]|  [0.9,0.1]|       0.0|
+--------------------+------------+-------------+-----------+----------+
only showing top 20 rows

Naive Bays prediction
+--------------------+------------+--------------------+--------------------+----------+
|      scaledFeatures|indexedLabel|       rawPrediction|         probability|prediction|
```

```
testoneRowDF.show()
```

```
+--------------------+------------+
|      scaledFeatures|indexedLabel|
+--------------------+------------+
|[2.43481955761292...|         1.0|
|[2.56018483986190...|         1.0|
|[2.68228230669681...|         1.0|
|[3.25296170764538...|         0.0|
|[3.39258654806486...|         0.0|
|[3.49359175177257...|         1.0|
|[3.54706509491195...|         0.0|
|[3.56488954262507...|         1.0|
|[3.69857290047351...|         0.0|
|[3.79660736289570...|         1.0|
|[3.82037329317987...|         1.0|
|[3.86196367117716...|         0.0|
|[3.91246627303102...|         0.0|
|[3.94217368588623...|         0.0|
|[3.96296887488487...|         1.0|
|[4.00455925288216...|         0.0|
|[4.00752999416769...|         1.0|
|[4.01941295930977...|         1.0|
|[4.07882778502019...|         0.0|
|[4.23330633186727...|         0.0|
+--------------------+------------+
only showing top 20 rows
```

```
from pyspark.sql.functions import lit
```

```
predictedByDt1=predictedByDt.withColumn("pred_DT",predictedByDt.prediction)
```

```
predictedByDt1.show()
```

```
+--------------------+------------+-------------+-----------+----------+-------+
|      scaledFeatures|indexedLabel|rawPrediction|probability|prediction|pred_DT|
+--------------------+------------+-------------+-----------+----------+-------+
|[2.43481955761292...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|    1.0|
|[2.56018483986190...|         1.0|    [2.0,0.0]|  [1.0,0.0]|       0.0|    0.0|
|[2.68228230669681...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|    1.0|
|[3.25296170764538...|         0.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|    1.0|
|[3.39258654806486...|         0.0|   [56.0,0.0]|  [1.0,0.0]|       0.0|    0.0|
|[3.49359175177257...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|    1.0|
|[3.54706509491195...|         0.0|    [2.0,0.0]|  [1.0,0.0]|       0.0|    0.0|
|[3.56488954262507...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|    1.0|
|[3.69857290047351...|         0.0|   [56.0,0.0]|  [1.0,0.0]|       0.0|    0.0|
|[3.79660736289570...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|    1.0|
|[3.82037329317987...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|    1.0|
|[3.86196367117716...|         0.0|   [56.0,0.0]|  [1.0,0.0]|       0.0|    0.0|
|[3.91246627303102...|         0.0|   [56.0,0.0]|  [1.0,0.0]|       0.0|    0.0|
|[3.94217368588623...|         0.0|    [7.0,0.0]|  [1.0,0.0]|       0.0|    0.0|
|[3.96296887488487...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|    1.0|
|[4.00455925288216...|         0.0|   [56.0,0.0]|  [1.0,0.0]|       0.0|    0.0|
|[4.00752999416769...|         1.0|    [0.0,1.0]|  [0.0,1.0]|       1.0|    1.0|
|[4.01941295930977...|         1.0|    [1.0,0.0]|  [1.0,0.0]|       0.0|    0.0|
|[4.07882778502019...|         0.0|   [56.0,0.0]|  [1.0,0.0]|       0.0|    0.0|
|[4.23330633186727...|         0.0|    [2.0,0.0]|  [1.0,0.0]|       0.0|    0.0|
```

```
+--------------------+------------+-------------+-----------+----------+-------+
only showing top 20 rows
```

```
predictedByRf1=predictedByRf.withColumn("pred_RF",predictedByRf.prediction).select("pred_RF")
```

```
predictedByRf1.show()
```

```
+-------+
|pred_RF|
+-------+
|    1.0|
|    1.0|
|    1.0|
|    0.0|
|    0.0|
|    1.0|
|    1.0|
|    1.0|
|    0.0|
|    1.0|
|    1.0|
|    0.0|
|    0.0|
|    0.0|
|    0.0|
|    0.0|
|    1.0|
|    1.0|
|    0.0|
|    0.0|
+-------+
only showing top 20 rows
```

```
predictedByRf1=predictedByRf1.join(predictedByDt1)
predictedByRf1.show()
```

```
+-------+--------------------+------------+-------------+-----------+----------+-------+
|pred_RF|      scaledFeatures|indexedLabel|rawPrediction|probability|prediction|pred_DT|
+-------+--------------------+------------+-------------+-----------+----------+-------+
|    1.0|[2.43481955761292...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|    1.0|
|    1.0|[2.43481955761292...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|    1.0|
|    1.0|[2.43481955761292...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|    1.0|
|    0.0|[2.43481955761292...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|    1.0|
|    0.0|[2.43481955761292...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|    1.0|
|    1.0|[2.43481955761292...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|    1.0|
|    1.0|[2.43481955761292...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|    1.0|
|    1.0|[2.43481955761292...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|    1.0|
|    0.0|[2.43481955761292...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|    1.0|
|    1.0|[2.43481955761292...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|    1.0|
|    1.0|[2.43481955761292...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|    1.0|
|    0.0|[2.43481955761292...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|    1.0|
|    0.0|[2.43481955761292...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|    1.0|
|    0.0|[2.43481955761292...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|    1.0|
|    0.0|[2.43481955761292...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|    1.0|
|    1.0|[2.43481955761292...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|    1.0|
|    1.0|[2.43481955761292...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|    1.0|
|    0.0|[2.43481955761292...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|    1.0|
|    0.0|[2.43481955761292...|         1.0|   [0.0,21.0]|  [0.0,1.0]|       1.0|    1.0|
+-------+--------------------+------------+-------------+-----------+----------+-------+
only showing top 20 rows
```

