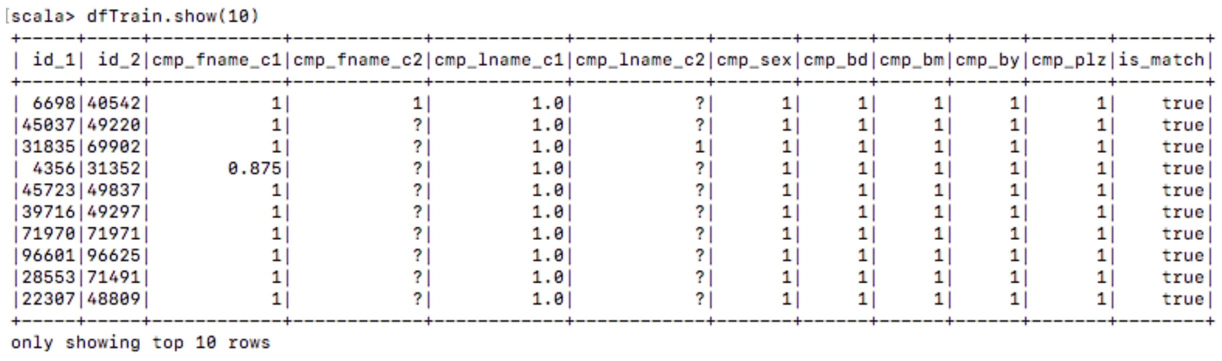
Domas Budrys A5 – CSCI5040

//Training: Block_2

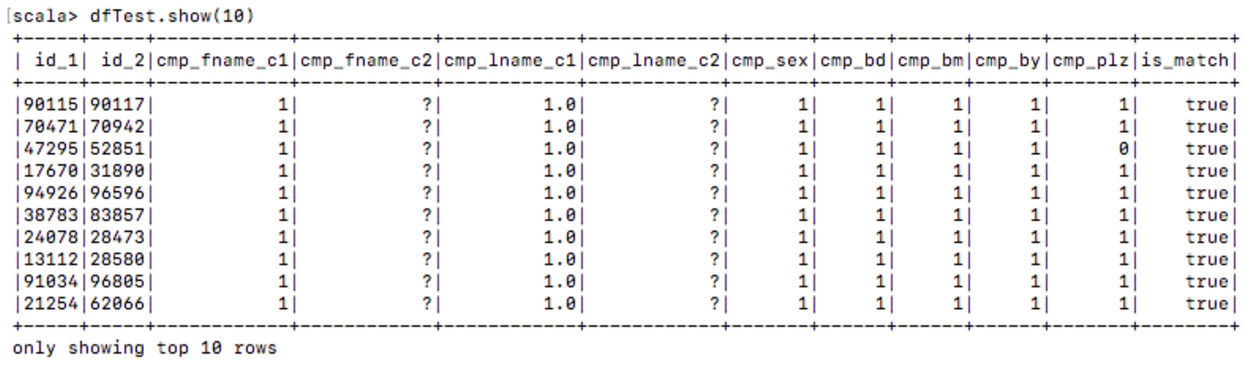
```
val fileNameTraining = "/Users/domo/Desktop/spark-
data/donation/block_2.csv"

val dfTrain = spark.read.option("inferSchema",
"true").option("header", "true").csv(fileNameTraining)
```



```
scala> dfTrain.show(10)
+-----+-----+------------+------------+------------+------------+-------+------+------+------+-------+--------+
| id_1| id_2|cmp_fname_c1|cmp_fname_c2|cmp_lname_c1|cmp_lname_c2|cmp_sex|cmp_bd|cmp_bm|cmp_by|cmp_plz|is_match|
+-----+-----+------------+------------+------------+------------+-------+------+------+------+-------+--------+
| 6698|40542|           1|           1|         1.0|           ?|      1|     1|     1|     1|      1|    true|
|45037|49220|           1|           ?|         1.0|           ?|      1|     1|     1|     1|      1|    true|
|31835|69902|           1|           ?|         1.0|           1|      1|     1|     1|     1|      1|    true|
| 4356|31352|       0.875|           ?|         1.0|           ?|      1|     1|     1|     1|      1|    true|
|45723|49837|           1|           ?|         1.0|           ?|      1|     1|     1|     1|      1|    true|
|39716|49297|           1|           ?|         1.0|           ?|      1|     1|     1|     1|      1|    true|
|71970|71971|           1|           ?|         1.0|           ?|      1|     1|     1|     1|      1|    true|
|96601|96625|           1|           ?|         1.0|           ?|      1|     1|     1|     1|      1|    true|
|28553|71491|           1|           ?|         1.0|           ?|      1|     1|     1|     1|      1|    true|
|22307|48809|           1|           ?|         1.0|           ?|      1|     1|     1|     1|      1|    true|
+-----+-----+------------+------------+------------+------------+-------+------+------+------+-------+--------+
only showing top 10 rows
```

