IFT 598 Analysing Big Data

Professor Robert Rucker

Human Activity Recognition with Smartphones using Multinomial Logistic Regression

Name: Venkata Siva Abhishek Munukutla

ASU ID: 1213296732

Date: 23rd April, 2018

Introduction

Today, the smartphones in the market today are capable of recording data on the position of a human and predict activities such as running, walking and cycling. Almost all smartphones contain various kinds of sensors such as accelerometer, photo-detector, and proximity sensor for different purposes. Moreover, some of the sensors can be put to use to detect human activity.

Dataset:

The dataset "Human Activity Recognition with Smartphones" from Kaggle is being used in the project. The dataset contains 563 columns and more than 10k rows. Each row contains an attribute describing the activity of the person based on 561 attributes extracted from the smartphone. 561 columns that describe features like acceleration and angular velocity contain numeric values between -1 to +1. The columns have been scaled for the ease of use. The columns of the dataset describe the position of the phone in 3 dimensions based on its default calibration. The columns can be used to describe if the person is standing, sitting, laying, walking, walking upstairs and walking downstairs.

UC	UD	UE	UF	UG	UH	UI	UJ	UK	UL	UM	UN	UO	UP	UQ
fBodyBody	fBodyBody	fBodyBody	fBodyBody	fBodyBody	fBodyBody	angle(tBoc	angle(tBoc	angle(tBoc	angle(tBoc	angle(X,gra	angle(Y,gra	angle(Z,gra	subject	Activity
-0.99046	-0.87131	-1	-0.07432	-0.29868	-0.7103	-0.11275	0.0304	-0.46476	-0.01845	-0.84125	0.179941	-0.05863	1	STANDING
-0.99454	-1	-1	0.158075	-0.59505	-0.8615	0.053477	-0.00743	-0.73263	0.703511	-0.84479	0.180289	-0.05432	1	STANDING
-0.99376	-1	-0.55556	0.414503	-0.39075	-0.7601	-0.11856	0.177899	0.100699	0.808529	-0.84893	0.180637	-0.04912	1	STANDING
-0.99523	-0.9557	-0.93651	0.404573	-0.11729	-0.48284	-0.03679	-0.01289	0.640011	-0.48537	-0.84865	0.181935	-0.04766	1	STANDING
-0.99549	-1	-0.93651	0.087753	-0.35147	-0.69921	0.12332	0.122542	0.693578	-0.61597	-0.84787	0.185151	-0.04389	1	STANDING
-0.99453	-1	-1	0.019953	-0.54541	-0.84462	0.082632	-0.14344	0.275041	-0.36822	-0.84963	0.184823	-0.04213	1	STANDING
-0.99394	-0.9557	-1	0.145844	-0.2172	-0.56443	-0.21275	-0.23062	0.014637	-0.18951	-0.85215	0.18217	-0.04301	1	STANDING
-0.99542	-0.9557	-1	0.136382	-0.08231	-0.42172	-0.02089	0.593996	-0.56187	0.467383	-0.85102	0.183779	-0.04198	1	STANDING
-0.9951	-0.9557	-1	0.314038	-0.2694	-0.573	0.012954	0.080936	-0.23431	0.117797	-0.84797	0.188982	-0.03736	1	STANDING
-0.99046	-0.9557	-1	0.267383	0.339526	0.140452	-0.02059	-0.12773	-0.48287	-0.07067	-0.84829	0.19031	-0.03442	1	STANDING
-0.99156	-0.92345	-1	0.120503	0.348771	0.057682	0.080699	0.595791	-0.4758	0.115931	-0.85156	0.187609	-0.03468	1	STANDING
-0.99773	-1	-1	0.351442	-0.61101	-0.87836	0.001761	-0.06598	0.578861	-0.65195	-0.85272	0.18605	-0.03585	1	STANDING
-0.99907	-1	-0.87302	0.689897	-0.68639	-0.87875	-0.07755	-0.10122	0.639084	0.765485	-0.85065	0.187611	-0.036	1	STANDING
-0.99878	-1	-0.96825	0.740023	-0.5641	-0.7659	0.10562	-0.09028	-0.1324	0.498814	-0.84977	0.188812	-0.03506	1	STANDING
-0.99201	-0.92345	-1	0.130958	0.207689	-0.06805	0.062297	-0.05872	0.031208	-0.26879	-0.73094	0.283159	0.036444	1	STANDING
-0.99811	-1	-0.07937	0.66154	-0.78214	-0.95352	-0.12185	-0.02908	-0.01303	-0.05693	-0.7611	0.263119	0.024172	1	STANDING
-0.99827	-1	-0.90476	0.560668	-0.77888	-0.94042	-0.00145	-0.04811	-0.34047	-0.22915	-0.75917	0.264324	0.027014	1	STANDING
-0.99678	-1	-0.80952	0.428614	-0.3289	-0.59686	-0.02833	0.092367	-0.82224	0.367557	-0.75936	0.264033	0.029664	1	STANDING
-0.99843	-1	-1	0.348413	-0.5013	-0.83824	-0.16585	-0.03301	-0.24057	0.788193	-0.76105	0.262886	0.029346	1	STANDING