```
predictedByRf1=predictedByRf1.select(predictedByRf1[0],predictedByRf1[1],predictedByRf1[2],predictedByRf1[6])
```

```
predictedByNb1=predictedByNb.withColumn("pred_NB",predictedByNb.prediction)
```

```
predictedByNb1=predictedByNb1.select("pred_NB")
```

```
predictedByNb1.show()
```

```
+-------+
|pred_NB|
+-------+
|    1.0|
```

```
|    1.0|
|    0.0|
|    1.0|
|    0.0|
|    1.0|
|    1.0|
|    1.0|
|    0.0|
|    1.0|
|    1.0|
|    0.0|
|    0.0|
|    0.0|
|    0.0|
|    0.0|
|    1.0|
|    1.0|
|    0.0|
|    0.0|
+-------+
only showing top 20 rows
```

```
predictedByNb1=predictedByNb1.join(predictedByRf1)
```

```
predictedByNb1.show()
```

```
+-------+-------+--------------------+------------+-------+
|pred_NB|pred_RF|      scaledFeatures|indexedLabel|pred_DT|
+-------+-------+--------------------+------------+-------+
|    1.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    1.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    0.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    1.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    0.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    1.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    1.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    1.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    0.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    1.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    1.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    0.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    0.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    0.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    0.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    0.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    1.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    1.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    0.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    0.0|    1.0|[2.43481955761292...|         1.0|    1.0|
+-------+-------+--------------------+------------+-------+
only showing top 20 rows
```

```
finalPredictedDF=predictedByNb1
```

```
finalPredictedDF.show()
```

```
+-------+-------+--------------------+------------+-------+
|pred_NB|pred_RF|      scaledFeatures|indexedLabel|pred_DT|
+-------+-------+--------------------+------------+-------+
|    1.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    1.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    0.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    1.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    0.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    1.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    1.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    1.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    0.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    1.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    1.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    0.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    0.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    0.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    0.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    0.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    1.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    1.0|    1.0|[2.43481955761292...|         1.0|    1.0|
|    0.0|    1.0|[2.43481955761292...|         1.0|    1.0|
```

```
|    0.0|    1.0|[2.43481955761292...|         1.0|    1.0|
+-------+-------+--------------------+------------+-------+
only showing top 20 rows
```

```
import pyspark.sql.functions as F
```

*Below is the condition for voting i-e if 2 of 3 classifier gives 1 result, then the votedLabel will be 1 otherwise 0 * bold text

```
#Below is the condition for voting i-e if 2 of 3 classifier
#gives 1 result, then the votedLabel will be 1 otherwise 0
finalVotedDF=finalPredictedDF.withColumn('votedLabel', F.when(((F.col("pred_NB") == 1) & (F.col("pred_RF") == 1)) |
                                            ((F.col("pred_NB") == 1) & (F.col("pred_DT") == 1)) |
                                            ((F.col("pred_DT") == 1) & (F.col("pred_RF") == 1))
                                            , 1).otherwise(0))
```

```
# indexedLabel shows actualLabel
# pred_NB shows prediction by Naive Bays
# pred_DT shows prediction by DecisionTree
# pred_RF shows prediction by RandomForest
```

```
finalVotedDF.select("scaledFeatures","indexedLabel","pred_NB","pred_DT","pred_RF","votedLabel").show(150)
```

```
+--------------------+------------+-------+-------+-------+----------+
|      scaledFeatures|indexedLabel|pred_NB|pred_DT|pred_RF|votedLabel|
+--------------------+------------+-------+-------+-------+----------+
|[2.43481955761292...|         1.0|    1.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    1.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    1.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    1.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    1.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    1.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    1.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    1.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    1.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    1.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    1.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    1.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    1.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    1.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    1.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    1.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    1.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    1.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
```

```
|[2.43481955761292...|         1.0|    1.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    1.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
|[2.43481955761292...|         1.0|    0.0|    1.0|    1.0|         1|
```

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator


from pyspark.sql.types import DoubleType
finalVotedDF = finalVotedDF.withColumn("votedLabel", finalVotedDF.votedLabel.cast(DoubleType()))


votedevaluator = MulticlassClassificationEvaluator(
    labelCol="indexedLabel", predictionCol="votedLabel", metricName="accuracy")
votedaccuracy = votedevaluator.evaluate(finalVotedDF)
print("voted Accuracy: ",accuracy)
```

```
voted Accuracy:  0.760932944606414
```

```
NBevaluator = MulticlassClassificationEvaluator(
    labelCol="indexedLabel", predictionCol="pred_NB", metricName="accuracy")
NBaccuracy = NBevaluator.evaluate(finalVotedDF)
print("Naive Bays Accuracy: ",NBaccuracy)
```

```
Naive Bays Accuracy:  0.5918367346938775
```

```
# so voted label accuracy is better than Naive bays accuracy
```