//Test: Block_8

```
val fileNameTest= "/Users/domo/Desktop/spark-
data/donation/block_8.csv"

val dfTest = spark.read.option("inferSchema", "true").option("header",
"true").csv(fileNameTest)
```



```
scala> dfTest.show(10)
+-----+-----+------------+------------+------------+------------+-------+------+------+------+-------+--------+
| id_1| id_2|cmp_fname_c1|cmp_fname_c2|cmp_lname_c1|cmp_lname_c2|cmp_sex|cmp_bd|cmp_bm|cmp_by|cmp_plz|is_match|
+-----+-----+------------+------------+------------+------------+-------+------+------+------+-------+--------+
|90115|90117|           1|           ?|         1.0|           ?|      1|     1|     1|     1|      1|    true|
|70471|70942|           1|           ?|         1.0|           ?|      1|     1|     1|     1|      1|    true|
|47295|52851|           1|           ?|         1.0|           ?|      1|     1|     1|     1|      0|    true|
|17670|31890|           1|           ?|         1.0|           ?|      1|     1|     1|     1|      1|    true|
|94926|96596|           1|           ?|         1.0|           ?|      1|     1|     1|     1|      1|    true|
|38783|83857|           1|           ?|         1.0|           ?|      1|     1|     1|     1|      1|    true|
|24078|28473|           1|           ?|         1.0|           ?|      1|     1|     1|     1|      1|    true|
|13112|28580|           1|           ?|         1.0|           ?|      1|     1|     1|     1|      1|    true|
|91034|96805|           1|           ?|         1.0|           ?|      1|     1|     1|     1|      1|    true|
|21254|62066|           1|           ?|         1.0|           ?|      1|     1|     1|     1|      1|    true|
+-----+-----+------------+------------+------------+------------+-------+------+------+------+-------+--------+
only showing top 10 rows
```

Classifier where you use everything except the target ⬇⬇⬇

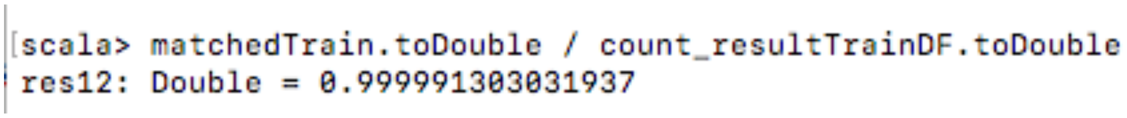
```
//combine train and test data sets into one
val combinedData = dfTrain.union(dfTest)

//imports
import org.apache.spark.ml.feature._
import org.apache.spark.ml.classification._


//Classifier to use everything
val supervised = new RFormula().setFormula("is_match ~ .")


val fittedRF = supervised.fit(combinedData)
val preparedCombinedData = fittedRF.transform(combinedData)



//Split prepared combined data set into train and test sets
//First, must get row count to determine how many rows will be defined
as Train set
val rowNum = dfTrain.count()

//Then
val trainSet = preparedCombinedData.limit(rowNum.toInt)
val testSet = preparedCombinedData.except(trainSet)


val lr = new LogisticRegression()
val fittedLR = lr.fit(trainSet)


val resultTrainDF = fittedLR.transform(trainSet)
```

//Results on Training data set

```
val matchedTrain = resultTrainDF.where("label == prediction").count()
val count_resultTrainDF =resultTrainDF.count()
matchedTrain.toDouble / count_resultTrainDF.toDouble
```

0.999991303031937

Screenshot:



```
scala> matchedTrain.toDouble / count_resultTrainDF.toDouble
res12: Double = 0.999991303031937
```

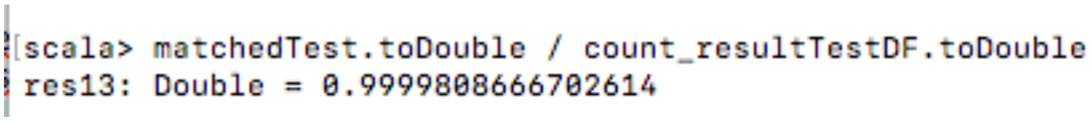
```
val resultTestDF = fittedLR.transform(testSet)
```

//Results on Test data set

```
val matchedTest = resultTestDF.where("label == prediction").count()
val count_resultTestDF = resultTestDF.count()
matchedTest.toDouble / count_resultTestDF.toDouble
```

0.9999808666702614

Screenshot:



```
scala> matchedTest.toDouble / count_resultTestDF.toDouble
res13: Double = 0.9999808666702614
```

