Program Running Step:

Usage:
spark-submit --master local TwitterClassify-0.0.1-SNAPSHOT.jar input_dir_path

Eg:
spark-submit --master local TwitterClassify-0.0.1-SNAPSHOT.jar /Users/baskars/Upgrad/AnalyticsAssignment/gender-classifier-DFE-791531.csv

1. Data Processing Steps:

    a. List of data issues found in the raw data.
    - There are null values in the columns
    - The “text” column is multiline and some rows got garbage/ different end of line indicator
    - Need to get the “https” link presence info from text
    - Need to get calculate the num tweets per day count

    b. How did you tackle the issues mentioned above
    - Removed null values rows using “na().drop()” functions;
    - Used .option("escape", "\"") & .option("mode”,”DROPMALFORMED") to read the “text” column properly
    - Used “HttpText” UDF to get the link presence info
    - Used “TweetCntData” UDF to get the num tweets per day info

```
//UDF:Check for http link in the string and return true if found
else false
sparkSession.udf().register("HttpText", new UDF1<String, Boolean>()
{
private static final long serialVersionUID = 1L;
@Override
public Boolean call(String t1) throws Exception{
    return t1.contains("https");
   }
}, DataTypes.BooleanType);

// Create column "HttpLink" to indicate whether text column contain
link data or not
cleandata = cleandata.withColumn("HttpLink",
functions.callUDF("HttpText", cleandata.col("text")));

// Create new Column "NoOfDays" by taking diff of account creation
date to present date
cleandata = cleandata.withColumn(
"NoOfDays", functions.datediff(functions.current_date(),
functions.to_date(functions.unix_timestamp(cleandata.col("created"),
"MM/dd/yy").cast(DataTypes.TimestampType))));
```

```
// UDF:Calculate Tweet count per day by doing
// tweetCnt/noOfDays of input params
sparkSession.udf().register("TweetCntData", new UDF2<Integer,
Integer, Double>() {
private static final long serialVersionUID = 1L;
@Override
public Double call(Integer noOfDays, Integer tweetCnt) throws
Exception{
    if (noOfDays > 0) {
       return (double) (tweetCnt/noOfDays);
    }
    return 0.0;
  }
}, DataTypes.DoubleType);

// Create column "TweetCntPerDay" to indicate tweets per day for
each user
cleandata = cleandata.withColumn("TweetCntPerDay",
functions.callUDF("TweetCntData", cleandata.col("NoOfDays"),
cleandata.col("tweet_count")));
```

2. Model Building :

   a. Random Forest Classifcation:

      Data Columns:

      From Input csv:
        Gender, text, fav_number, created, tweet_count

      Derived Columns:
        IDFfeatures using text column
        HttpLink using text column
        TweetCntPerDay using tweet_count & created column

      HyperParameters:
        MaxDepth is set 10:
          - Setting this parameters increased the accuracy of model by 2%
          - Larger then 10 creates overfitting scenarios

```
//Assembling the features in the dataFrame as Dense Vector
VectorAssembler assembler = new VectorAssembler()
.setInputCols(new String[]
{"IDFfeatures","HttpLink","fav_number","TweetCntPerDay"})
.setOutputCol("features");

// Set up the Random Forest Model
RandomForestClassifier rf =
    new RandomForestClassifier().setMaxDepth(10);

// Create and Run Random Forest Pipeline
Pipeline pipelineRF = new Pipeline()
```

```
 .setStages(new PipelineStage[]
{labelindexer, tokenizer, remover, hashingTF, idf, assembler,
rf,labelConverter});
// Fit the pipeline to training documents.
PipelineModel modelRF = pipelineRF.fit(traindata);

// Make predictions on test documents.
Dataset<Row> predictionsTestDataRF = modelRF.transform(testdata);
```

b. Decision Tree Classification:

Data Columns:

From Input csv:

Gender, text, fav_number, created, tweet_count

Derived Columns:

IDFfeatures using text column
HttpLink using text column
TweetCntPerDay using tweet_count & created column

HyperParameters:

MaxDepth is set 10:

- Setting this parameters increased the accuracy of model by 1%
- Larger then 10 creates overfitting scenarios

```
//Assembling the features in the dataFrame as Dense Vector
VectorAssembler assembler = new VectorAssembler()
.setInputCols(new String[]
{"IDFfeatures","HttpLink","fav_number","TweetCntPerDay"})
.setOutputCol("features");

DecisionTreeClassifier dt = new
DecisionTreeClassifier().setMaxDepth(10);

// Create and Run Decision Tree Pipeline
Pipeline pipelineDT = new Pipeline()
.setStages(new PipelineStage[]
{labelindexer, tokenizer, remover, hashingTF, idf, assembler,
dt,labelConverter});

// Fit the pipeline to training documents.
PipelineModel modelDT = pipelineDT.fit(traindata);
Dataset<Row> predictionsDT = modelDT.transform(testdata);
```

3. Evaluation Metrics:

Random Forest Classification Metrics:

a. Accuracy, Precision, Recall, F1 Score:

Test Accuracy for Random Forest   = 0.5113842434011686
Train Accuracy for Random Forest. = 0.5711095843352799

Test F1 for Random Forest = 0.46912557407027267
Train F1 for Random Forest. = 0.5389596786167257

Test precision for Random Forest. = 0.5271534513305328
Train precision for Random Forest. = 0.6297683669516918

Test recall for Random Forest. = 0.5113842434011686
Train recall for Random Forest. = 0.5711095843352799

Confusion Matrix for Random Forest Classification:

gender	predictedLabel	count
male	brand	178
male	female	1241
female	female	1433
male	male	201
brand	brand	904
female	male	142
brand	male	127
female	brand	172
brand	female	565

b. Overfitting or under fitting:

I compared the metrics (Accuracy/F1/precision/recall) between
Training and testing data set to check for overfitting/under fit scenarios.
From the results above I see the model is overfit (around 6% difference in accuracy better testing and trained data).

Test Accuracy for Random Forest = 0.5113842434011686
Train Accuracy for Random Forest. = 0.5711095843352799

Decision Tree Classification:
a. Accuracy, Precision, Recall, F1 Score:

Test Accuracy for Decision Tree. = 0.5444287729196051
Train Accuracy for Decision Tree. = 0.5796976983854346

Test F1 for Decision Forest = 0.5334745939092491
Train F1 for Decision Forest = 0.5678569335290667

Test precision for Decision Forest = 0.5352113828571744
Train precision for Decision Forest = 0.5726290528753477

Test recall for Decision Forest. = 0.5444287729196051
Train recall for Decision Forest. = 0.5796976983854345

Decision Tree Confusion Matrix for Test Data :

```
+------+-------------+-----+
```

```
|gender|predictedLabel|count|
+------+--------------+-----+
|  male|         brand|  364|
|  male|        female|  727|
|female|        female| 1028|
|  male|          male|  529|
| brand|         brand| 1145|
|female|          male|  399|
| brand|          male|  191|
|female|         brand|  320|
| brand|        female|  260|
+------+--------------+-----+
```

b. Overfitting or under fitting:
I compared the metrics (Accuracy/F1/precision/recall) between
Training and testing data set to check for overfitting/under fit scenarios.
From the results above I see the model is overfit (around
3% difference in accuracy better testing and trained data).

```
Test Accuracy for Decision Tree.   = 0.5444287729196051
Train Accuracy for Decision Tree.  = 0.5796976983854346
```

4. Inferences & Suggestions:

a. Compare the results from the models and mention drawbacks and advantages for each of them.

Random Forest:
- The metrics show the model is overfitted
- From the confusion matrix the “female” gender
  Has more accuracy compare to other labels
- Overall accuracy is around 51%

Decision Tree:
- Overall accuracy is around 54.4%
- From the contusion matrix the accuracy is
  Distributed across the labels properly
- Compare to random forest it is less overfitted

b. Suggest some improvisation techniques to those models.

Random Forests:
- Can try to improve the metrics using other hyper params
  like setNumTrees()

Decision Tree:
- Can try to improve the metrics using setMinInfoGain
  Or setMinInfoGain

C. Choose from the two models present and justify why did you choose that particular model for the given problem statement.

Decision Tree is better in this case. Since this model has better accuracy and from the confusion matrix I see that the accuracy is consistent across the gender labels. In case of Random Forest the “female” gender is predicated more accurately but the percentage of “male” & “brand” Accuracy is low.

As the given statement “Gender recognition is essential and critical for many applications in the commercial domains. Imagine that Twitter needs to push advertisements based on gender”

It is important to get the model which is more accurate so that we can gain more commercial value out of it.