```
!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 4.2 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 20.6 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=281824028 sha256=a13aca676936528aa26bc4bd3baad454c1f1204ec
  Stored in directory: /root/.cache/pip/wheels/6c/e3/9b/0525ce8a69478916513509d43693511463c6468db0de237c86
Successfully built pyspark
Installing collected packages: py4j, pyspark
  Attempting uninstall: py4j
    Found existing installation: py4j 0.10.9.7
    Uninstalling py4j-0.10.9.7:
      Successfully uninstalled py4j-0.10.9.7
Successfully installed py4j-0.10.9.5 pyspark-3.3.2
```

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.master("local[*]").getOrCreate()
```

```
'''
load models
'''
from pyspark.ml import Pipeline
from pyspark.ml.feature import StringIndexer, VectorIndexer, IndexToString
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.mllib.evaluation import MulticlassMetrics
```

```
from google.colab import drive

# Mount Google Drive to this Notebook instance
drive.mount('/content/drive')
```

```
nt/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Saved successfully!

```
data = spark.read.format("libsvm").load("/content/drive/My Drive/dataset.txt")
```

```
data.select("features").show(1,False)
```

```
+-------------------------------------------------+
|features                                         |
+-------------------------------------------------+
|(4,[0,1,2,3],[-0.222222,0.5,-0.762712,-0.833333])|
+-------------------------------------------------+
only showing top 1 row
```

```
labelIndexer = StringIndexer(inputCol="label", outputCol="indexedLabel").fit(data)
```

```
featureIndexer =VectorIndexer(inputCol="features", outputCol="indexedFeatures", maxCategories=4).fit(data)
```

```
(trainingData, testData) = data.randomSplit([0.7, 0.3])
```

```
from pyspark.ml.classification import RandomForestClassifier
for n in (7,5):
  rf = RandomForestClassifier(numTrees=n,featuresCol="indexedFeatures",labelCol="indexedLabel")
  rf_pipeline = Pipeline(stages=[labelIndexer, featureIndexer, rf])
 ·rf_model·=·rf_pipeline.fit(trainingData)
  rf_predictions = rf_model.transform(testData)
  print(rf_model.stages[2])
  rf_predictions.show(5)
```

```
RandomForestClassificationModel: uid=RandomForestClassifier_6334c0ec4696, numTrees=7, numClasses=3, numFeatures=4
+-----+--------------------+------------+--------------------+--------------------+--------------------+----------+
|label|            features|indexedLabel|     indexedFeatures|       rawPrediction|         probability|prediction|
+-----+--------------------+------------+--------------------+--------------------+--------------------+----------+
|  0.0|(4,[0,1,2,3],[-0....|         0.0|(4,[0,1,2,3],[-0....|[5.83333333333333...|[0.83333333333333...|       0.0|
|  0.0|(4,[0,1,2,3],[-0....|         0.0|(4,[0,1,2,3],[-0....|[5.83333333333333...|[0.83333333333333...|       0.0|
|  0.0|(4,[0,1,2,3],[-0....|         0.0|(4,[0,1,2,3],[-0....|       [7.0,0.0,0.0]|       [1.0,0.0,0.0]|       0.0|
|  0.0|(4,[0,1,2,3],[0.1...|         0.0|(4,[0,1,2,3],[0.1...|[6.83333333333333...|[0.97619047619047...|       0.0|
|  0.0|(4,[0,1,2,3],[0.1...|         0.0|(4,[0,1,2,3],[0.1...|       [7.0,0.0,0.0]|       [1.0,0.0,0.0]|       0.0|
+-----+--------------------+------------+--------------------+--------------------+--------------------+----------+
only showing top 5 rows

RandomForestClassificationModel: uid=RandomForestClassifier_f991bb141c24, numTrees=5, numClasses=3, numFeatures=4
+-----+--------------------+------------+--------------------+--------------------+--------------------+----------+
|label|            features|indexedLabel|     indexedFeatures|       rawPrediction|         probability|prediction|
+-----+--------------------+------------+--------------------+--------------------+--------------------+----------+
|  0.0|(4,[0,1,2,3],[-0....|         0.0|(4,[0,1,2,3],[-0....|       [5.0,0.0,0.0]|       [1.0,0.0,0.0]|       0.0|
|  0.0|(4,[0,1,2,3],[-0....|         0.0|(4,[0,1,2,3],[-0....|       [5.0,0.0,0.0]|       [1.0,0.0,0.0]|       0.0|
|  0.0|(4,[0,1,2,3],[-0....|         0.0|(4,[0,1,2,3],[-0....|       [5.0,0.0,0.0]|       [1.0,0.0,0.0]|       0.0|
|  0.0|(4,[0,1,2,3],[0.1...|         0.0|(4,[0,1,2,3],[0.1...|[4.94736842105263...|[0.98947368421052...|       0.0|
|  0.0|(4,[0,1,2,3],[0.1...|         0.0|(4,[0,1,2,3],[0.1...|[4.94736842105263...|[0.98947368421052...|       0.0|
+-----+--------------------+------------+--------------------+--------------------+--------------------+----------+
only showing top 5 rows
```

