Big Data Analytics – Spark ML Classification Assignment

NOTES:

1. Please read readme file provided.

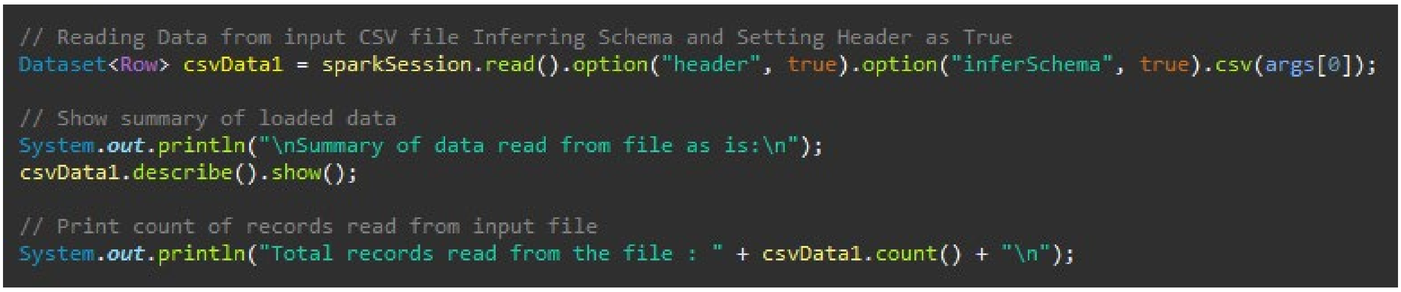
2. I have submitted a separate program to show data pre-processing steps namely **SparkMLClassificationAssignmentDataPreProcessing.java**. Screenshots during data processing steps, are taken from the output of this program.

3. I have submitted a separate program to show testing on various models and tuning of hyper parameters being used to create models for Decision Tree and Random Forest, namely **SparkMLClassificationAssignmentModelTesting.java**. Screenshots during testing of various models along with accuracy, are taken from the output of this program.

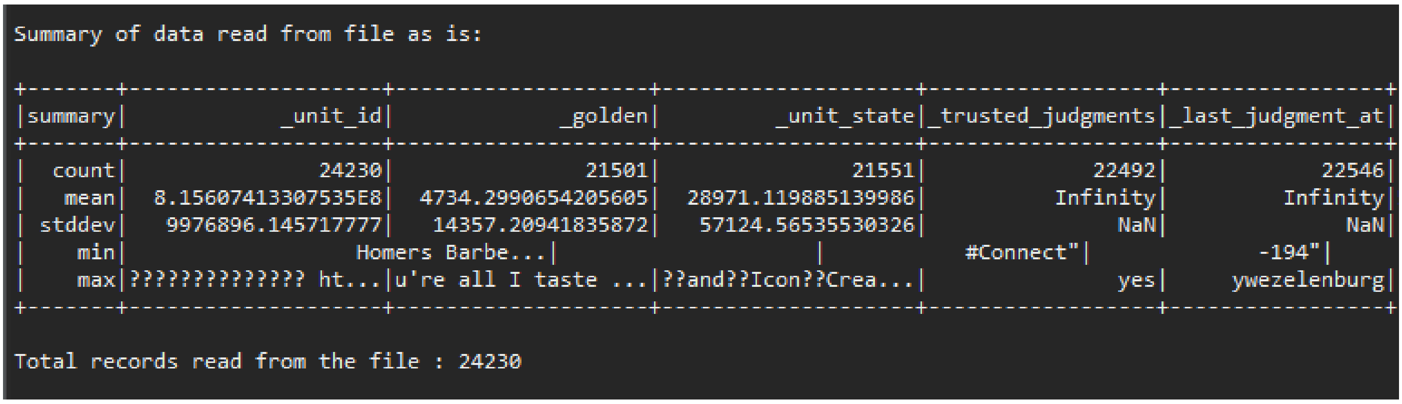
4. Main program name is **SparkMLClassificationAssignment.java** which builds final model on finally chosen hyper parameters for Decision Tree and Random Forest. And this program does not show any data during data pre-processing steps. This program displays mainly performance metrics for both the models. Screenshots during performance metrics evaluation of both final models, are taken from the output of this program.

1. Data Processing Steps:

a. There are total 20050 records in the provided input data. But when we read the file in Spark and try to describe().show(), we get more count. Below are the screenshots of code and output. There are some corrupted records in the input file.

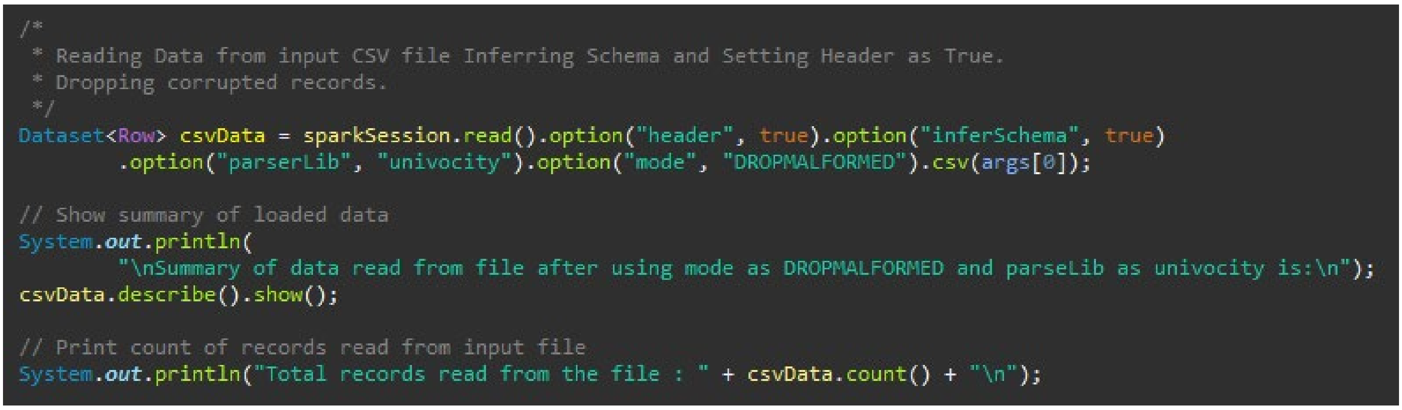


```java
// Reading Data from input CSV file Inferring Schema and Setting Header as True
Dataset<Row> csvData1 = sparkSession.read().option("header", true).option("inferSchema", true).csv(args[0]);

// Show summary of loaded data
System.out.println("\nSummary of data read from file as is:\n");
csvData1.describe().show();

// Print count of records read from input file
System.out.println("Total records read from the file : " + csvData1.count() + "\n");
```

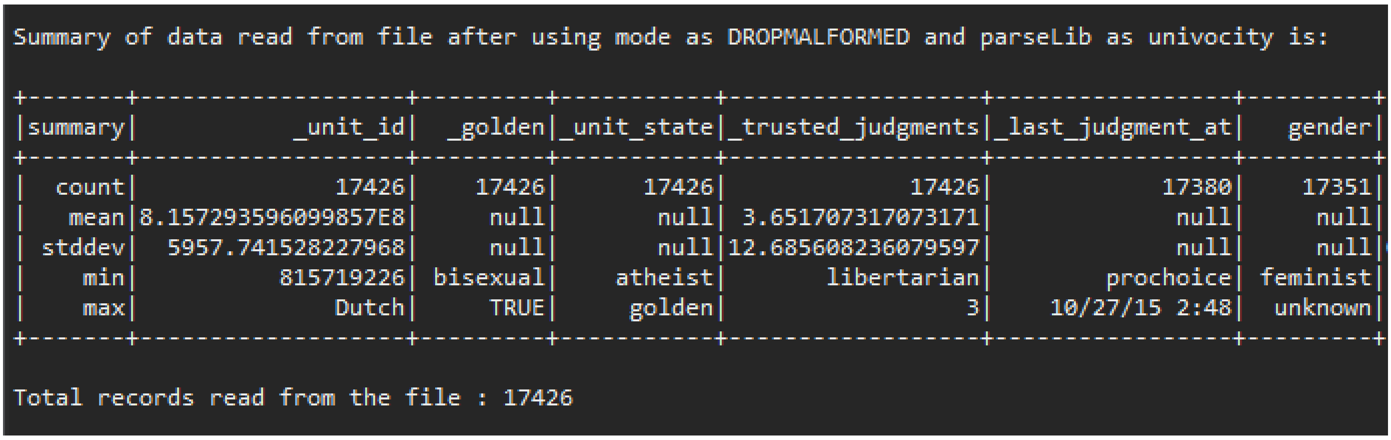


```
Summary of data read from file as is:

+-------+--------------------+--------------------+--------------------+------------------+-----------------+
|summary|            _unit_id|             _golden|         _unit_state|_trusted_judgments|_last_judgment_at|
+-------+--------------------+--------------------+--------------------+------------------+-----------------+
|  count|               24230|               21501|               21551|             22492|            22546|
|   mean|  8.15607413307535E8|  4734.2990654205605|  28971.119885139986|          Infinity|         Infinity|
| stddev|   9976896.145717777|   14357.20941835872|   57124.56535530326|               NaN|              NaN|
|    min|                  Homers Barbe...|                    |           #Connect"|             -194"|
|    max|?????????????? ht...|u're all I taste ...|??and??Icon??Crea...|               yes|     ywezelenburg|
+-------+--------------------+--------------------+--------------------+------------------+-----------------+

Total records read from the file : 24230
```

In order to remove corrupted records, using **option("mode", "DROPMALFORMED")** and also using **option("parseLib", "univocity")** as well to handle multiline data. Below are the screenshots of code and output.

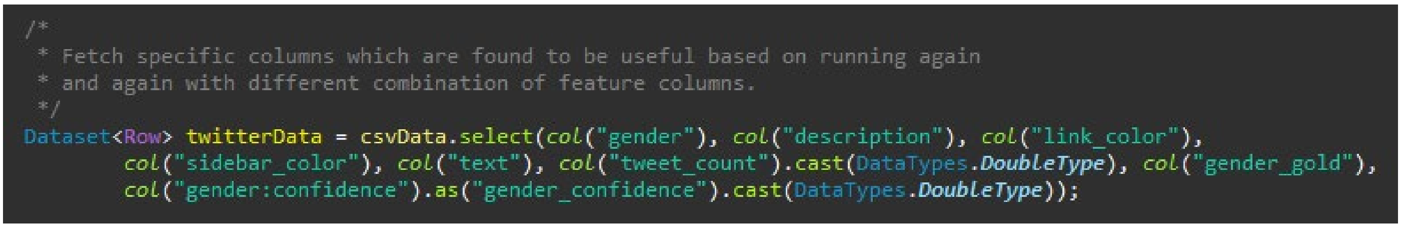


```
/*
 * Reading Data from input CSV file Inferring Schema and Setting Header as True.
 * Dropping corrupted records.
 */
Dataset<Row> csvData = sparkSession.read().option("header", true).option("inferSchema", true)
        .option("parserLib", "univocity").option("mode", "DROPMALFORMED").csv(args[0]);

// Show summary of loaded data
System.out.println(
        "\nSummary of data read from file after using mode as DROPMALFORMED and parseLib as univocity is:\n");
csvData.describe().show();

// Print count of records read from input file
System.out.println("Total records read from the file : " + csvData.count() + "\n");
```



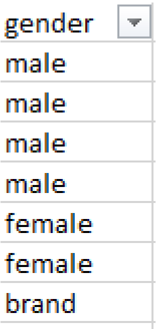
```
Summary of data read from file after using mode as DROPMALFORMED and parseLib as univocity is:

+-------+-------------------+---------+-----------+------------------+-----------------+--------+
|summary|           _unit_id|  _golden|_unit_state|_trusted_judgments|_last_judgment_at|  gender|
+-------+-------------------+---------+-----------+------------------+-----------------+--------+
|  count|              17426|    17426|      17426|             17426|            17380|   17351|
|   mean|8.157293596099857E8|     null|       null| 3.651707317073171|             null|    null|
| stddev|  5957.741528227968|     null|       null|12.685608236079597|             null|    null|
|    min|          815719226| bisexual|    atheist|       libertarian|        prochoice|feminist|
|    max|              Dutch|     TRUE|     golden|                 3|    10/27/15 2:48| unknown|
+-------+-------------------+---------+-----------+------------------+-----------------+--------+

Total records read from the file : 17426
```

b. After testing various models and various fields, selecting only few fields namely gender, description, link_color, sidebar_color, text, tweet_count (casting as double), gender_gold and gender:confidence as gender_confidence (casting as double) which are helping in achieving a better model, in final program.



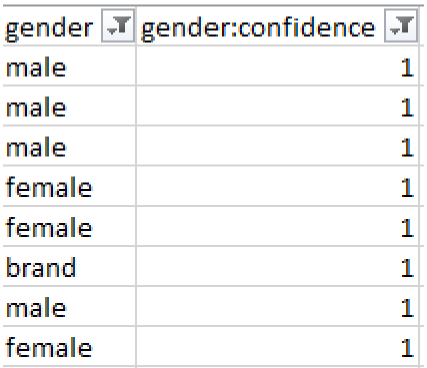
```
/*
 * Fetch specific columns which are found to be useful based on running again
 * and again with different combination of feature columns.
 */
Dataset<Row> twitterData = csvData.select(col("gender"), col("description"), col("link_color"),
        col("sidebar_color"), col("text"), col("tweet_count").cast(DataTypes.DoubleType), col("gender_gold"),
        col("gender:confidence").as("gender_confidence").cast(DataTypes.DoubleType));
```

c. We should use only those records where gender is present as male/female/brand so accordingly we can try to predict and compare prediction against original data and find accuracy of our model.



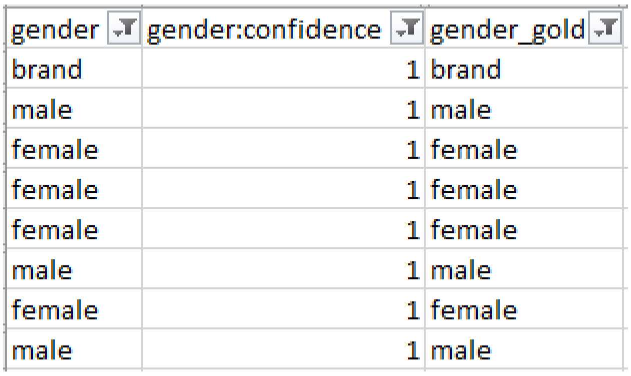
gender
male
male
male
male
female
female
brand

d. During data analysis, it has been observed, when gender_confidence is 1.0, gender is clearly defined as male/female/brand so picking only those records where gender_confidence as 1.0. This helps in achieving better accuracy by the model.



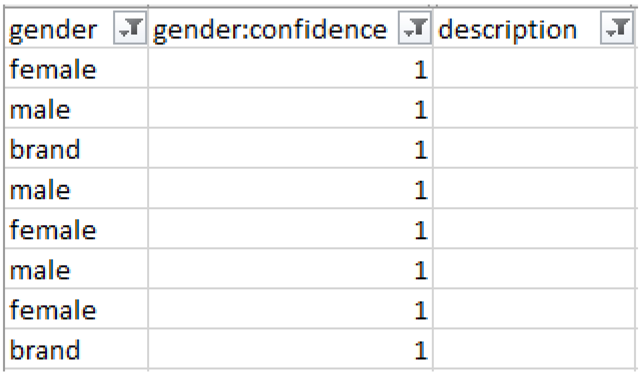
gender	gender:confidence
male	1
male	1
male	1
female	1
female	1
brand	1
male	1
female	1

e. There is another field gender_gold. Including those records as well where gender_gold is present as male/female/brand. There are 50 such records. When gender_gold is present as male/female/brand then gender field has same values.