Figure 1. Screenshot of dataset

The objective is to predict the activity based on the features such as mean of body acceleration on X, Y and Z axes, standard deviation of acceleration on X, Y and Z axes, etc. The prediction can be made using logistic regression. However, to do so, K-means Clustering will be used to see if the features form groups that indicate a certain type of activity. Based

on the clusters, the features relevant to the activity can be used in logistic regression to predict that particular activity. The dataset is divided into two parts – training data and testing data. The ratio for distribution is 7:3.

Why do this at all?

The dataset contains reading from 30 smartphones. The column "Subject" suggests the same. However, the model of the smartphone and the name of the company have not been mentioned. Therefore, for the sake of simplicity, it is assumed that the smartphones belong to the same company and the same model.

Today, smartphones are being used for various purposes including tracking fitness activities like running, cycling, and walking. It is imperative that the smartphones are able to predict the activity properly. And to do so, there must be a tool that can provide an insight into the same for each smartphone of the company under the particular model.

Logistic Regression can be applied to the dataset to predict the activity of the human holding/carrying the smartphone. The data mining model can be used to capture the accuracy of the smartphone in detecting the activity. This gives the developer a scope for improvement in the smartphone.

Meaning, the data mining model can be used while testing the device to determine its accuracy in recognizing activity. The analysis can give either negative or positive results. If the analysis gives us negative results, i.e., if the results tell us that the predictions are inaccurate, the particular model of the smartphone is incapable of performing the said task. Moreover, budget should be allocated to improve the model. However, if the result is positive, the particular model of the smartphone is ready to go into the market for sale.

Workflow

1. Data loading
2. Data cleaning
3. Data exploration
4. Create a Multinomial Regression Model
5. Visualise the prediction model

Logistic Regression

Logistic regression is a statistical method for analysing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes).

Logistic slope coefficients can be interpreted as the effect of a unit of change in the X variable on the predicted logits with the other variables in the model held constant. That is, how a one unit change in X effects the log of the odds when the other variables in the model held constant

Therefore, initially, Logistic Regression will be applied to determine whether or not human is standing.

Multinomial-Logistic Regression

Multinomial Logistic Regression is the linear regression analysis to conduct when the dependent variable is nominal with more than two levels. Thus it is an extension of logistic regression, which analyzes dichotomous (binary) dependents.

After successfully applying Logistic Regression, MLR will be applied for human activity classification.

Limitations

The study is limited by the dataset as it contains reading from only 30 smartphones. Moreover, there is no information on which company and model each smartphone belongs to. It is assumed for the sake of simplicity that the smartphones belong to the same company and model. Moreover, to predict various activities, a multi-class classification model has to be developed. However, a simple logistic regression model is created to determine the probability of a particular type of activity. Initially, a model to predict whether or not the human is standing is determined.