Combine the two components of the German family name and combine the two components of the Germinal first name. ⬇⬇⬇

```
//Defying classifier
val supervised2 = new RFormula().setFormula("is_match ~
cmp_lname_c1:cmp_lname_c2 + cmp_fname_c1:cmp_fname_c2")




val fittedRF = supervised2.fit(combinedData)
val preparedCombinedData = fittedRF.transform(combinedData)



//Split prepared combined data set into train and test sets
//First, must get row count to determine how many rows will be defined
as Train set
val rowNum = dfTrain.count()

//Then
val trainSet = preparedCombinedData.limit(rowNum.toInt)
val testSet = preparedCombinedData.except(trainSet)


//val lr = new LogisticRegression()
```

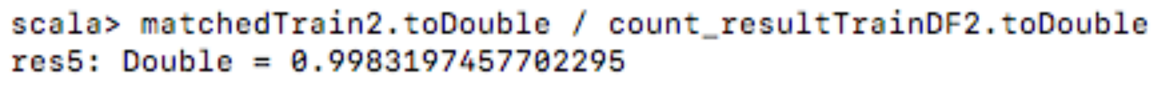
```
val fittedLR = lr.fit(trainSet)


val resultTrainDF = fittedLR.transform(trainSet)
```

//Results on Training data set

```
val matchedTrain2 = resultTrainDF.where("label == prediction").count()
val count_resultTrainDF2 =resultTrainDF.count()
matchedTrain2.toDouble / count_resultTrainDF2.toDouble
```

0.9983197457702295

Screenshot:



```
scala> matchedTrain2.toDouble / count_resultTrainDF2.toDouble
res5: Double = 0.9983197457702295
```

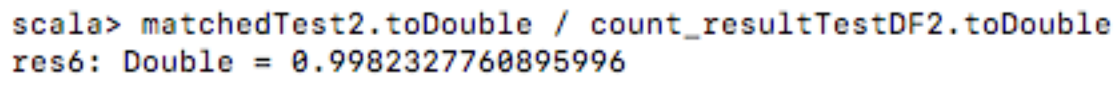
```
val resultTestDF = fittedLR.transform(testSet)
```

//Results on Test data set

```
val matchedTest2 = resultTestDF.where("label == prediction").count()
val count_resultTestDF2 = resultTestDF.count()
matchedTest2.toDouble / count_resultTestDF2.toDouble
```

0.9982327760895996

Screenshot:



```
scala> matchedTest2.toDouble / count_resultTestDF2.toDouble
res6: Double = 0.9982327760895996
```

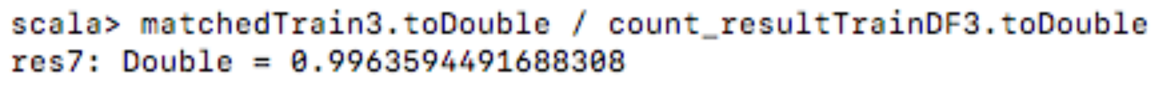
Combine each part of the birthday into one part.

```
//Defying classifier
val supervised3 = new RFormula().setFormula("is_match ~
cmp_bd:cmp_bm:cmp_by")




val fittedRF = supervised3.fit(combinedData)
val preparedCombinedData = fittedRF.transform(combinedData)



//Split prepared combined data set into train and test sets
//First, must get row count to determine how many rows will be defined
as Train set
val rowNum = dfTrain.count()

//Then
val trainSet = preparedCombinedData.limit(rowNum.toInt)
val testSet = preparedCombinedData.except(trainSet)


//val lr = new LogisticRegression()
```

```
val fittedLR = lr.fit(trainSet)


val resultTrainDF = fittedLR.transform(trainSet)
```

```
//Results on Training data set
val matchedTrain3 = resultTrainDF.where("label == prediction").count()
val count_resultTrainDF3 =resultTrainDF.count()
matchedTrain3.toDouble / count_resultTrainDF3.toDouble
0.9963594491688308
```

Screenshot:

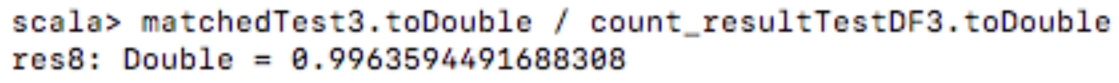


```
scala> matchedTrain3.toDouble / count_resultTrainDF3.toDouble
res7: Double = 0.9963594491688308
```

```
val resultTestDF = fittedLR.transform(testSet)
```

```
//Results on Test data set
val matchedTest3 = resultTestDF.where("label == prediction").count()
val count_resultTestDF3 = resultTestDF.count()
matchedTest3.toDouble / count_resultTestDF3.toDouble
0.9963594491688308
```

Screenshot:



```
scala> matchedTest3.toDouble / count_resultTestDF3.toDouble
res8: Double = 0.9963594491688308
```