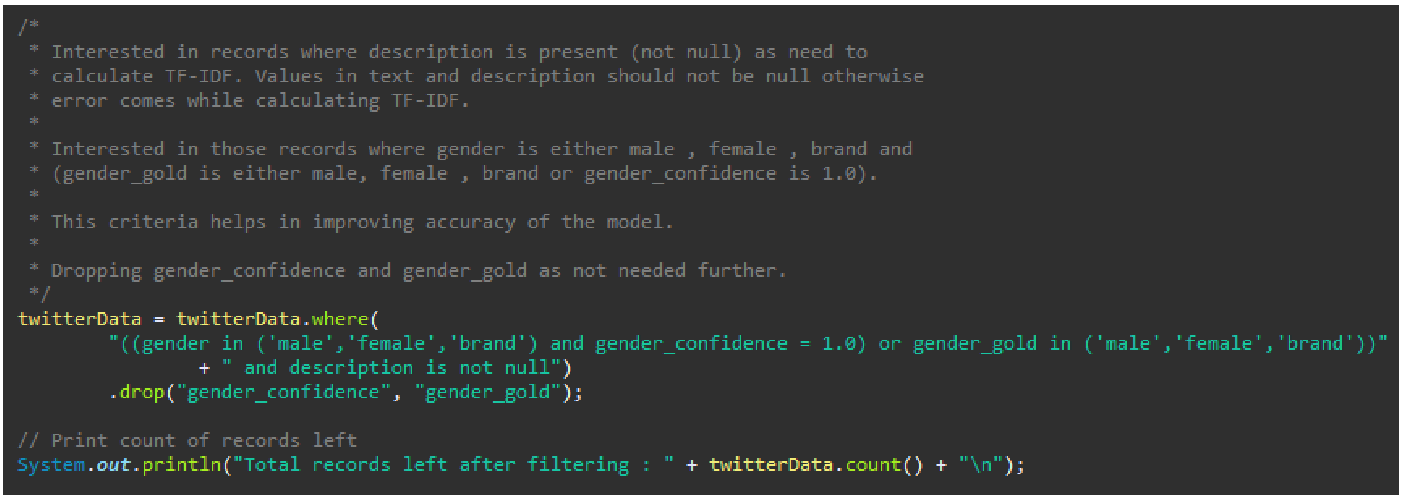
gender	gender:confidence	gender_gold
brand	1	brand
male	1	male
female	1	female
female	1	female
female	1	female
male	1	male
female	1	female
male	1	male

f. Need to pass text and description fields through TF-IDF computation so should not be null. Field text is not null already. Need to pick only those records where description field is not null otherwise TF-IDF computation fails. We could have filled description field with some values but then that reduces the accuracy. Choosing only those records where description field is present, helps in achieving better accuracy by the model.

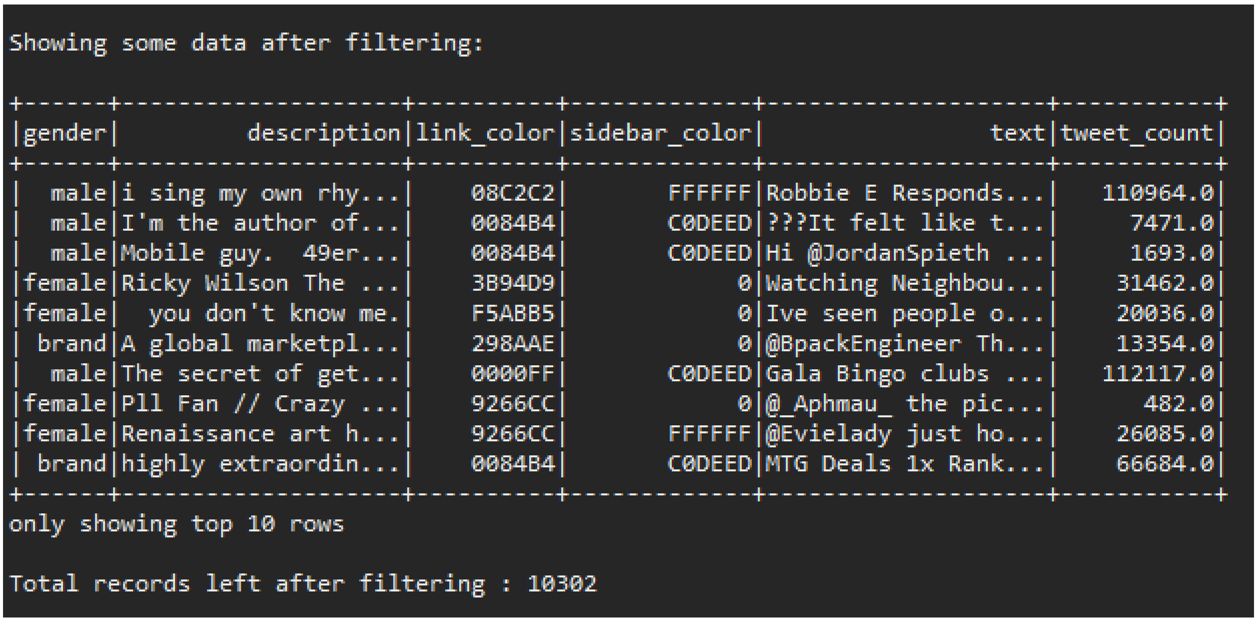


gender	gender:confidence	description
female	1	
male	1	
brand	1	
male	1	
female	1	
male	1	
female	1	
brand	1	

g. Post selection of right set of records, dropping gender_confidence and gender_gold fields as not needed further so dropping them. Below are the screenshots of code and filtered data.

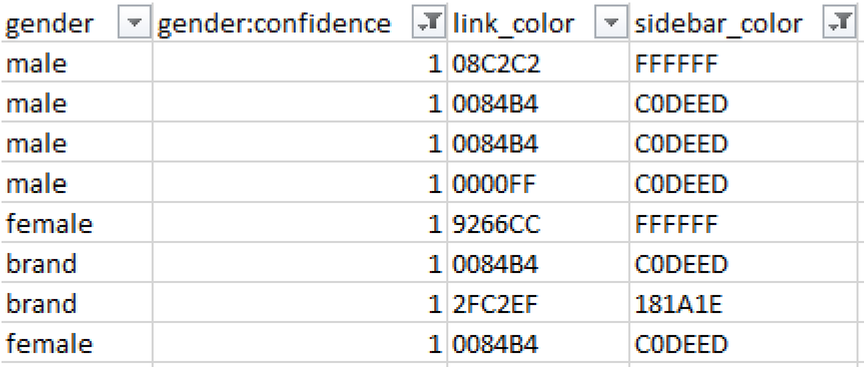


```
/*
 * Interested in records where description is present (not null) as need to
 * calculate TF-IDF. Values in text and description should not be null otherwise
 * error comes while calculating TF-IDF.
 *
 * Interested in those records where gender is either male , female , brand and
 * (gender_gold is either male, female , brand or gender_confidence is 1.0).
 *
 * This criteria helps in improving accuracy of the model.
 *
 * Dropping gender_confidence and gender_gold as not needed further.
 */
twitterData = twitterData.where(
        "((gender in ('male','female','brand') and gender_confidence = 1.0) or gender_gold in ('male','female','brand'))"
            + " and description is not null")
        .drop("gender_confidence", "gender_gold");

// Print count of records left
System.out.println("Total records left after filtering : " + twitterData.count() + "\n");
```

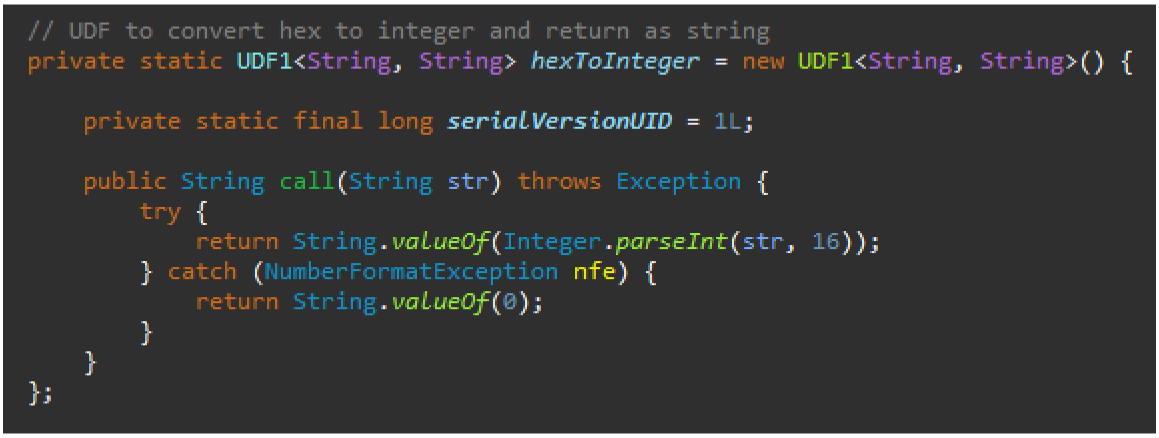


```
Showing some data after filtering:

+------+--------------------+----------+-------------+--------------------+-----------+
|gender|         description|link_color|sidebar_color|                text|tweet_count|
+------+--------------------+----------+-------------+--------------------+-----------+
|  male|i sing my own rhy...|    08C2C2|       FFFFFF|Robbie E Responds...|   110964.0|
|  male|I'm the author of...|    0084B4|       C0DEED|???It felt like t...|     7471.0|
|  male|Mobile guy.  49er...|    0084B4|       C0DEED|Hi @JordanSpieth ...|     1693.0|
|female|Ricky Wilson The ...|    3B94D9|            0|Watching Neighbou...|    31462.0|
|female|  you don't know me.|    F5ABB5|            0|Ive seen people o...|    20036.0|
| brand|A global marketpl...|    298AAE|            0|@BpackEngineer Th...|    13354.0|
|  male|The secret of get...|    0000FF|       C0DEED|Gala Bingo clubs ...|   112117.0|
|female|Pll Fan // Crazy ...|    9266CC|            0|@_Aphmau_ the pic...|      482.0|
|female|Renaissance art h...|    9266CC|       FFFFFF|@Evielady just ho...|    26085.0|
| brand|highly extraordin...|    0084B4|       C0DEED|MTG Deals 1x Rank...|    66684.0|
+------+--------------------+----------+-------------+--------------------+-----------+
only showing top 10 rows

Total records left after filtering : 10302
```

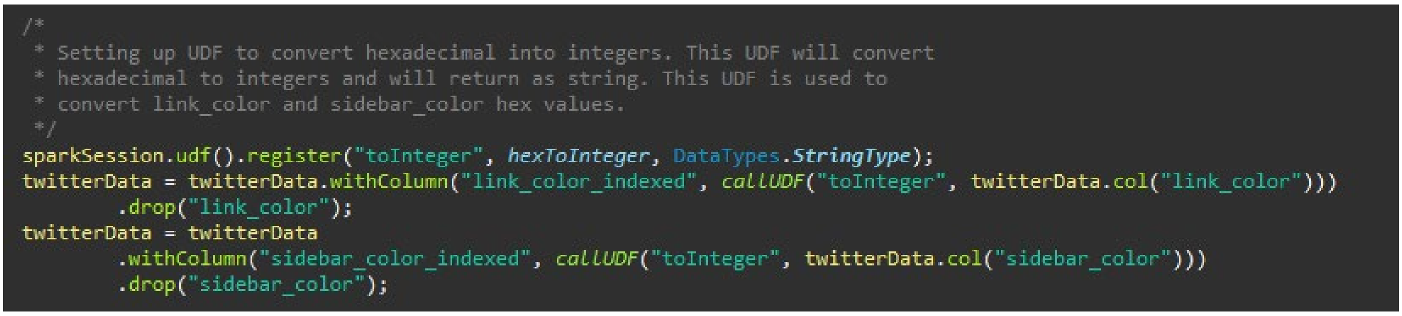
h. Fields link_color and sidebar_color are in hexadecimal. We need to convert these values into integers so can be used in the model. I have setup a UDF for the same. Below is data in hexadecimal format as provided in input file, then screenshot of UDF, then screenshot of how UDF is being used and then screenshot of data obtained after use of UDF.



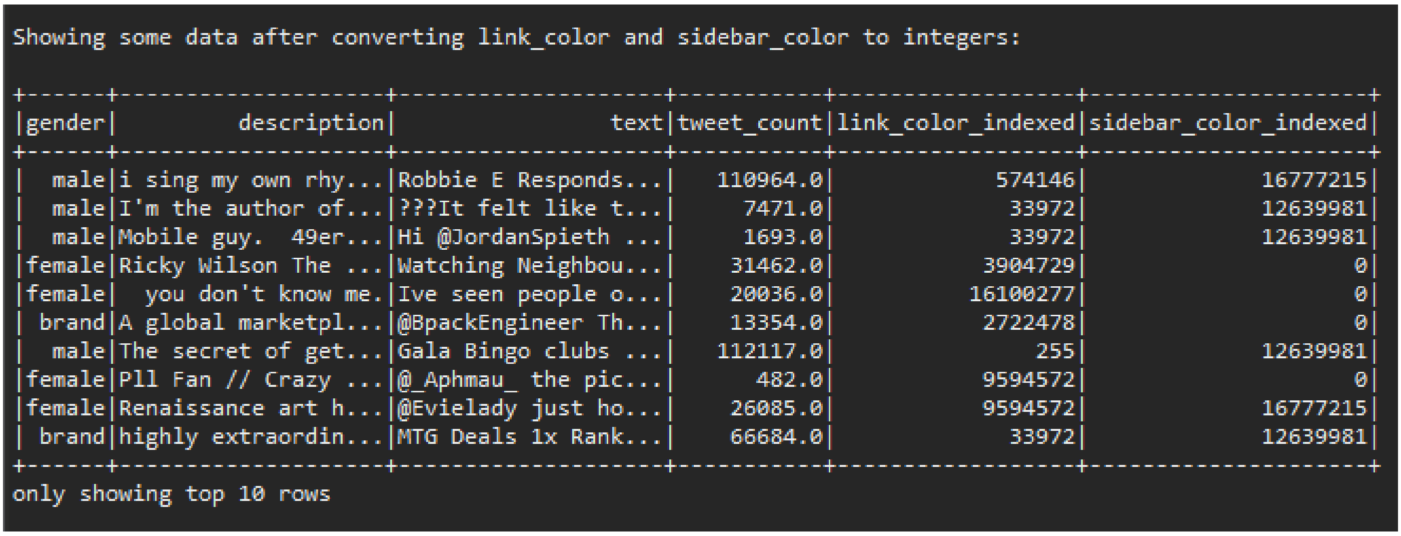
gender	gender:confidence	link_color	sidebar_color
male	1	08C2C2	FFFFFF
male	1	0084B4	C0DEED
male	1	0084B4	C0DEED
male	1	0000FF	C0DEED
female	1	9266CC	FFFFFF
brand	1	0084B4	C0DEED
brand	1	2FC2EF	181A1E
female	1	0084B4	C0DEED



```
// UDF to convert hex to integer and return as string
private static UDF1<String, String> hexToInteger = new UDF1<String, String>() {

    private static final long serialVersionUID = 1L;

    public String call(String str) throws Exception {
        try {
            return String.valueOf(Integer.parseInt(str, 16));
        } catch (NumberFormatException nfe) {
            return String.valueOf(0);
        }
    }
};
```