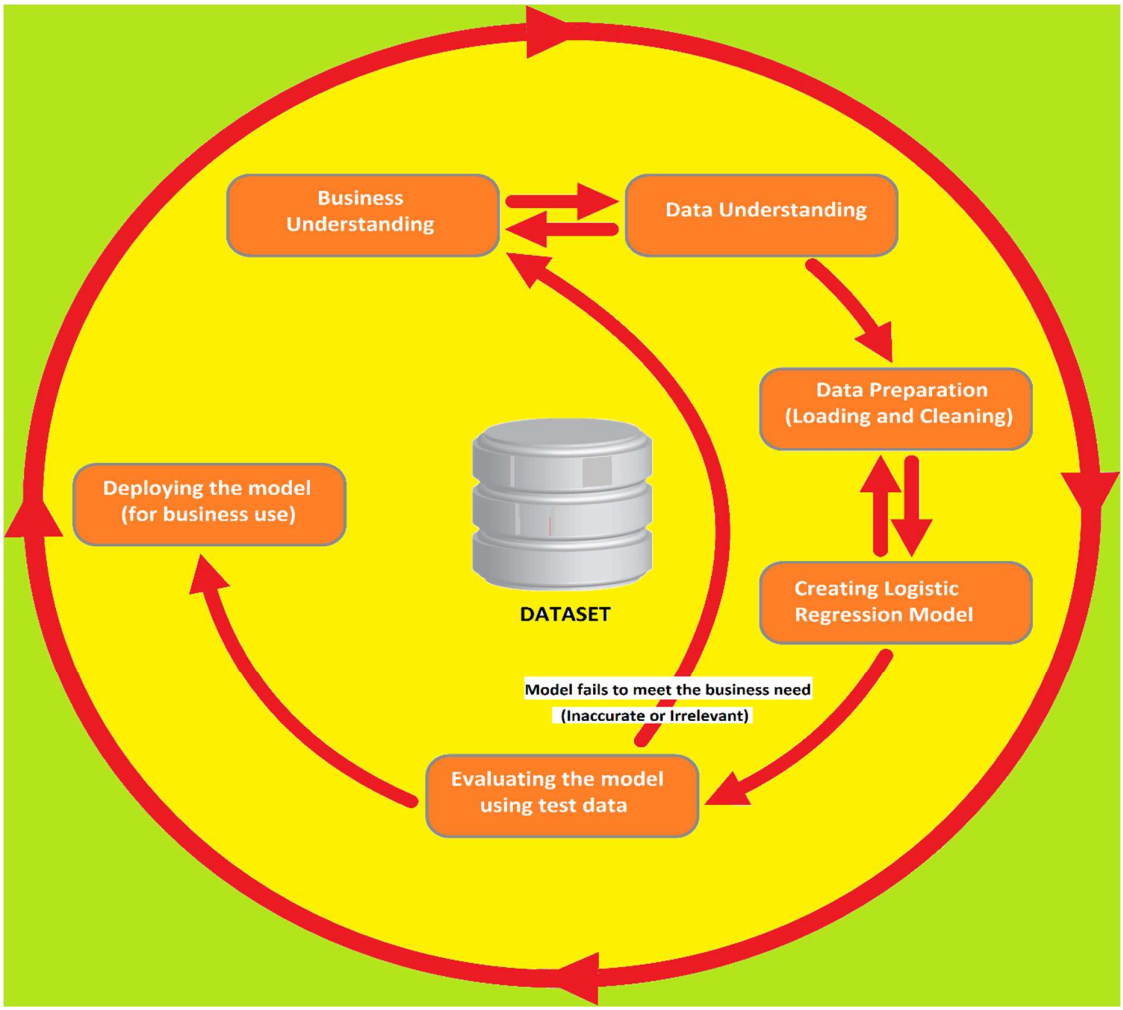
Scope

The data mining model can be used for testing different models of smartphones if massive amounts of relevant data is presented. As there are multiple activities, a multi-class classification model, i.e., a neural network can be created to predict the activity accurately. The model can be modified to apply to not just different kinds of smartphones but also the fitness apps to predict their accuracy.

Technologies used

Apache Spark and Scala will be used to accomplish the task of analysis. Spark's ML Lib will be used for creating the data mining model (Logistic Regression/ Multinomial Logistic Regression).

CRISP Diagram



OUTPUTS

Output for Binomial Logistic Regression:

Testing Data Stats:

Count: 2947

Correct: 2415

Wrong: 532

Wrong Percentage: 18.052256532066508

Correct Percentage: 81.94774346793349

Coefficients:

[3.3079642196754513, -2.790387818885029, 0.6891756516673883, -0.433021164014858, -0.3590531791892386, -0.13261396939088096, -0.5011165845648017, -0.36564862019219635, -0.17676211766421296, -0.294061397360343, -0.19037136572297622, -0.47653242415448416, ……. 0.6630254450659299, 0.3440621591169276, -1.9699655214993212, 6.084572166146445, 0.09259686269272807] [561 Coefficients]