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.sql.functions import col

# calculate accuracy
accuracy_evaluator = MulticlassClassificationEvaluator(labelCol="indexedLabel", predictionCol="prediction", metricName="accuracy")
accuracy = accuracy_evaluator.evaluate(rf_predictions)

# calculate precision
precision_evaluator = MulticlassClassificationEvaluator(labelCol="indexedLabel", predictionCol="prediction", metricName="weightedPrecision")
precision = precision_evaluator.evaluate(rf_predictions)

# calculate recall
recall_evaluator = MulticlassClassificationEvaluator(labelCol="indexedLabel", predictionCol="prediction", metricName="weightedRecall")
recall = recall_evaluator.evaluate(rf_predictions)

# calculate F1 score
f1_evaluator = MulticlassClassificationEvaluator(labelCol="indexedLabel", predictionCol="prediction", metricName="f1")
f1_score = f1_evaluator.evaluate(rf_predictions)
```



```
print("Recall: ", recall)
print("F1 score: ", f1_score)
print("F1 score: ", f1_score)
```

```
Accuracy:  0.9629629629629629
Precision:  0.9673202614379085
Recall:  0.9629629629629629
F1 score:  0.9628858024691358
F1 score:  0.9628858024691358
```

```
# create a confusion matrix
predictionsAndLabels = rf_predictions.select("prediction", "indexedLabel").rdd
metrics = MulticlassMetrics(predictionsAndLabels)
confusion_matrix = metrics.confusionMatrix().toArray()

# print the confusion matrix
print("Confusion matrix:")
print(confusion_matrix)

# calculate precision, recall, and F1-score
tp = confusion_matrix[1, 1]
fp = confusion_matrix[0, 1]
tn = confusion_matrix[0, 0]
fn = confusion_matrix[1, 0]

precision = tp / (tp + fp)
recall = tp / (tp + fn)
f1_score = 2 * (precision * recall) / (precision + recall)
accuracy = (tp + tn) / (tp + fp + tn + fn)

# print the evaluation metrics
```

```
print("Precision: ", precision)
print("Recall: ", recall)
print("F1 score: ", f1_score)
print("Accuracy: ", accuracy)
```

```
/usr/local/lib/python3.9/dist-packages/pyspark/sql/context.py:157: FutureWarning: Deprecated in 3.0.0. Use SparkSession.builder.getOrCre
  warnings.warn(
Confusion matrix:
[[15.  0.  0.]
 [ 0. 23.  0.]
 [ 2.  0. 14.]]
Precision:  1.0
Recall:  1.0
F1 score:  1.0
Accuracy:  1.0
```

0s completed at 2:50 PM

Saved successfully!