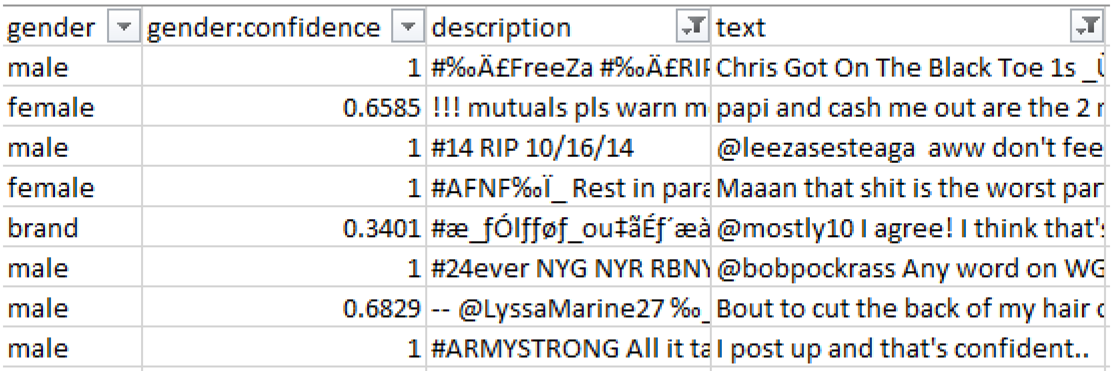


```
/*
 * Setting up UDF to convert hexadecimal into integers. This UDF will convert
 * hexadecimal to integers and will return as string. This UDF is used to
 * convert link_color and sidebar_color hex values.
 */
sparkSession.udf().register("toInteger", hexToInteger, DataTypes.StringType);
twitterData = twitterData.withColumn("link_color_indexed", callUDF("toInteger", twitterData.col("link_color")))
        .drop("link_color");
twitterData = twitterData
        .withColumn("sidebar_color_indexed", callUDF("toInteger", twitterData.col("sidebar_color")))
        .drop("sidebar_color");
```

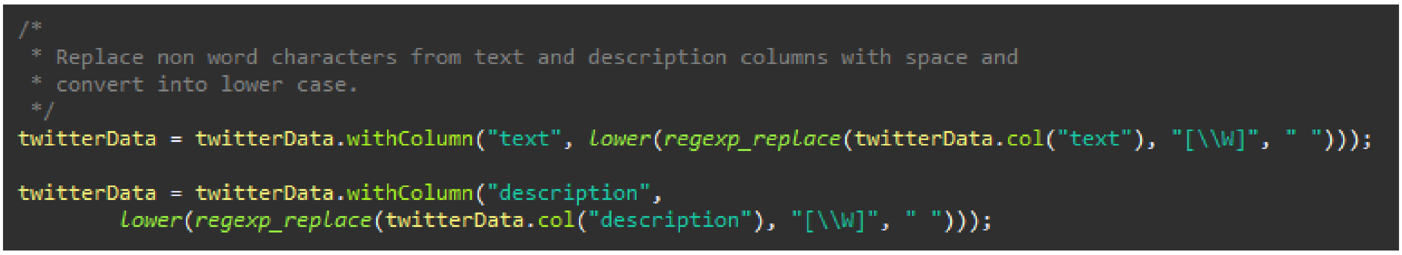


```
Showing some data after converting link_color and sidebar_color to integers:

+------+--------------------+--------------------+-----------+------------------+---------------------+
|gender|         description|                text|tweet_count|link_color_indexed|sidebar_color_indexed|
+------+--------------------+--------------------+-----------+------------------+---------------------+
|  male|i sing my own rhy...|Robbie E Responds...|   110964.0|            574146|             16777215|
|  male|I'm the author of...|???It felt like t...|     7471.0|             33972|             12639981|
|  male|Mobile guy.  49er...|Hi @JordanSpieth ...|     1693.0|             33972|             12639981|
|female|Ricky Wilson The ...|Watching Neighbou...|    31462.0|           3904729|                    0|
|female|   you don't know me.|Ive seen people o...|    20036.0|          16100277|                    0|
| brand|A global marketpl...|@BpackEngineer Th...|    13354.0|           2722478|                    0|
|  male|The secret of get...|Gala Bingo clubs ...|   112117.0|               255|             12639981|
|female|Pll Fan // Crazy ...|@_Aphmau_ the pic...|      482.0|           9594572|                    0|
|female|Renaissance art h...|@Evielady just ho...|    26085.0|           9594572|             16777215|
| brand|highly extraordin...|MTG Deals 1x Rank...|    66684.0|             33972|             12639981|
+------+--------------------+--------------------+-----------+------------------+---------------------+
only showing top 10 rows
```

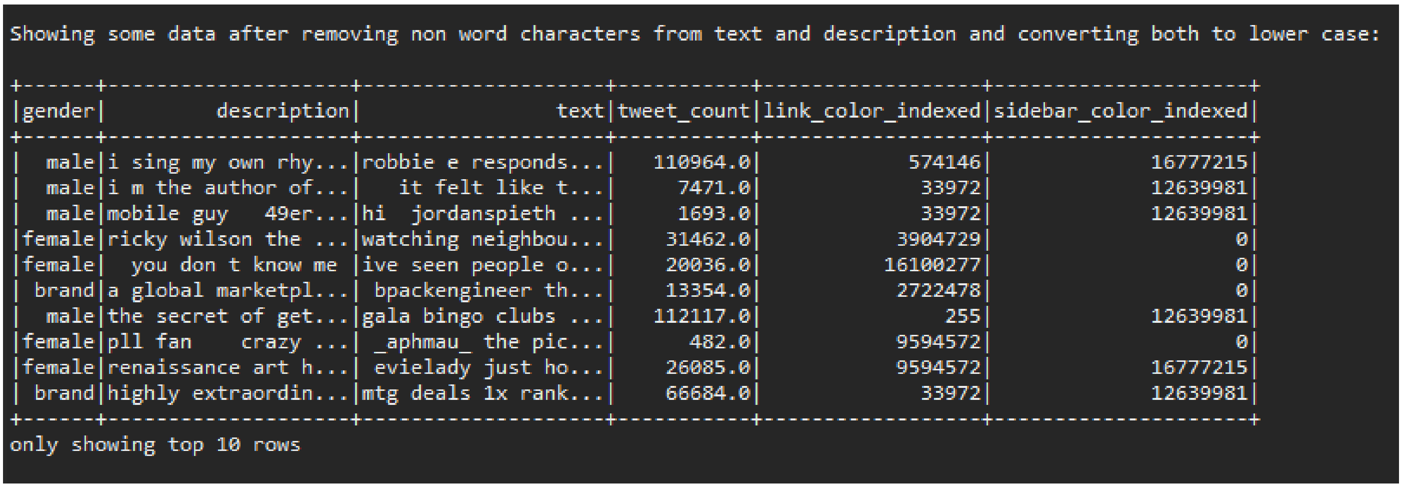
i. Fields text and description contain non word characters as well, need to remove non word characters and convert all into lower case before passing to TF-IDF computation. Below is the data as provided in input file, screenshot of code to remove non word characters and convert into lower case and screenshot of output.



gender	gender:confidence	description	text
male	1	#‰Ä£FreeZa #‰Ä£RI	Chris Got On The Black Toe 1s _
female	0.6585	!!! mutuals pls warn m	papi and cash me out are the 2
male	1	#14 RIP 10/16/14	@leezasesteaga aww don't fee
female	1	#AFNF‰Ï_ Rest in par	Maaan that shit is the worst par
brand	0.3401	#æ_fÓlfføf_ou‡ãÉf´æà	@mostly10 I agree! I think that'
male	1	#24ever NYG NYR RBN	@bobpockrass Any word on WG
male	0.6829	-- @LyssaMarine27 ‰_	Bout to cut the back of my hair
male	1	#ARMYSTRONG All it ta	I post up and that's confident..

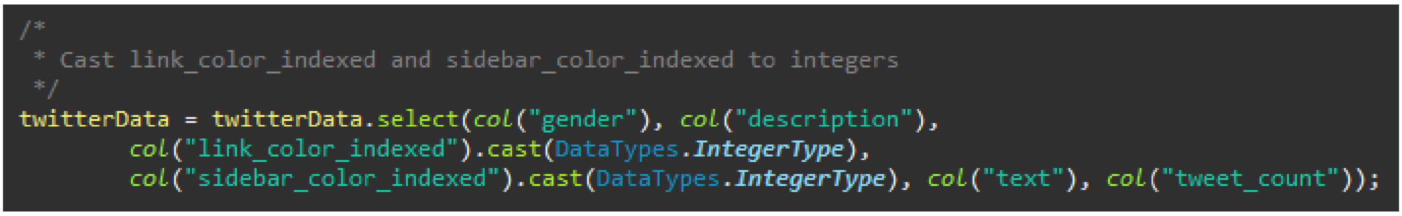


```
/*
 * Replace non word characters from text and description columns with space and
 * convert into lower case.
 */
twitterData = twitterData.withColumn("text", lower(regexp_replace(twitterData.col("text"), "[\\W]", " ")));

twitterData = twitterData.withColumn("description",
        lower(regexp_replace(twitterData.col("description"), "[\\W]", " ")));
```



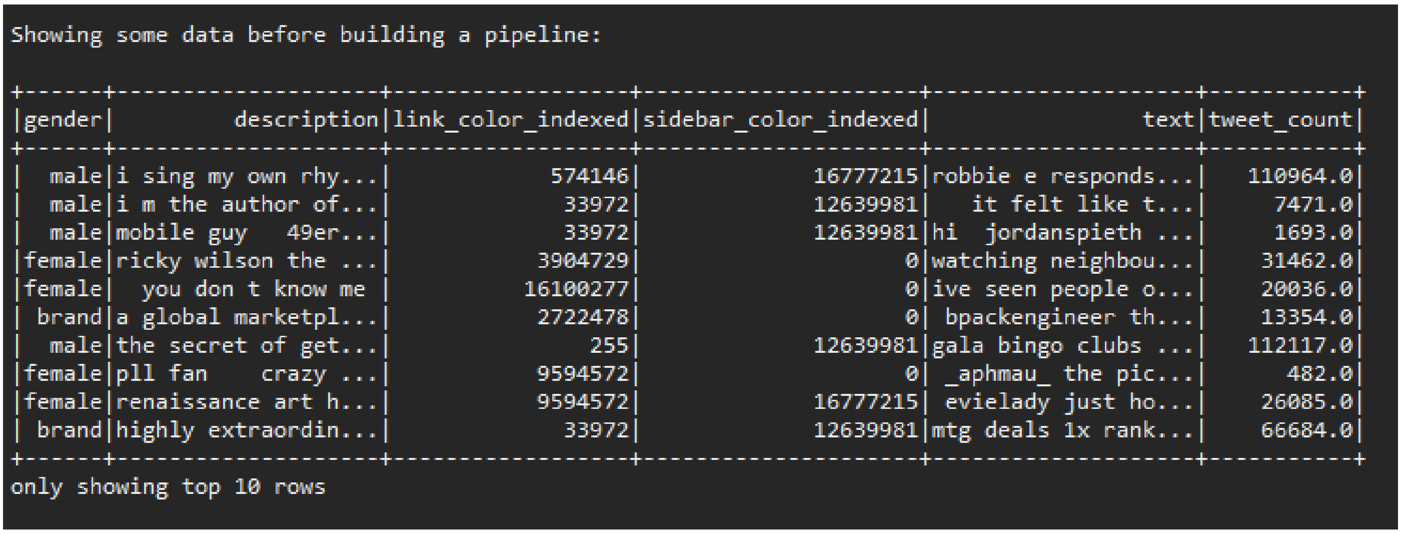
```
Showing some data after removing non word characters from text and description and converting both to lower case:

+------+--------------------+--------------------+-----------+------------------+---------------------+
|gender|         description|                text|tweet_count|link_color_indexed|sidebar_color_indexed|
+------+--------------------+--------------------+-----------+------------------+---------------------+
|  male|i sing my own rhy...|robbie e responds...|   110964.0|            574146|             16777215|
|  male|i m the author of...|   it felt like t...|     7471.0|             33972|             12639981|
|  male|mobile guy   49er...|hi  jordanspieth ...|     1693.0|             33972|             12639981|
|female|ricky wilson the ...|watching neighbou...|    31462.0|           3904729|                    0|
|female|  you don t know me |ive seen people o...|    20036.0|          16100277|                    0|
| brand|a global marketpl...| bpackengineer th...|    13354.0|           2722478|                    0|
|  male|the secret of get...|gala bingo clubs ...|   112117.0|               255|             12639981|
|female|pll fan    crazy ...| _aphmau_ the pic...|      482.0|           9594572|                    0|
|female|renaissance art h...| evielady just ho...|    26085.0|           9594572|             16777215|
| brand|highly extraordin...|mtg deals 1x rank...|    66684.0|             33972|             12639981|
+------+--------------------+--------------------+-----------+------------------+---------------------+
only showing top 10 rows
```

j. Casting link_color and sidebar_color as integer and selecting gender, text, description, tweet_count as well.



```
/*
 * Cast link_color_indexed and sidebar_color_indexed to integers
 */
twitterData = twitterData.select(col("gender"), col("description"),
        col("link_color_indexed").cast(DataTypes.IntegerType),
        col("sidebar_color_indexed").cast(DataTypes.IntegerType), col("text"), col("tweet_count"));
```

k. Below is screenshot of final data before building a pipeline.



```
Showing some data before building a pipeline:

+------+--------------------+------------------+---------------------+--------------------+-----------+
|gender|         description|link_color_indexed|sidebar_color_indexed|                text|tweet_count|
+------+--------------------+------------------+---------------------+--------------------+-----------+
|  male|i sing my own rhy...|            574146|             16777215|robbie e responds...|   110964.0|
|  male|i m the author of...|             33972|             12639981|   it felt like t...|     7471.0|
|  male|mobile guy   49er...|             33972|             12639981|hi  jordanspieth ...|     1693.0|
|female|ricky wilson the ...|           3904729|                    0|watching neighbou...|    31462.0|
|female|  you don t know me |          16100277|                    0|ive seen people o...|    20036.0|
| brand|a global marketpl...|           2722478|                    0| bpackengineer th...|    13354.0|
|  male|the secret of get...|               255|             12639981|gala bingo clubs ...|   112117.0|
|female|pll fan    crazy ...|           9594572|                    0| _aphmau_ the pic...|      482.0|
|female|renaissance art h...|           9594572|             16777215| evielady just ho...|    26085.0|
| brand|highly extraordin...|             33972|             12639981|mtg deals 1x rank...|    66684.0|
+------+--------------------+------------------+---------------------+--------------------+-----------+
only showing top 10 rows
```

2. Model Building:

a. Main program is **SparkMLClassificationAssignment.java** in the project.