Intercept: -1.496841170810312

Logit Function:

The logit function derived from the Coefficients and Intercept is shown below.

$$\ln(P) = 3.30796 * X_1 - 2.7903 * X_2 + \ldots\ldots\ldots + 0.09259 * X_{560} - 1.49684$$

Sample Output:

```
+--------------------+-----+
|             Feature|_c561|
+--------------------+-----+
|[0.25717778,-0.02...|    1|
|[0.28602671,-0.01...|    1|

|[0.27252874,-0.02...|    1|
|[0.2757457,-0.010...|    1|
|[0.27859589,-0.01...|    1|
|[0.27915178,-0.02...|    1|
|[0.27454383,-0.02...|    1|
|[0.26906631,-0.02...|    1|
|[0.28016952,-0.01...|    1|
|[0.27746102,-0.01...|    1|
|[0.28837469,-0.00...|    1|
|[0.28590496,-0.00...|    1|
|[0.29648714,-0.01...|    0|
```

|[0.27723542,-0.02...| 0|

Output for Multinomial Logistic Regression:

Count of Each Activity in Training Data:

```
+------------------+-----+
|              _c561|count|
+------------------+-----+
|            LAYING| 1407|
|WALKING_DOWNSTAIRS|  986|
|           WALKING| 1226|
|          STANDING| 1374|
|  WALKING_UPSTAIRS| 1073|
|           SITTING| 1286|
+------------------+-----+
```

Test Error = 0.6217355821545159

Coefficients:

6 x 561 CSCMatrix

(4,16) 0.008887688388606223

(1,40) -0.1350549840631362

(1,49) -0.1360835942894135

(1,52) -0.1366634090343332

(1,56) -0.5317850814183785

(1,57) 0.006057968438755048

(4,281) 0.008997185348537067

(4,302) 0.0077512205828994835

(4,310) 0.008822934601012775

(4,314) 0.008900611887314676

(1,333) -0.003930149300485898

(1,334) -0.011075129454884458

(1,343) -0.0062726954922294265 ……

Intercepts:

[0.12139393481512424,

0.1479055104917021,

0.0553408275799166,

-0.12537326050825578,

-0.2069251933388115,

0.007658180960324274]

Sample Prediction:

+--------------------+-----+

| Feature|_c561|

+--------------------+-----+

|[0.28858451,-0.02...| 0|

|[0.27841883,-0.01...| 0|

|[0.27965306,-0.01...| 0|

|[0.27917394,-0.02...| 0|

|[0.27662877,-0.01...| 0|

|[0.27719877,-0.01...| 0|

|[0.27945388,-0.01...| 0|

|[0.27743247,-0.03...| 0|

|[0.27729342,-0.02...| 0|

|[0.28058569,-0.00...| 0|

|[0.27688027,-0.01...| 0|

|[0.27090823,-0.01...| 2|

|[0.2795534,-0.017...| 2|

|[0.27635864,-0.01...| 2|

|[0.23715407,0.007...| 2|

|[0.2731084,0.0065...| 2|

|[0.28150539,-0.01...| 2|

|[0.27843225,-0.01...| 2|

|[0.2733796,-0.009...| 2|

|[0.27905615,-0.01...| 2|

|[0.2785991,-0.019...| 2|

.

.

[0.27588359,-0.01...| 1|

|[0.27783021,-0.01...| 1|

|[0.27937115,-0.01...| 1|

.

.

|[0.28202157,-0.03...| 5|

|[0.25584075,-0.06...| 5|

|[0.25486723,0.003...| 5|

|[0.34337048,-0.01...| 5|

|[0.27623973,-0.02...| 5|

|[0.25546822,0.021...| 5|

|[0.32113983,0.001...| 5|

.

.

|[0.28431748,-0.02...| 5|

|[0.22172674,-0.02...| 5|

+--------------------+-----+

only showing top 100 rows

Conclusion:

In this project, I attempted to test the accuracy of a cell/mobile phone prediction of human activity with the help of Multinomial Logistic Regression. The dataset contains 561 columns and over 10000 records. The columns describe features like acceleration and angular velocity contain numeric values between -1 to +1. The columns have been scaled for the ease of use. The columns of the dataset describe the position of the phone in 3 dimensions based on its default calibration. With the help of these values, human activities like sitting, standing, walking, etc. were classified.