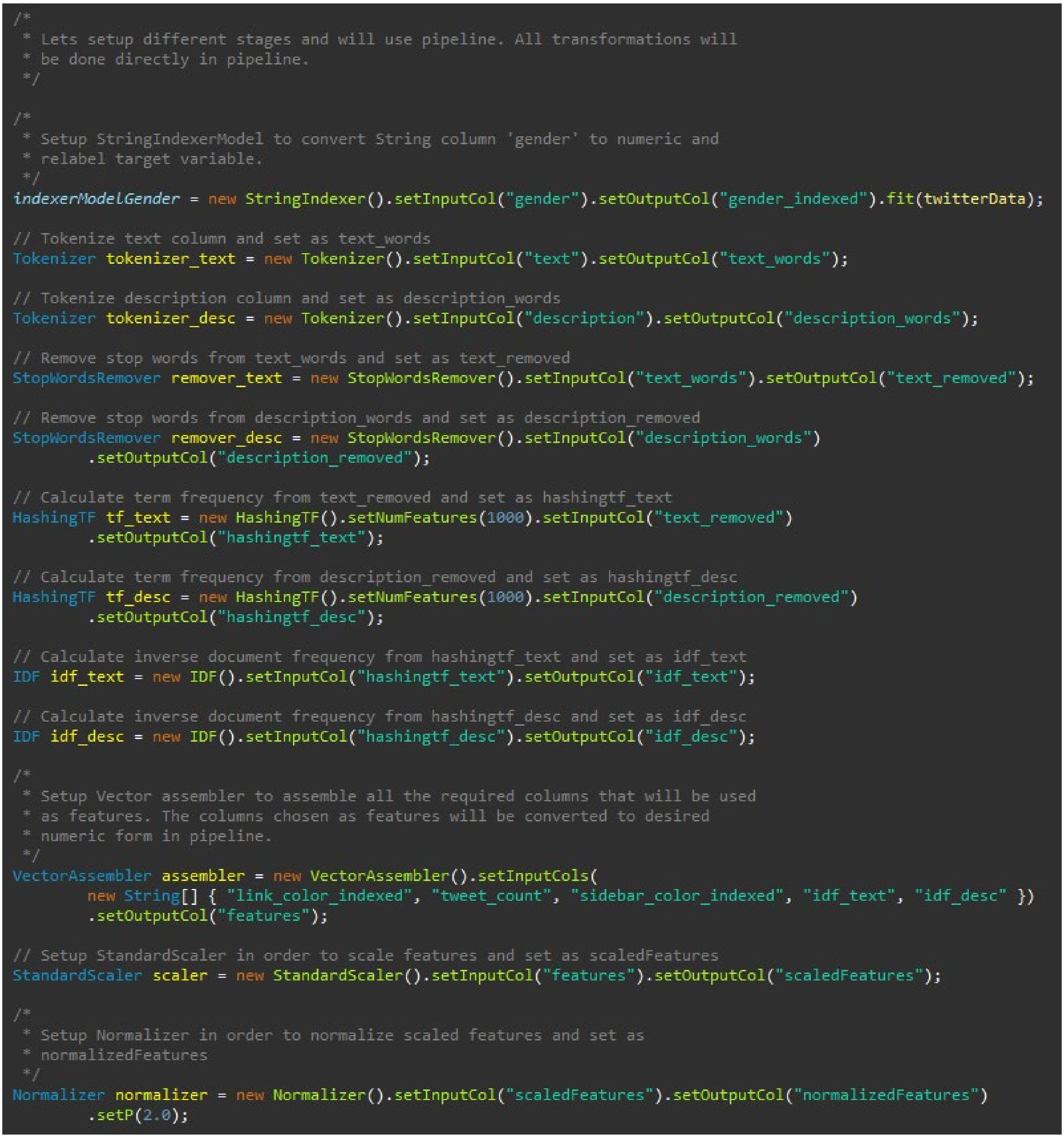
b. I am using Decision Tree and Random Forest algorithms for creating ML Classification models.

c. I am using same data columns for both Decision Tree and Random Forest in which gender field is converted into numeric using StringIndexer and other fields (**tweet_count, text, description, link_color, sidebar_color**) are used as features. Fields text and description will go through TF-IDF computation and final IDF vectors will be used for the both in place of actual fields in feature vector.

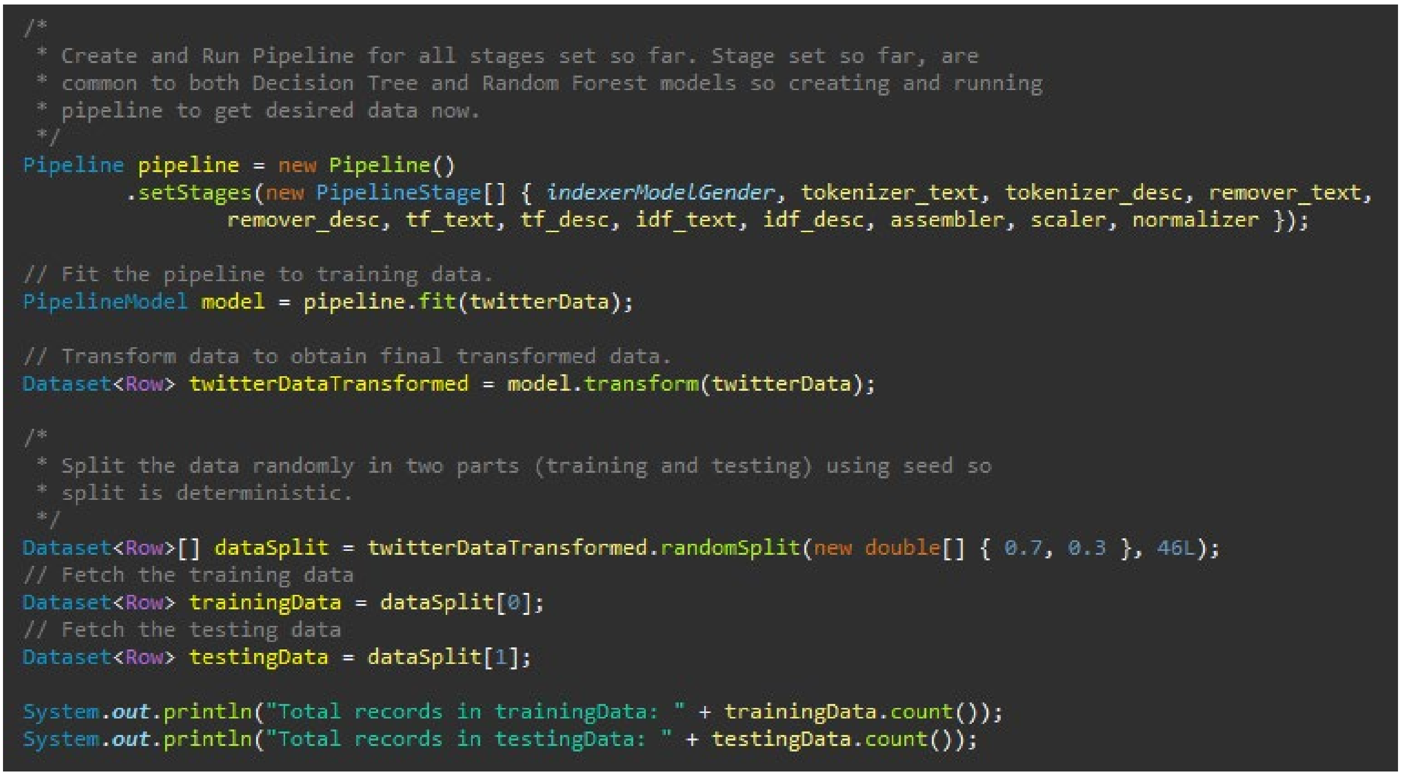
d. Based on different runs and checking accuracy in each run with different features combination, other fields are not considered in the models.

e. I am using a pipeline so no need to perform transformations for every stage in model building. I have setup different stages and using in pipeline. All transformations will be done directly in pipeline. Once pipeline is executed, I am splitting transformed data using random split into 70%-30% ratio so 70% is considered as training dataset and 30% is considered as testing dataset. Passing seed to random split function so splitting is deterministic and I get consistent result every time.

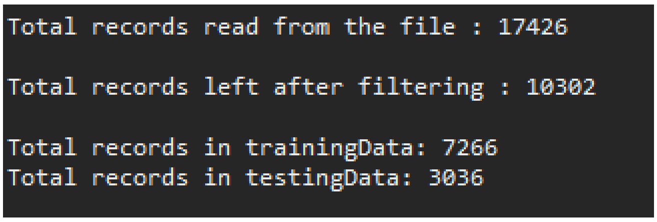
f. Below are screenshots of code showing all stages of pipeline, execution and splitting of data between training and testing datasets.



```java
/*
 * Lets setup different stages and will use pipeline. All transformations will
 * be done directly in pipeline.
 */

/*
 * Setup StringIndexerModel to convert String column 'gender' to numeric and
 * relabel target variable.
 */
indexerModelGender = new StringIndexer().setInputCol("gender").setOutputCol("gender_indexed").fit(twitterData);

// Tokenize text column and set as text_words
Tokenizer tokenizer_text = new Tokenizer().setInputCol("text").setOutputCol("text_words");

// Tokenize description column and set as description_words
Tokenizer tokenizer_desc = new Tokenizer().setInputCol("description").setOutputCol("description_words");

// Remove stop words from text_words and set as text_removed
StopWordsRemover remover_text = new StopWordsRemover().setInputCol("text_words").setOutputCol("text_removed");

// Remove stop words from description_words and set as description_removed
StopWordsRemover remover_desc = new StopWordsRemover().setInputCol("description_words")
        .setOutputCol("description_removed");

// Calculate term frequency from text_removed and set as hashingtf_text
HashingTF tf_text = new HashingTF().setNumFeatures(1000).setInputCol("text_removed")
        .setOutputCol("hashingtf_text");

// Calculate term frequency from description_removed and set as hashingtf_desc
HashingTF tf_desc = new HashingTF().setNumFeatures(1000).setInputCol("description_removed")
        .setOutputCol("hashingtf_desc");

// Calculate inverse document frequency from hashingtf_text and set as idf_text
IDF idf_text = new IDF().setInputCol("hashingtf_text").setOutputCol("idf_text");

// Calculate inverse document frequency from hashingtf_desc and set as idf_desc
IDF idf_desc = new IDF().setInputCol("hashingtf_desc").setOutputCol("idf_desc");

/*
 * Setup Vector assembler to assemble all the required columns that will be used
 * as features. The columns chosen as features will be converted to desired
 * numeric form in pipeline.
 */
VectorAssembler assembler = new VectorAssembler().setInputCols(
        new String[] { "link_color_indexed", "tweet_count", "sidebar_color_indexed", "idf_text", "idf_desc" })
        .setOutputCol("features");

// Setup StandardScaler in order to scale features and set as scaledFeatures
StandardScaler scaler = new StandardScaler().setInputCol("features").setOutputCol("scaledFeatures");

/*
 * Setup Normalizer in order to normalize scaled features and set as
 * normalizedFeatures
 */
Normalizer normalizer = new Normalizer().setInputCol("scaledFeatures").setOutputCol("normalizedFeatures")
        .setP(2.0);
```

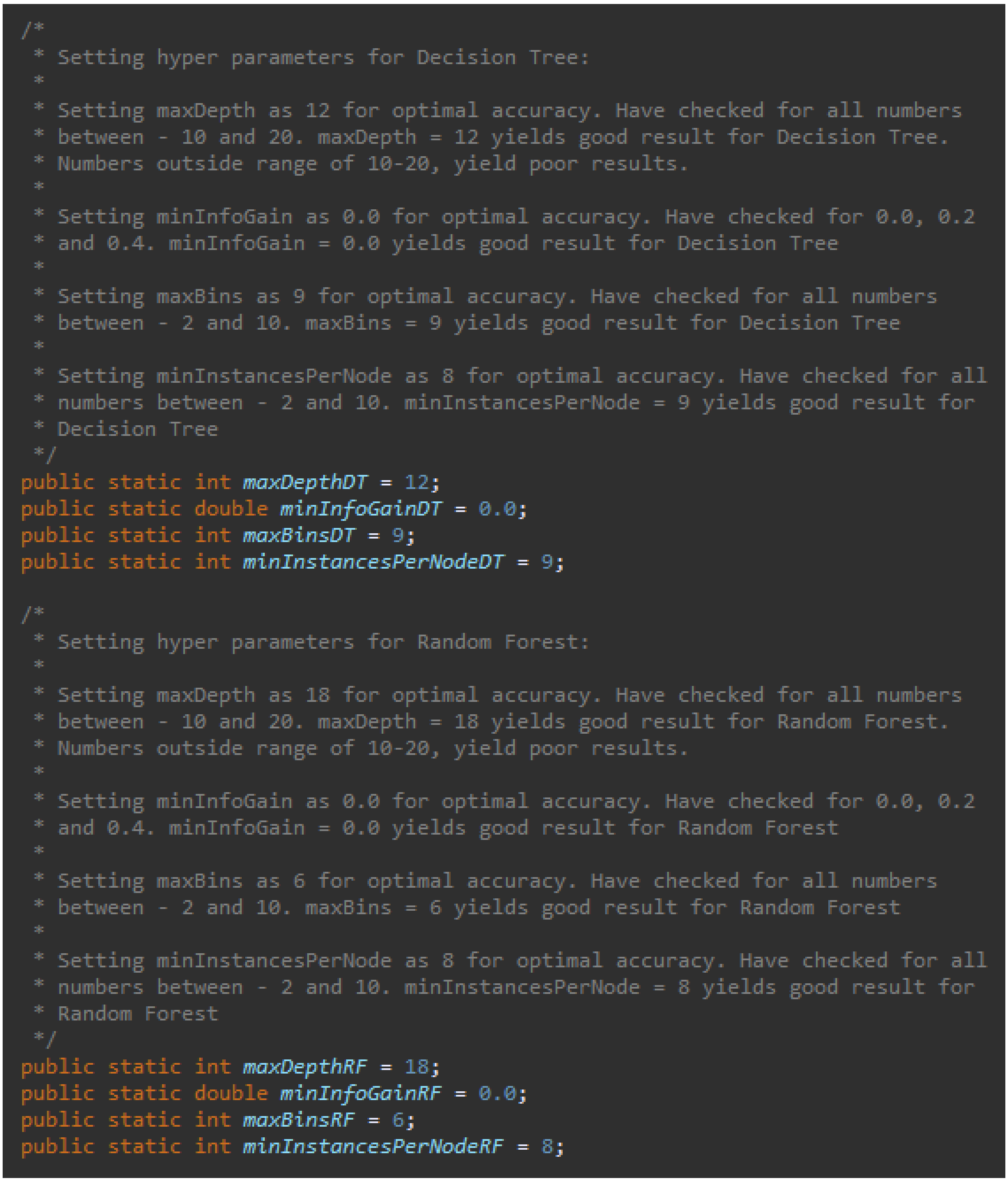


```
/*
 * Create and Run Pipeline for all stages set so far. Stage set so far, are
 * common to both Decision Tree and Random Forest models so creating and running
 * pipeline to get desired data now.
 */
Pipeline pipeline = new Pipeline()
        .setStages(new PipelineStage[] { indexerModelGender, tokenizer_text, tokenizer_desc, remover_text,
                remover_desc, tf_text, tf_desc, idf_text, idf_desc, assembler, scaler, normalizer });

// Fit the pipeline to training data.
PipelineModel model = pipeline.fit(twitterData);

// Transform data to obtain final transformed data.
Dataset<Row> twitterDataTransformed = model.transform(twitterData);

/*
 * Split the data randomly in two parts (training and testing) using seed so
 * split is deterministic.
 */
Dataset<Row>[] dataSplit = twitterDataTransformed.randomSplit(new double[] { 0.7, 0.3 }, 46L);
// Fetch the training data
Dataset<Row> trainingData = dataSplit[0];
// Fetch the testing data
Dataset<Row> testingData = dataSplit[1];

System.out.println("Total records in trainingData: " + trainingData.count());
System.out.println("Total records in testingData: " + testingData.count());
```



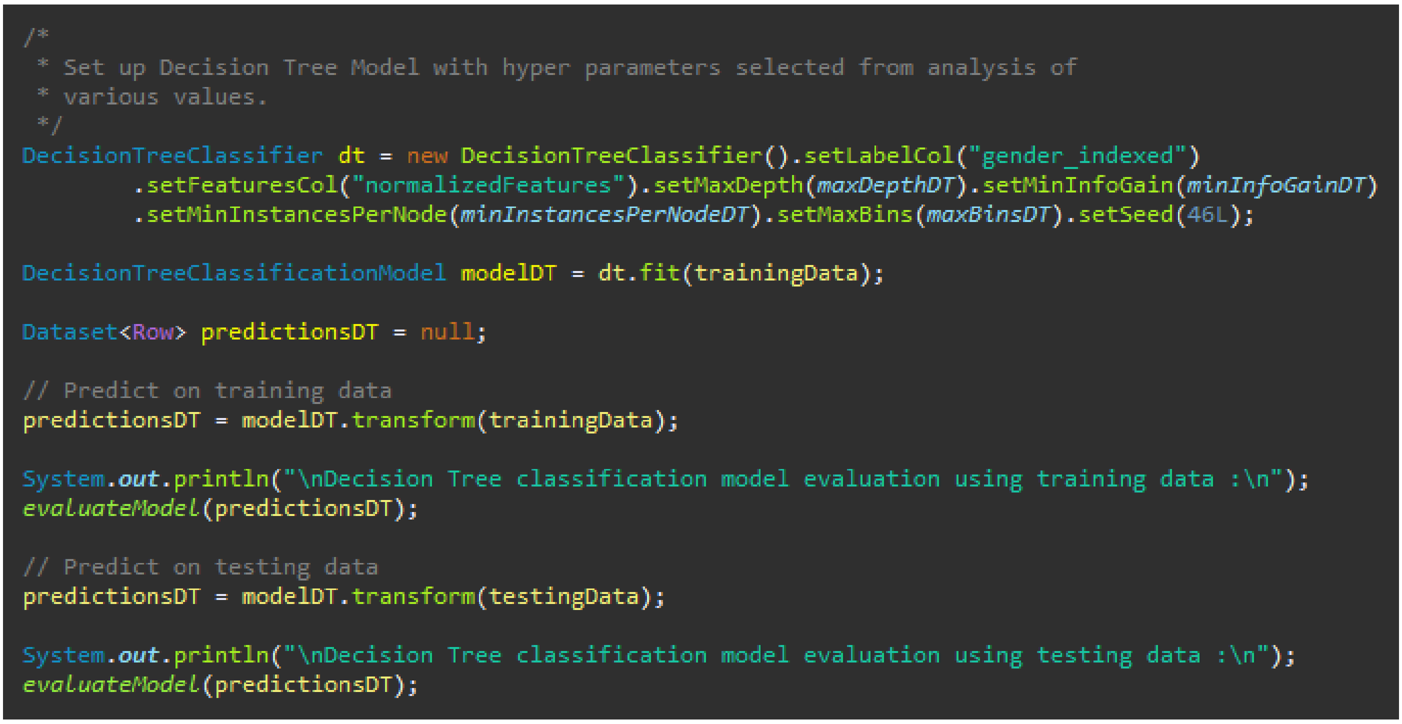
```
Total records read from the file : 17426

Total records left after filtering : 10302

Total records in trainingData: 7266
Total records in testingData: 3036
```

g. I am using 4 hyperparameters namely maxDepth, minInfoGain, maxBins and minInstancesPerNode for both Decision Tree and Random Forest, shown as follows:



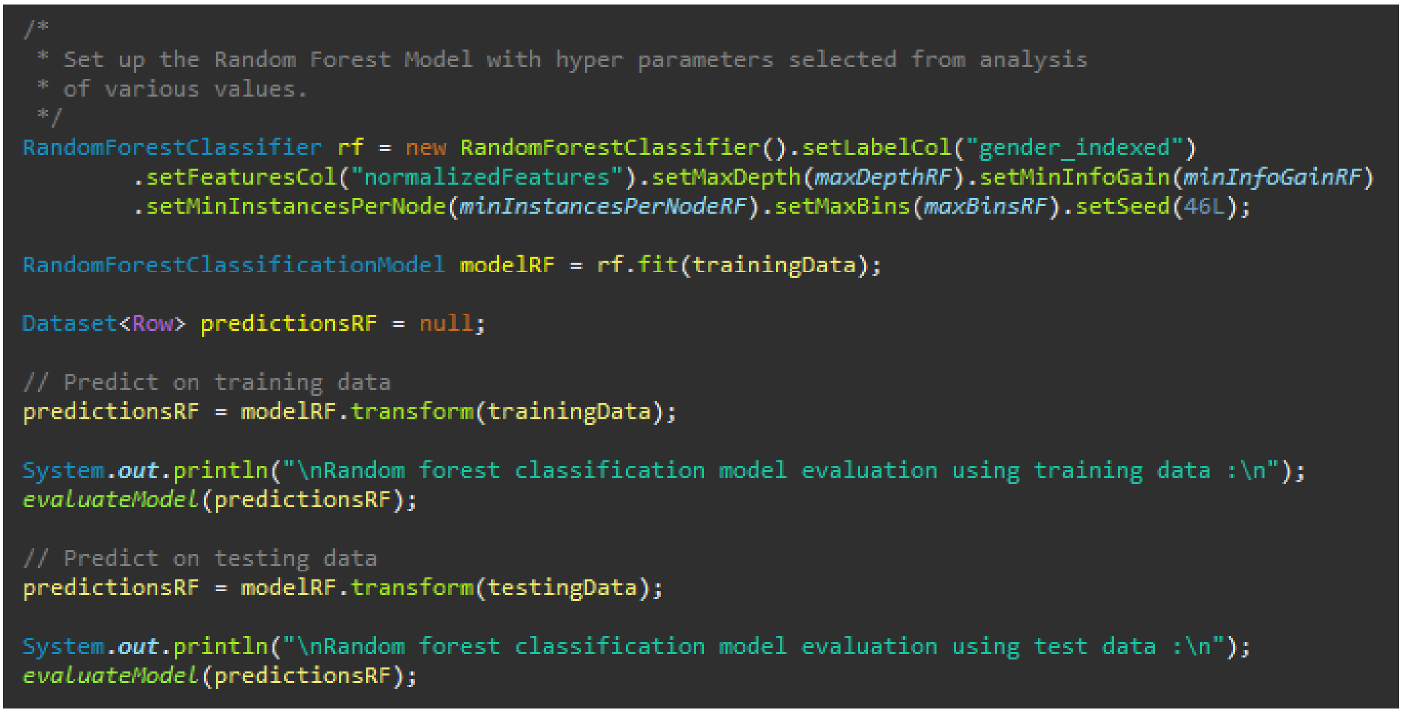
```
/*
 * Setting hyper parameters for Decision Tree:
 *
 * Setting maxDepth as 12 for optimal accuracy. Have checked for all numbers
 * between - 10 and 20. maxDepth = 12 yields good result for Decision Tree.
 * Numbers outside range of 10-20, yield poor results.
 *
 * Setting minInfoGain as 0.0 for optimal accuracy. Have checked for 0.0, 0.2
 * and 0.4. minInfoGain = 0.0 yields good result for Decision Tree
 *
 * Setting maxBins as 9 for optimal accuracy. Have checked for all numbers
 * between - 2 and 10. maxBins = 9 yields good result for Decision Tree
 *
 * Setting minInstancesPerNode as 8 for optimal accuracy. Have checked for all
 * numbers between - 2 and 10. minInstancesPerNode = 9 yields good result for
 * Decision Tree
 */
public static int maxDepthDT = 12;
public static double minInfoGainDT = 0.0;
public static int maxBinsDT = 9;
public static int minInstancesPerNodeDT = 9;

/*
 * Setting hyper parameters for Random Forest:
 *
 * Setting maxDepth as 18 for optimal accuracy. Have checked for all numbers
 * between - 10 and 20. maxDepth = 18 yields good result for Random Forest.
 * Numbers outside range of 10-20, yield poor results.
 *
 * Setting minInfoGain as 0.0 for optimal accuracy. Have checked for 0.0, 0.2
 * and 0.4. minInfoGain = 0.0 yields good result for Random Forest
 *
 * Setting maxBins as 6 for optimal accuracy. Have checked for all numbers
 * between - 2 and 10. maxBins = 6 yields good result for Random Forest
 *
 * Setting minInstancesPerNode as 8 for optimal accuracy. Have checked for all
 * numbers between - 2 and 10. minInstancesPerNode = 8 yields good result for
 * Random Forest
 */
public static int maxDepthRF = 18;
public static double minInfoGainRF = 0.0;
public static int maxBinsRF = 6;
public static int minInstancesPerNodeRF = 8;
```

h. Below is screenshot of **Decision Tree Model**. I have first trained the model on training data and calculated evaluation matrix. Then trained model on testing data and calculated evaluation matrix. Evaluation matrix for both training data and testing data is shown in point 3 in this document.



```
/*
 * Set up Decision Tree Model with hyper parameters selected from analysis of
 * various values.
 */
DecisionTreeClassifier dt = new DecisionTreeClassifier().setLabelCol("gender_indexed")
        .setFeaturesCol("normalizedFeatures").setMaxDepth(maxDepthDT).setMinInfoGain(minInfoGainDT)
        .setMinInstancesPerNode(minInstancesPerNodeDT).setMaxBins(maxBinsDT).setSeed(46L);

DecisionTreeClassificationModel modelDT = dt.fit(trainingData);

Dataset<Row> predictionsDT = null;

// Predict on training data
predictionsDT = modelDT.transform(trainingData);

System.out.println("\nDecision Tree classification model evaluation using training data :\n");
evaluateModel(predictionsDT);

// Predict on testing data
predictionsDT = modelDT.transform(testingData);

System.out.println("\nDecision Tree classification model evaluation using testing data :\n");
evaluateModel(predictionsDT);
```

i. Below is the screenshot of **Random Forest Model**. I have first trained the model on training data and calculated evaluation matrix. Then trained model on testing data and calculated evaluation matrix. Evaluation matrix for both training data and testing data is shown in point 3 in this document.



```
/*
 * Set up the Random Forest Model with hyper parameters selected from analysis
 * of various values.
 */
RandomForestClassifier rf = new RandomForestClassifier().setLabelCol("gender_indexed")
        .setFeaturesCol("normalizedFeatures").setMaxDepth(maxDepthRF).setMinInfoGain(minInfoGainRF)
        .setMinInstancesPerNode(minInstancesPerNodeRF).setMaxBins(maxBinsRF).setSeed(46L);

RandomForestClassificationModel modelRF = rf.fit(trainingData);

Dataset<Row> predictionsRF = null;

// Predict on training data
predictionsRF = modelRF.transform(trainingData);

System.out.println("\nRandom forest classification model evaluation using training data :\n");
evaluateModel(predictionsRF);

// Predict on testing data
predictionsRF = modelRF.transform(testingData);

System.out.println("\nRandom forest classification model evaluation using test data :\n");
evaluateModel(predictionsRF);
```

j. I have submitted **SparkMLClassificationAssignmentModelTesting.java** program in the same project, in which I have tried different models for both Decision Tree and Random Forest with different hyperparameters and calculated accuracy of each model on testing data. Based on performance, I have chosen hyper parameters for Decision Tree and Random Forest. Below are the finally chosen hyper parameters for Decision Tree and Random Forest algorithms.

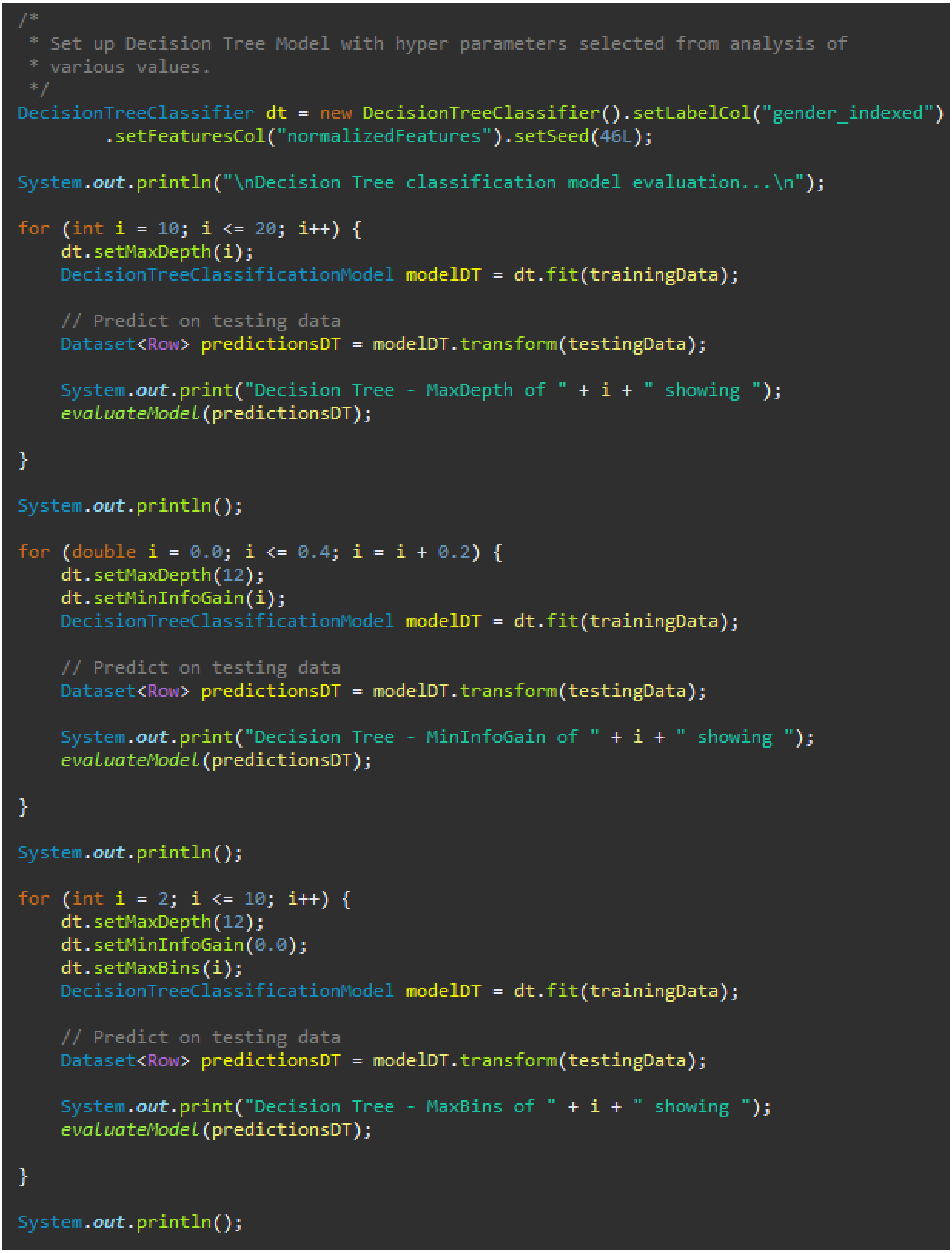
For Decision Tree:

maxDepth = 12
minInfoGain = 0.0
maxBins = 9
minInstancesPerNode = 9

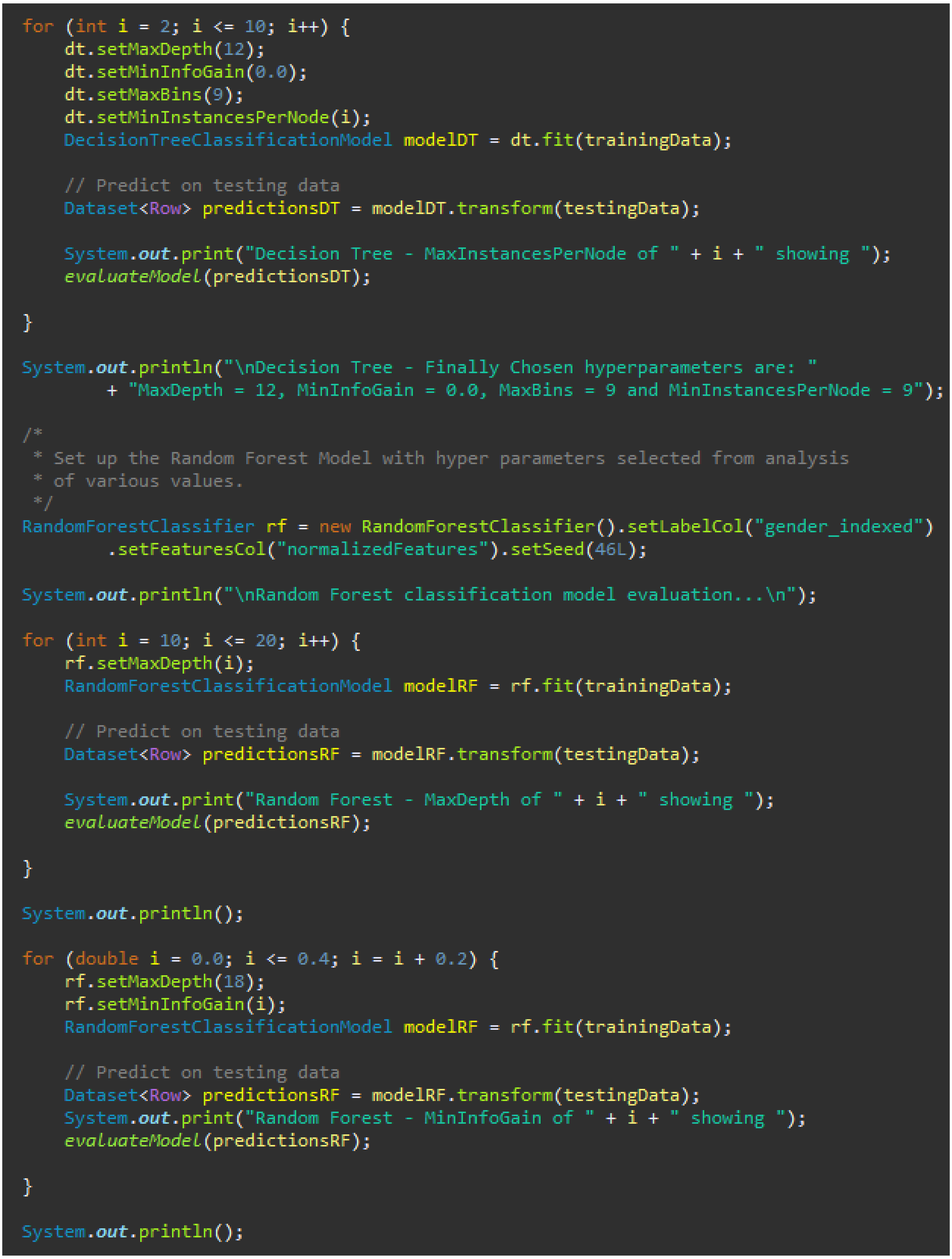
For Random Forest:

maxDepth = 18
minInfoGain = 0.0
maxBins = 6
minInstancesPerNode = 8

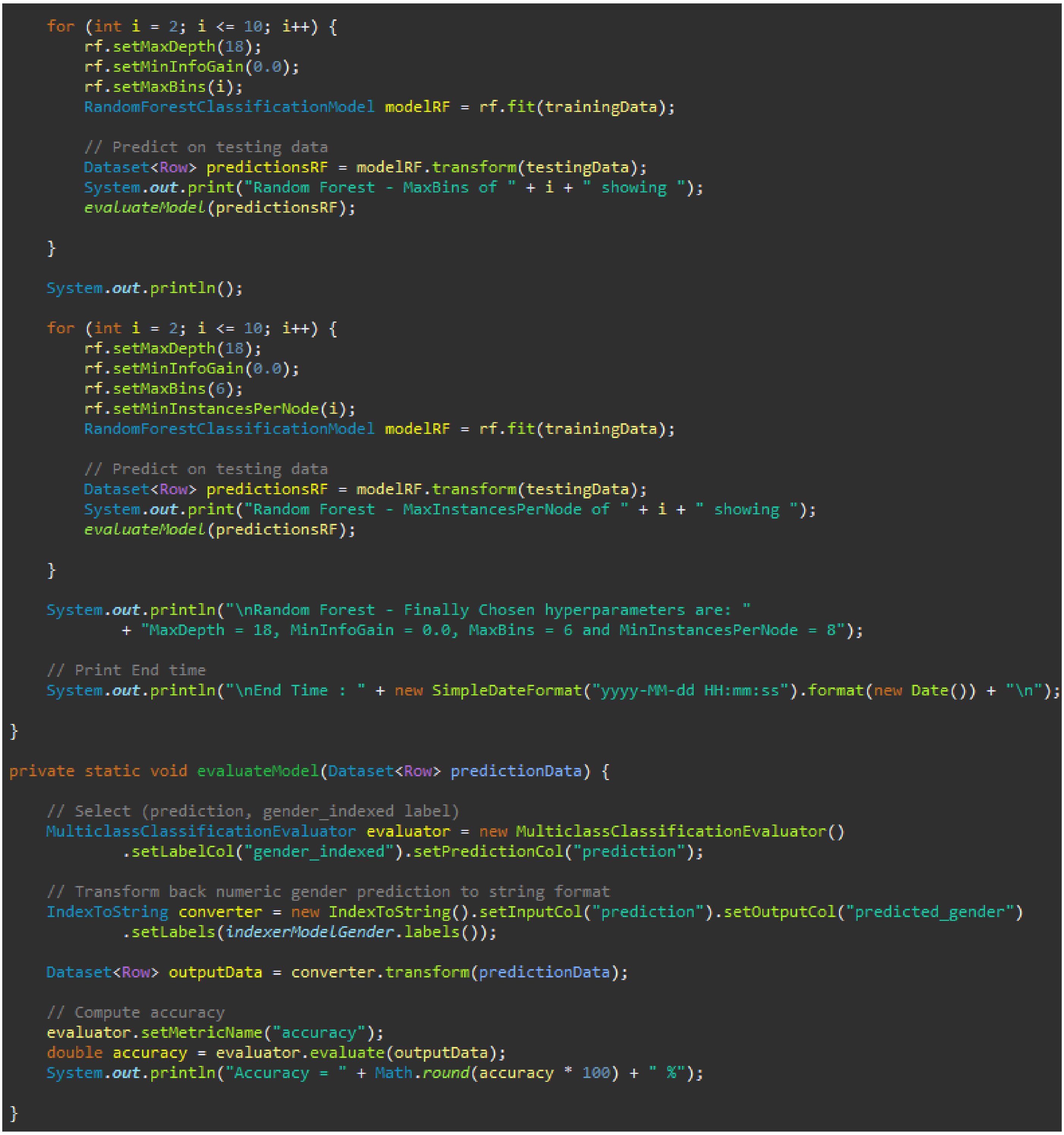
k. Below are the screenshots of code for various models and hyper parameters used for testing and finalizing hyper parameters for final models. Please note that values of hyper parameters beyond the range used in the code for testing, show poor performance so have been ignored.



```
/*
 * Set up Decision Tree Model with hyper parameters selected from analysis of
 * various values.
 */
DecisionTreeClassifier dt = new DecisionTreeClassifier().setLabelCol("gender_indexed")
        .setFeaturesCol("normalizedFeatures").setSeed(46L);

System.out.println("\nDecision Tree classification model evaluation...\n");

for (int i = 10; i <= 20; i++) {
    dt.setMaxDepth(i);
    DecisionTreeClassificationModel modelDT = dt.fit(trainingData);

    // Predict on testing data
    Dataset<Row> predictionsDT = modelDT.transform(testingData);

    System.out.print("Decision Tree - MaxDepth of " + i + " showing ");
    evaluateModel(predictionsDT);

}

System.out.println();

for (double i = 0.0; i <= 0.4; i = i + 0.2) {
    dt.setMaxDepth(12);
    dt.setMinInfoGain(i);
    DecisionTreeClassificationModel modelDT = dt.fit(trainingData);

    // Predict on testing data
    Dataset<Row> predictionsDT = modelDT.transform(testingData);

    System.out.print("Decision Tree - MinInfoGain of " + i + " showing ");
    evaluateModel(predictionsDT);

}

System.out.println();

for (int i = 2; i <= 10; i++) {
    dt.setMaxDepth(12);
    dt.setMinInfoGain(0.0);
    dt.setMaxBins(i);
    DecisionTreeClassificationModel modelDT = dt.fit(trainingData);

    // Predict on testing data
    Dataset<Row> predictionsDT = modelDT.transform(testingData);

    System.out.print("Decision Tree - MaxBins of " + i + " showing ");
    evaluateModel(predictionsDT);

}

System.out.println();
```

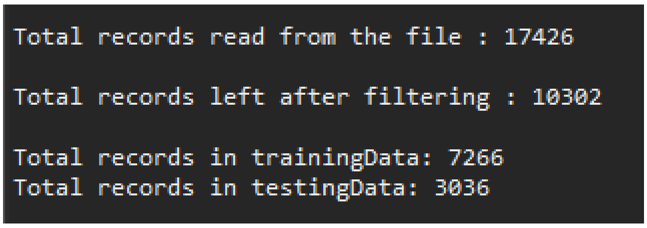


```java
for (int i = 2; i <= 10; i++) {
    dt.setMaxDepth(12);
    dt.setMinInfoGain(0.0);
    dt.setMaxBins(9);
    dt.setMinInstancesPerNode(i);
    DecisionTreeClassificationModel modelDT = dt.fit(trainingData);

    // Predict on testing data
    Dataset<Row> predictionsDT = modelDT.transform(testingData);

    System.out.print("Decision Tree - MaxInstancesPerNode of " + i + " showing ");
    evaluateModel(predictionsDT);

}

System.out.println("\nDecision Tree - Finally Chosen hyperparameters are: "
        + "MaxDepth = 12, MinInfoGain = 0.0, MaxBins = 9 and MinInstancesPerNode = 9");

/*
 * Set up the Random Forest Model with hyper parameters selected from analysis
 * of various values.
 */
RandomForestClassifier rf = new RandomForestClassifier().setLabelCol("gender_indexed")
        .setFeaturesCol("normalizedFeatures").setSeed(46L);

System.out.println("\nRandom Forest classification model evaluation...\n");

for (int i = 10; i <= 20; i++) {
    rf.setMaxDepth(i);
    RandomForestClassificationModel modelRF = rf.fit(trainingData);

    // Predict on testing data
    Dataset<Row> predictionsRF = modelRF.transform(testingData);

    System.out.print("Random Forest - MaxDepth of " + i + " showing ");
    evaluateModel(predictionsRF);

}

System.out.println();

for (double i = 0.0; i <= 0.4; i = i + 0.2) {
    rf.setMaxDepth(18);
    rf.setMinInfoGain(i);
    RandomForestClassificationModel modelRF = rf.fit(trainingData);

    // Predict on testing data
    Dataset<Row> predictionsRF = modelRF.transform(testingData);
    System.out.print("Random Forest - MinInfoGain of " + i + " showing ");
    evaluateModel(predictionsRF);

}

System.out.println();
```

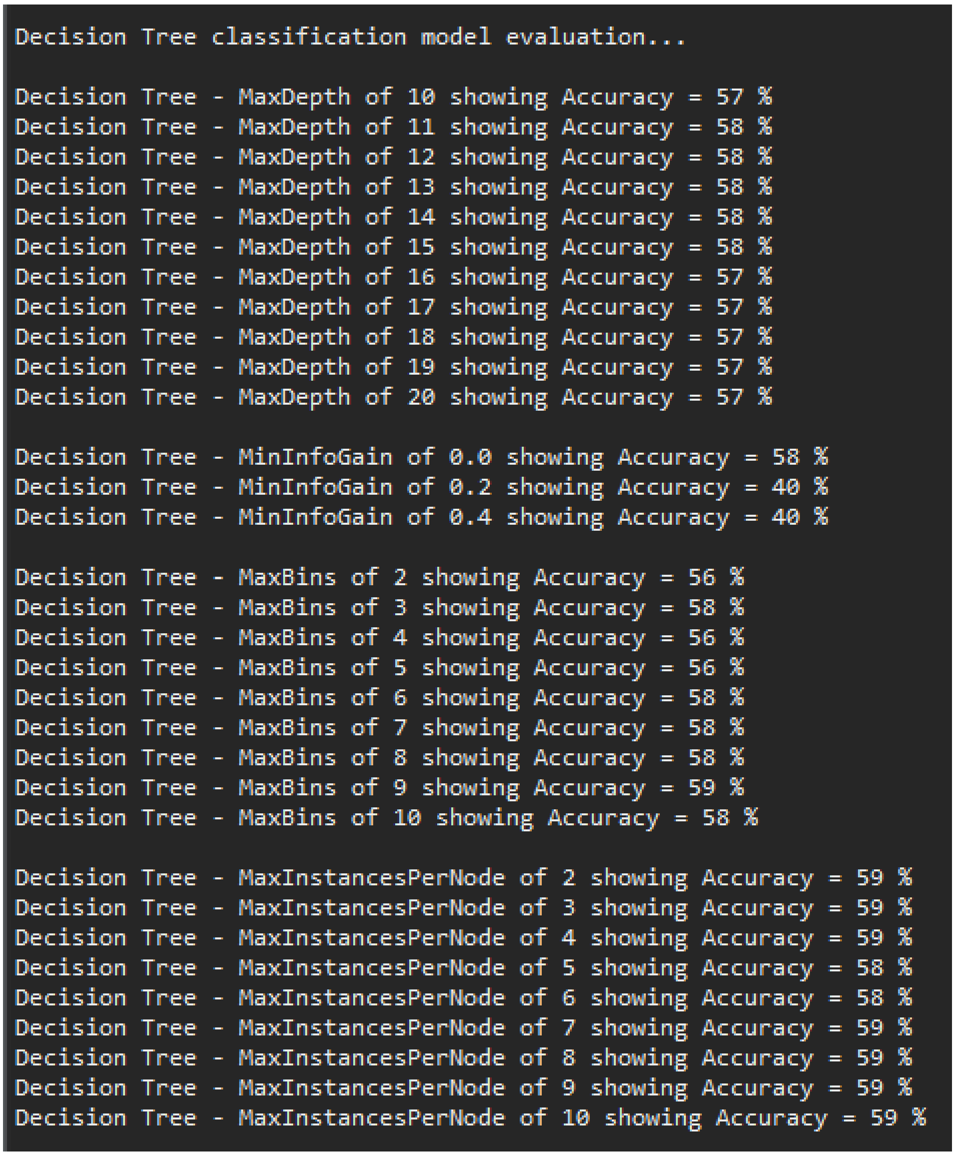


```java
    for (int i = 2; i <= 10; i++) {
        rf.setMaxDepth(18);
        rf.setMinInfoGain(0.0);
        rf.setMaxBins(i);
        RandomForestClassificationModel modelRF = rf.fit(trainingData);

        // Predict on testing data
        Dataset<Row> predictionsRF = modelRF.transform(testingData);
        System.out.print("Random Forest - MaxBins of " + i + " showing ");
        evaluateModel(predictionsRF);

    }

    System.out.println();

    for (int i = 2; i <= 10; i++) {
        rf.setMaxDepth(18);
        rf.setMinInfoGain(0.0);
        rf.setMaxBins(6);
        rf.setMinInstancesPerNode(i);
        RandomForestClassificationModel modelRF = rf.fit(trainingData);

        // Predict on testing data
        Dataset<Row> predictionsRF = modelRF.transform(testingData);
        System.out.print("Random Forest - MaxInstancesPerNode of " + i + " showing ");
        evaluateModel(predictionsRF);

    }

    System.out.println("\nRandom Forest - Finally Chosen hyperparameters are: "
            + "MaxDepth = 18, MinInfoGain = 0.0, MaxBins = 6 and MinInstancesPerNode = 8");

    // Print End time
    System.out.println("\nEnd Time : " + new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(new Date()) + "\n");

}

private static void evaluateModel(Dataset<Row> predictionData) {

    // Select (prediction, gender_indexed label)
    MulticlassClassificationEvaluator evaluator = new MulticlassClassificationEvaluator()
            .setLabelCol("gender_indexed").setPredictionCol("prediction");

    // Transform back numeric gender prediction to string format
    IndexToString converter = new IndexToString().setInputCol("prediction").setOutputCol("predicted_gender")
            .setLabels(indexerModelGender.labels());

    Dataset<Row> outputData = converter.transform(predictionData);

    // Compute accuracy
    evaluator.setMetricName("accuracy");
    double accuracy = evaluator.evaluate(outputData);
    System.out.println("Accuracy = " + Math.round(accuracy * 100) + " %");

}
```

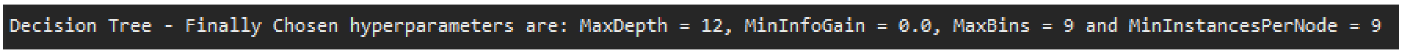
l. Below is output of **SparkMLClassificationAssignmentModelTesting.java** based on which hyper parameters for Decision Tree and Random Forest, are finally chosen.