Initially, I used Binomial Logistic Regression to classify the activity into either STANDING or NOT STANDING. I was able to achieve 81% accuracy. The model was based on the following equation.

$$\ln (P) = 3.30796 * X_1 - 2.7903 * X_2 + \ldots\ldots \ldots + 0.09259 * X_{560} - 1.49684$$

Later, I attempted to classify the activity into six categories. However, I ran into nearly 62 % inaccuracy. This made me think that the model could not fit the given data. I will have to use a different Classification Models to achieve the objective.

If I am successful at maximizing the accuracy, I can classify the activity of a person based on the various parameters. This makes the cell/mobile phone equipped with the functionality of predicting the person's activity and thereby helping the person track his/her daily activity efficiently.

Code for Multinomial Logistic Regression:

```
import org.apache.spark.sql.{DataFrame, Dataset}
import org.apache.log4j.{Level, Logger}
import org.apache.spark.ml.Pipeline
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.{DataFrame, Dataset}
import org.apache.spark.ml.classification.{LogisticRegression, MultilayerPerceptronClassifier}
import org.apache.spark.ml.evaluation.{BinaryClassificationEvaluator, MulticlassClassificationEvaluator}
import org.apache.spark.ml.feature.VectorAssembler


object proj {

  def main(args: Array[String]): Unit = {
    Logger.getLogger("org").setLevel(Level.OFF)

    val spark = SparkSession.builder
      .master("local[*]")
      .appName("Project")
      .getOrCreate()

    import spark.implicits._
    println(s" Spark version , ${spark.version} ")

    val sc = spark.sparkContext

    val fn = "D:/train.csv"
    val regressDF = spark.read
      .format("csv")
      .option("header", "true") // Use first line of all files as header
      .option("inferSchema", "true") // Automatically infer data types
      .load(fn)

    import org.apache.spark.sql.functions._
```

```
regressDF.groupBy("_c561").count().show()

def udfAct = udf((_c561: String) => {
  _c561 match {
    case t if t == "STANDING" => 0
    case t if t == "LAYING" => 1
    case t if t == "SITTING" => 2
    case t if t == "WALKING_UPSTAIRS" => 3
    case t if t == "WALKING_DOWNSTAIRS" => 4
    case t if t == "WALKING" => 5
  }})


val newdf = regressDF.withColumn("_c561", udfAct(regressDF("_c561")))

newdf.show(50)


import org.apache.spark.ml.feature.VectorAssembler
val assembler = new VectorAssembler()

.setInputCols(Array("_c0","_c1","_c2","_c3","_c4","_c5","_c6","_c7","_c8","_c9","_c10","_c11","_c12","_c
13","_c14","_c15","_c16","_c17","_c18","_c19","_c20","_c21","_c22","_c23","_c24","_c25","_c26","_c27",
"_c28","_c29","_c30","_c31  …… ….. …. ,"_c554","_c555","_c556","_c557","_c558","_c559","_c560"))
  .setOutputCol("Feature")
val inputFormat = assembler.transform(newdf)


val lr = new LogisticRegression()
  .setMaxIter(10)
  .setRegParam(0.3)
  .setElasticNetParam(0.8)
  .setFeaturesCol("Feature")
  .setLabelCol("_c561")



val lrModel = lr.fit(inputFormat)
println(s"Coefficients: \n${lrModel.coefficientMatrix}")
println(s"Intercepts: ${lrModel.interceptVector}")


  val fn1 = "D:/test.csv"
  val regressDF2 = spark.read
    .format("csv")
```

```
          .option("header", "true") // Use first line of all files as header
          .option("inferSchema", "true") // Automatically infer data types
          .load(fn1)



        val newdf2 = regressDF.withColumn("_c561", udfAct(regressDF("_c561")))

         import org.apache.spark.ml.feature.VectorAssembler
         val assembler1 = new VectorAssembler()

.setInputCols(Array("_c0","_c1","_c2","_c3","_c4","_c5","_c6","_c7","_c8","_c9","_c10","_c11","_c12","_c
13","_c14","_c15","_c16","_c17","_c18 ”  ……. ,"_c555","_c556","_c557","_c558","_c559","_c560"))
          .setOutputCol("Feature")
         val inputFormat1 = assembler1.transform(newdf2)

         inputFormat1.show(30)


         val predictions = lrModel.transform(inputFormat1)
         val pl = predictions.select("Feature","_c561")
         pl.show(100)
    val evaluator = new MulticlassClassificationEvaluator()
      .setLabelCol("_c561")
      .setPredictionCol("prediction")
      .setMetricName("accuracy")
    val accuracy = evaluator.evaluate(predictions)
    println("\nTest Error = " + (1.0 - accuracy))

  }
 }
```