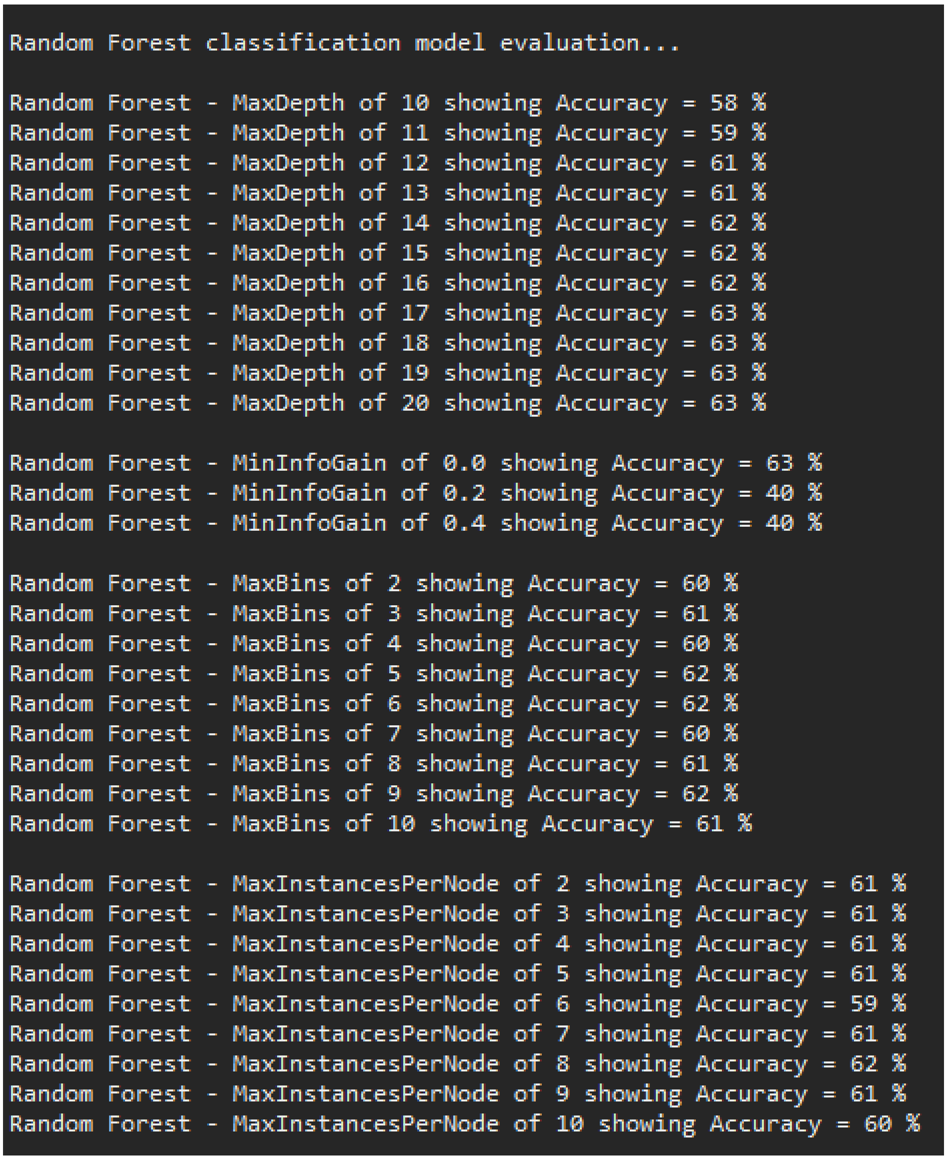
```
Total records read from the file : 17426

Total records left after filtering : 10302

Total records in trainingData: 7266
Total records in testingData: 3036
```



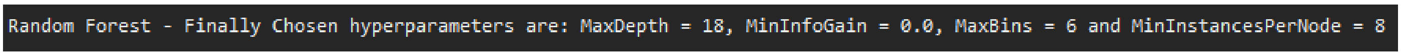
```
Decision Tree classification model evaluation...

Decision Tree - MaxDepth of 10 showing Accuracy = 57 %
Decision Tree - MaxDepth of 11 showing Accuracy = 58 %
Decision Tree - MaxDepth of 12 showing Accuracy = 58 %
Decision Tree - MaxDepth of 13 showing Accuracy = 58 %
Decision Tree - MaxDepth of 14 showing Accuracy = 58 %
Decision Tree - MaxDepth of 15 showing Accuracy = 58 %
Decision Tree - MaxDepth of 16 showing Accuracy = 57 %
Decision Tree - MaxDepth of 17 showing Accuracy = 57 %
Decision Tree - MaxDepth of 18 showing Accuracy = 57 %
Decision Tree - MaxDepth of 19 showing Accuracy = 57 %
Decision Tree - MaxDepth of 20 showing Accuracy = 57 %

Decision Tree - MinInfoGain of 0.0 showing Accuracy = 58 %
Decision Tree - MinInfoGain of 0.2 showing Accuracy = 40 %
Decision Tree - MinInfoGain of 0.4 showing Accuracy = 40 %

Decision Tree - MaxBins of 2 showing Accuracy = 56 %
Decision Tree - MaxBins of 3 showing Accuracy = 58 %
Decision Tree - MaxBins of 4 showing Accuracy = 56 %
Decision Tree - MaxBins of 5 showing Accuracy = 56 %
Decision Tree - MaxBins of 6 showing Accuracy = 58 %
Decision Tree - MaxBins of 7 showing Accuracy = 58 %
Decision Tree - MaxBins of 8 showing Accuracy = 58 %
Decision Tree - MaxBins of 9 showing Accuracy = 59 %
Decision Tree - MaxBins of 10 showing Accuracy = 58 %

Decision Tree - MaxInstancesPerNode of 2 showing Accuracy = 59 %
Decision Tree - MaxInstancesPerNode of 3 showing Accuracy = 59 %
Decision Tree - MaxInstancesPerNode of 4 showing Accuracy = 59 %
Decision Tree - MaxInstancesPerNode of 5 showing Accuracy = 58 %
Decision Tree - MaxInstancesPerNode of 6 showing Accuracy = 58 %
Decision Tree - MaxInstancesPerNode of 7 showing Accuracy = 59 %
Decision Tree - MaxInstancesPerNode of 8 showing Accuracy = 59 %
Decision Tree - MaxInstancesPerNode of 9 showing Accuracy = 59 %
Decision Tree - MaxInstancesPerNode of 10 showing Accuracy = 59 %
```



```
Decision Tree - Finally Chosen hyperparameters are: MaxDepth = 12, MinInfoGain = 0.0, MaxBins = 9 and MinInstancesPerNode = 9
```



```
Random Forest classification model evaluation...

Random Forest - MaxDepth of 10 showing Accuracy = 58 %
Random Forest - MaxDepth of 11 showing Accuracy = 59 %
Random Forest - MaxDepth of 12 showing Accuracy = 61 %
Random Forest - MaxDepth of 13 showing Accuracy = 61 %
Random Forest - MaxDepth of 14 showing Accuracy = 62 %
Random Forest - MaxDepth of 15 showing Accuracy = 62 %
Random Forest - MaxDepth of 16 showing Accuracy = 62 %
Random Forest - MaxDepth of 17 showing Accuracy = 63 %
Random Forest - MaxDepth of 18 showing Accuracy = 63 %
Random Forest - MaxDepth of 19 showing Accuracy = 63 %
Random Forest - MaxDepth of 20 showing Accuracy = 63 %

Random Forest - MinInfoGain of 0.0 showing Accuracy = 63 %
Random Forest - MinInfoGain of 0.2 showing Accuracy = 40 %
Random Forest - MinInfoGain of 0.4 showing Accuracy = 40 %

Random Forest - MaxBins of 2 showing Accuracy = 60 %
Random Forest - MaxBins of 3 showing Accuracy = 61 %
Random Forest - MaxBins of 4 showing Accuracy = 60 %
Random Forest - MaxBins of 5 showing Accuracy = 62 %
Random Forest - MaxBins of 6 showing Accuracy = 62 %
Random Forest - MaxBins of 7 showing Accuracy = 60 %
Random Forest - MaxBins of 8 showing Accuracy = 61 %
Random Forest - MaxBins of 9 showing Accuracy = 62 %
Random Forest - MaxBins of 10 showing Accuracy = 61 %

Random Forest - MaxInstancesPerNode of 2 showing Accuracy = 61 %
Random Forest - MaxInstancesPerNode of 3 showing Accuracy = 61 %
Random Forest - MaxInstancesPerNode of 4 showing Accuracy = 61 %
Random Forest - MaxInstancesPerNode of 5 showing Accuracy = 61 %
Random Forest - MaxInstancesPerNode of 6 showing Accuracy = 59 %
Random Forest - MaxInstancesPerNode of 7 showing Accuracy = 61 %
Random Forest - MaxInstancesPerNode of 8 showing Accuracy = 62 %
Random Forest - MaxInstancesPerNode of 9 showing Accuracy = 61 %
Random Forest - MaxInstancesPerNode of 10 showing Accuracy = 60 %
```



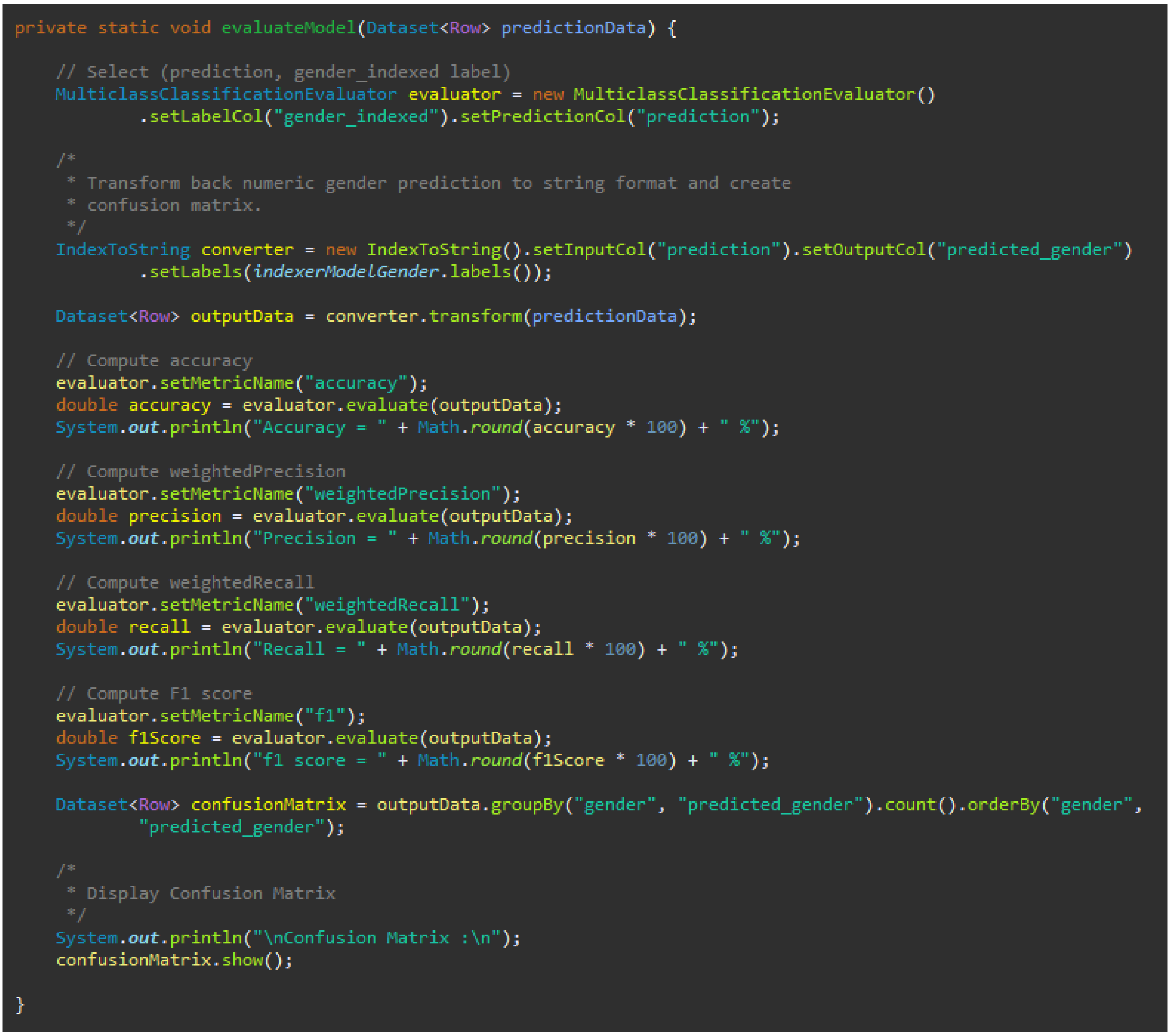
```
Random Forest - Finally Chosen hyperparameters are: MaxDepth = 18, MinInfoGain = 0.0, MaxBins = 6 and MinInstancesPerNode = 8
```

3. Evaluation Metrics:

a. Below is screenshot of code for performance metrics in **SparkMLClassificationAssignment.java**:

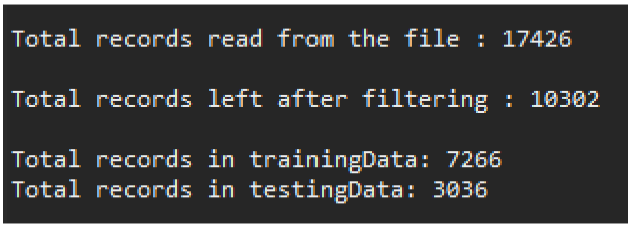
(i) Evaluation scores: Accuracy, Precision, Recall, F1 Score