Code for Binomial Logistic Regression:

```
import org.apache.spark.sql.{DataFrame, Dataset}
import org.apache.log4j.{Level, Logger}
import org.apache.spark.ml.Pipeline
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.{DataFrame, Dataset}
import org.apache.spark.ml.classification.{BinaryLogisticRegressionSummary, LogisticRegression,
MultilayerPerceptronClassifier}
import org.apache.spark.ml.evaluation.{BinaryClassificationEvaluator, MulticlassClassificationEvaluator}
import org.apache.spark.ml.feature.VectorAssembler
```

```
object proj2 {

  def main(args: Array[String]): Unit = {
    Logger.getLogger("org").setLevel(Level.OFF)

    val spark = SparkSession.builder
      .master("local[*]")
      .appName("Project")
      .getOrCreate()

    import spark.implicits._
    println(s" Spark version , ${spark.version} ")

    val sc = spark.sparkContext

    val fn = "D:/train.csv"
    val regressDF = spark.read
      .format("csv")
      .option("header", "true") // Use first line of all files as header
      .option("inferSchema", "true") // Automatically infer data types
      .load(fn)

    import org.apache.spark.sql.functions._


    regressDF.groupBy("_c561").count().show()


    val newdf = regressDF.withColumn("_c561", when($"_c561" === "STANDING", 1).otherwise(0))

    newdf.show(10)


    import org.apache.spark.ml.feature.VectorAssembler
    val assembler = new VectorAssembler()

.setInputCols(Array("_c0","_c1","_c2","_c3","_c4","_c5","_c6","_c7","_c8","_c9","_c10","_c11","_c12","_c
13","_c14","_c15","_c16","_c17","_c18","_c19","_c20", … …. “_c556","_c557","_c558","_c559","_c560"))
      .setOutputCol("Feature")
    val inputFormat = assembler.transform(newdf)


    val lr = new LogisticRegression()
      .setFeaturesCol("Feature")
      .setLabelCol("_c561")
```

```
   val lrModel = lr.fit(inputFormat)
   println(s"Coefficients: ${lrModel.coefficients} \nIntercept: ${lrModel.intercept}")


   val fn1 = "D:/test.csv"
   val regressDF2 = spark.read
     .format("csv")
     .option("header", "true") // Use first line of all files as header
     .option("inferSchema", "true") // Automatically infer data types
     .load(fn1)


   val newdf2 = regressDF2.withColumn("_c561", when($"_c561" === "STANDING", 1).otherwise(0))

   import org.apache.spark.ml.feature.VectorAssembler
   val assembler1 = new VectorAssembler()

.setInputCols(Array("_c0","_c1","_c2","_c3","_c4","_c5","_c6","_c7","_c8","_c9","_c10","_c11","_c12","_c
13","_c14","_c15","_c16","_c17","_c18","_c19","_c20","_c21","_c22","_c23","_c24","_c25","_c26","_c27",
"_c28", …  ….  "_c555","_c556","_c557","_c558","_c559","_c560"))
     .setOutputCol("Feature")
   val inputFormat1 = assembler1.transform(newdf2)

   inputFormat1.show(30)

   val predictions = lrModel.transform(inputFormat1)
   val pl = predictions.select("Feature","_c561")
   pl.show(100)

   val lp = predictions.select( "_c561", "prediction")
   val counttotal = predictions.count()
   println(s"Count: ${counttotal}")
   val correct = lp.filter($"_c561" === $"prediction").count()
   println(s"Correct: ${correct}")
   val wrong = lp.filter(not($"_c561" === $"prediction")).count()
   println(s"Wrong: ${wrong}")
   val ratioWrong=wrong.toDouble/counttotal.toDouble
   println(s"Wrong Percentage: ${ratioWrong*100}")
   val ratioCorrect=correct.toDouble/counttotal.toDouble
   println(s"Correct Percentage: ${ratioCorrect*100}")

  }
}
```