(ii) Confusion Matrix

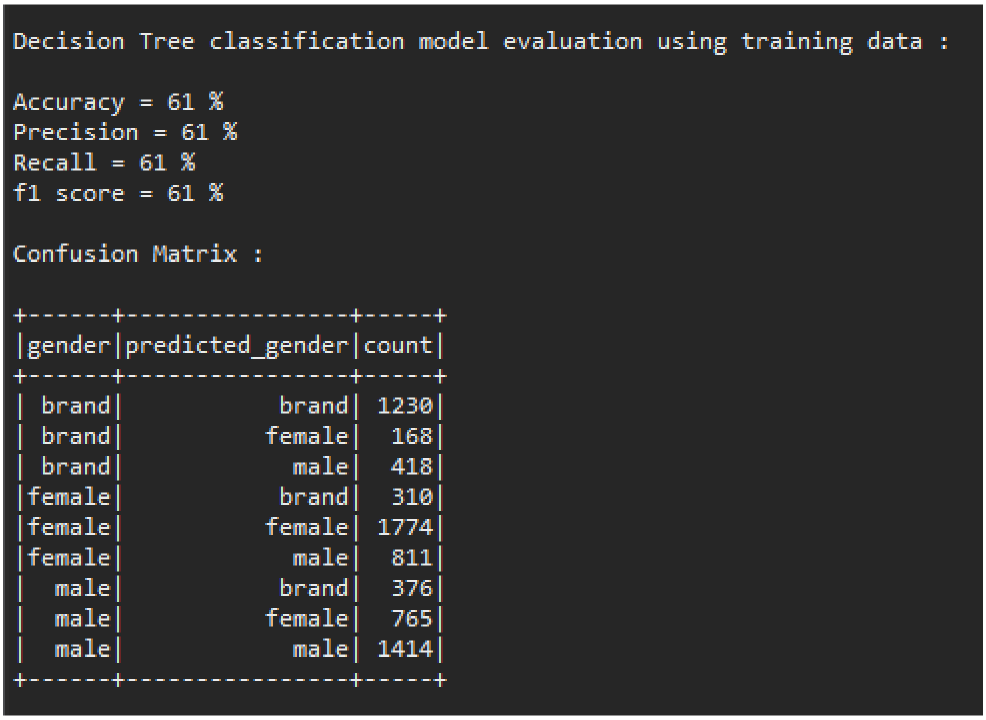


```
private static void evaluateModel(Dataset<Row> predictionData) {

    // Select (prediction, gender_indexed label)
    MulticlassClassificationEvaluator evaluator = new MulticlassClassificationEvaluator()
            .setLabelCol("gender_indexed").setPredictionCol("prediction");

    /*
     * Transform back numeric gender prediction to string format and create
     * confusion matrix.
     */
    IndexToString converter = new IndexToString().setInputCol("prediction").setOutputCol("predicted_gender")
            .setLabels(indexerModelGender.labels());

    Dataset<Row> outputData = converter.transform(predictionData);

    // Compute accuracy
    evaluator.setMetricName("accuracy");
    double accuracy = evaluator.evaluate(outputData);
    System.out.println("Accuracy = " + Math.round(accuracy * 100) + " %");

    // Compute weightedPrecision
    evaluator.setMetricName("weightedPrecision");
    double precision = evaluator.evaluate(outputData);
    System.out.println("Precision = " + Math.round(precision * 100) + " %");

    // Compute weightedRecall
    evaluator.setMetricName("weightedRecall");
    double recall = evaluator.evaluate(outputData);
    System.out.println("Recall = " + Math.round(recall * 100) + " %");

    // Compute F1 score
    evaluator.setMetricName("f1");
    double f1Score = evaluator.evaluate(outputData);
    System.out.println("f1 score = " + Math.round(f1Score * 100) + " %");

    Dataset<Row> confusionMatrix = outputData.groupBy("gender", "predicted_gender").count().orderBy("gender",
            "predicted_gender");

    /*
     * Display Confusion Matrix
     */
    System.out.println("\nConfusion Matrix :\n");
    confusionMatrix.show();

}
```

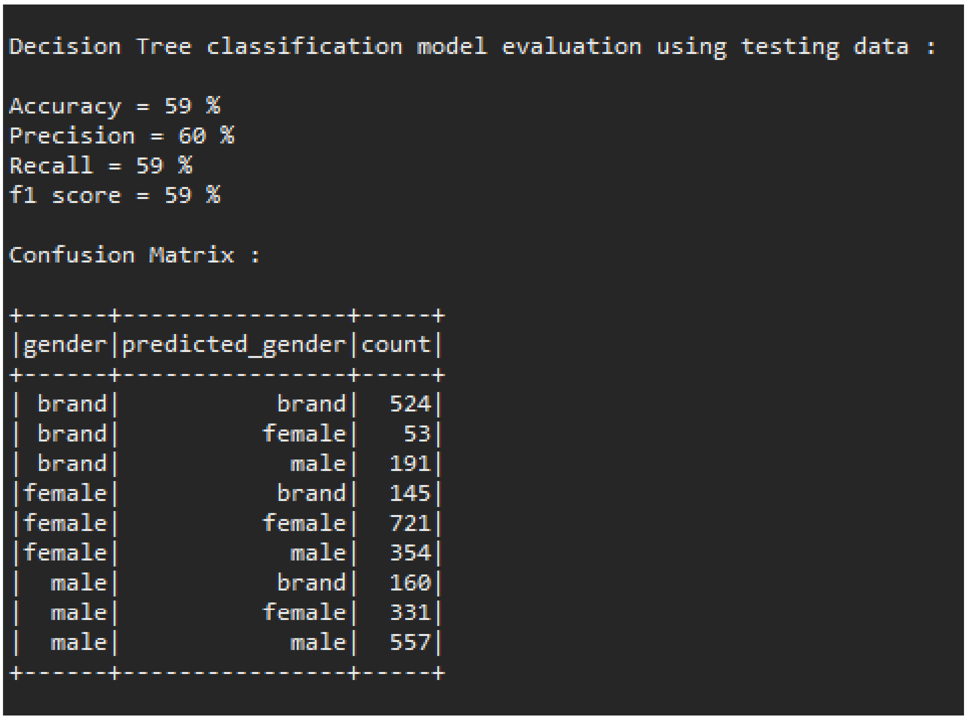
b. Showing below performance metrics and confusion matrix for **Decision Tree**:



```
Total records read from the file : 17426

Total records left after filtering : 10302

Total records in trainingData: 7266
Total records in testingData: 3036
```

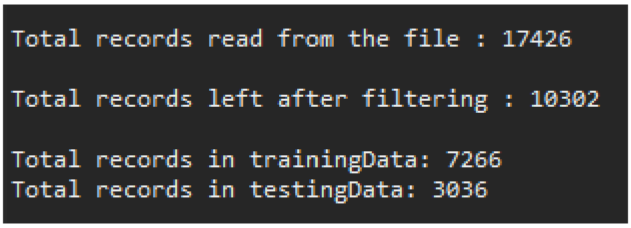


```
Decision Tree classification model evaluation using training data :

Accuracy = 61 %
Precision = 61 %
Recall = 61 %
f1 score = 61 %

Confusion Matrix :

+------+----------------+-----+
|gender|predicted_gender|count|
+------+----------------+-----+
| brand|           brand| 1230|
| brand|          female|  168|
| brand|            male|  418|
|female|           brand|  310|
|female|          female| 1774|
|female|            male|  811|
|  male|           brand|  376|
|  male|          female|  765|
|  male|            male| 1414|
+------+----------------+-----+
```

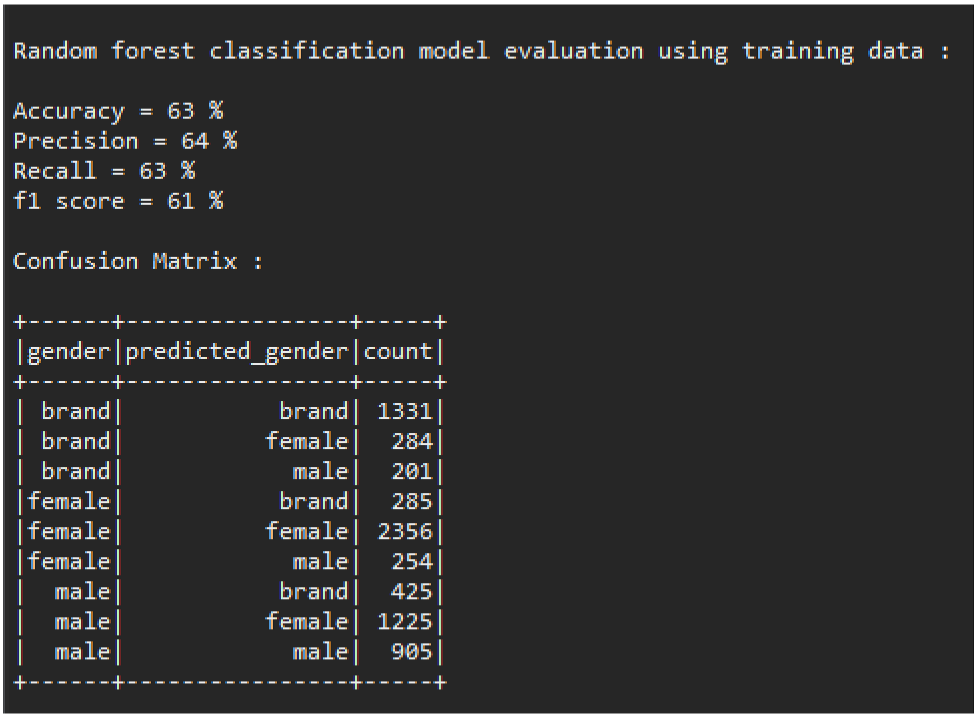


```
Decision Tree classification model evaluation using testing data :

Accuracy = 59 %
Precision = 60 %
Recall = 59 %
f1 score = 59 %

Confusion Matrix :

+------+----------------+-----+
|gender|predicted_gender|count|
+------+----------------+-----+
| brand|           brand|  524|
| brand|          female|   53|
| brand|            male|  191|
|female|           brand|  145|
|female|          female|  721|
|female|            male|  354|
|  male|           brand|  160|
|  male|          female|  331|
|  male|            male|  557|
+------+----------------+-----+
```

c. Showing below performance metrics and confusion matrix for **Random Forest**:

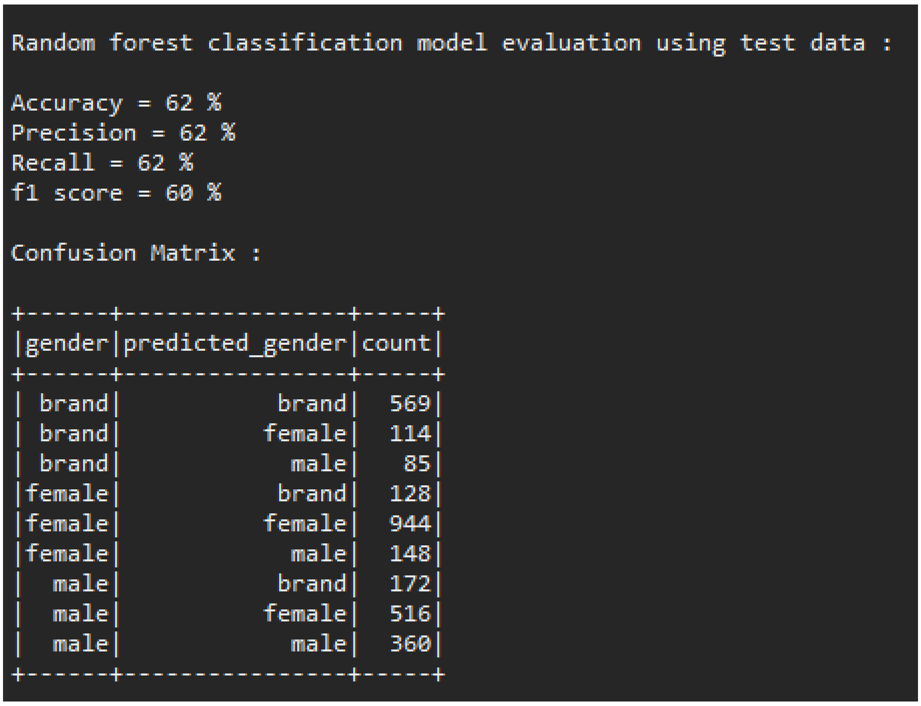


```
Total records read from the file : 17426

Total records left after filtering : 10302

Total records in trainingData: 7266
Total records in testingData: 3036
```



```
Random forest classification model evaluation using training data :

Accuracy = 63 %
Precision = 64 %
Recall = 63 %
f1 score = 61 %

Confusion Matrix :

+------+----------------+-----+
|gender|predicted_gender|count|
+------+----------------+-----+
| brand|           brand| 1331|
| brand|          female|  284|
| brand|            male|  201|
|female|           brand|  285|
|female|          female| 2356|
|female|            male|  254|
|  male|           brand|  425|
|  male|          female| 1225|
|  male|            male|  905|
+------+----------------+-----+
```

`



```
Random forest classification model evaluation using test data :

Accuracy = 62 %
Precision = 62 %
Recall = 62 %
f1 score = 60 %

Confusion Matrix :

+------+----------------+-----+
|gender|predicted_gender|count|
+------+----------------+-----+
| brand|           brand|  569|
| brand|          female|  114|
| brand|            male|   85|
|female|           brand|  128|
|female|          female|  944|
|female|            male|  148|
|  male|           brand|  172|
|  male|          female|  516|
|  male|            male|  360|
+------+----------------+-----+
```

d. I trained both models (Decision Tree and Random Forest) on training data and testing data both and **both models do not show any sign of overfitting or underfitting**. Difference between performance metrics for model trained on training data and testing data is less than 5% for both Decision Tree and Random Forest.

4. Inferences & Suggestions:

1) Advantages and Drawbacks of Decision Tree Algorithm:

Advantages:

a. Decision Trees are easy to interpret.
b. Decision trees are good at dealing with noisy or incomplete data.
c. Universal for solving both classification and regression problems.
d. Decision Trees are applicable for both continuous variables and categorical inputs.

Drawbacks:

a. Tree might get too large even after some pruning leading to instability. It can be difficult to control the size of the tree.
b. The high classification error rate while training set is small in comparison with the number of classes.
c. In some complex cases, splitting data into classes might not be helpful.

2) Advantages and Drawbacks of Random Forest Algorithm:

Advantages:

a. All advantages of Decision Tree algorithm are applicable to Random Forest algorithm too.
b. Random Forest Classification combines a group of average performing classifiers to form a good classifier. In the algorithm, we construct multiple decision trees by considering random subsets of data each time and finally take cumulative measure when we predict the results so we get more accurate results.
c. It is good for training even small samples and can be easily parallelized.
d. Powerful and Accurate.

Drawbacks:

a. All drawbacks of Decision Tree algorithm are applicable to Random Forest algorithm too.
b. It fails when there are rare outcomes or rare predictors.

3) Comparison:

Random Forest Algorithm seemed to be better based on the output received.

a. Less Prone to Overfitting – Difference between performance metrics on training data and testing data is less in comparison to difference between performance metrics on training data and testing data for Decision Tree.
b. More Accurate
c. Faster

4) Improvisation Techniques:

a. Pruning: It is a technique in machine learning that reduces the size of decision trees by removing sections of the tree that provide little power to classify instances. Pruning reduces the complexity of the final classifier, and hence improves predictive accuracy by the reduction of overfitting.
b. K-Fold Cross Validation: Cross validation in the training data itself can improve the performance of the model a bit.
c. Hybrid Model: Use a hybrid model, i.e. use logistic regression after using decision trees to improve performance.
d. Profile Image Processing: If we could have extracted profile images and processed those using artificial intelligence techniques then that would have improved accuracy of the models to great extent.

5) Choosing one among the two models:

Between the two models used - Decision Tree and Random Forest, I would choose **Random Forest.**

a. With a relatively small dataset in this assignment, performance metrics are better than Decision Tree.

b. Difference between performance metrics on trained data and testing data is less in comparison to difference between performance metrics on trained data and testing data for Decision Tree.

END OF DOCUMENT