AFONSO Benjamin
CRINON Quentin
RICOEUR Bastien
ZAMBRANO Nicolas
IG5

WIBrave

Web Intelligence

Tabmo

Under the direction of Mr.Castelltort

Summary

1. **Data cleaning process** **2**

2. **Data visualization** **4**

3. **Training process and Algorithms used** **5**
    a. Logistic Regression **5**
    b. Decision Tree **5**
    c. Random Forest **6**

4. **Results & post mortem** **7**
    a. Results **7**
    b. Post mortem **7**

1. Data cleaning process

Having imported in a dataframe the set of data supplied by Tabmo (in JSON format), we analyzed the data to be treated. We tried to understand the sense of each column of our dataframe to look for the most significant features and for the one that could be ignored.

Once we have analyzed the content of the data, we noticed that certain fields, for example the OS, could not be exploitable before it has been cleaned. So we established a process of several transformations on the dataframe to have the most exploitable set of data possible for our algorithm of prediction.

Here is the series of transformations realized on the data :

- **OS :** This field informs us about the OS of the device used by the user when he saw the advertisement. We quickly noticed that the values were predominantly Androïd or iOS compare to some other devices like Blackberry, WindowsPhone for example. We decided to group these others OS in a single category “others” because we thought that they would not have a bit impact on the prediction if they are not grouped and they would not help us to detect a particular trend. We had to also unify the fields because, for example, iOS was written in different ways (ios, IOS ...).
- **label :** This field informs about the fact that a user clicked on the advertisement or not, it is this field which we shall have to predict. We decided to transform the values boolean by numerical values that we think it more relevant for the algorithms of prediction. Thus the values "true" becomes 1.0 and in the other case 0.0.
- **bidfloor :** It represents the minimum bid for the advertisement. The value is a float, but as there are a million of rows there can be a million of different bidfloors. So we thought that group the bidfloors in intervals would help us to have a more relevant prediction.
- **timestamp :** This field representing a date is inexploitable as is, so we turned it into date and thought that it would be useful to keep only the hour, to observe trends according to the hours of the day.
- **sizeBanner :** This is the size of the advertisement. We had a lot of size different, so we thought that we could classify all these sizes in three categories : *small, medium* and *big*.
- **interests :** This field contains values like “IAB1” or “IAB2-15”, corresponding to categories. We thought that it will not be useful to keep all these categories but it would be better if we group by the main category. So each sub-category “IAB1-X” (with X an integer) became “IAB1” which correspond to the main category.
- **media :** This field is encrypted but as it contains only two different values, we supposed that media means the type of the advertisement. We have decided to not transform this field.

- **publisher :** This field represents the entity that controls the content, in the dataset we have more than 600 different publishers. We have decided to not transform this field too.

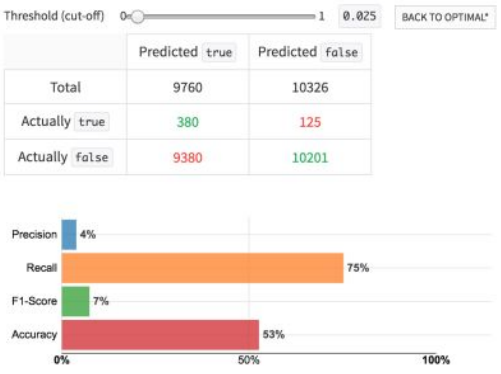
We also add a column "**classWeightCol**" that serves as a weighting for labels, we wanted rows with labels equal to 1 to count as much as rows with labels equal to 0. We use this column for our training dataset when we used logistic regression.

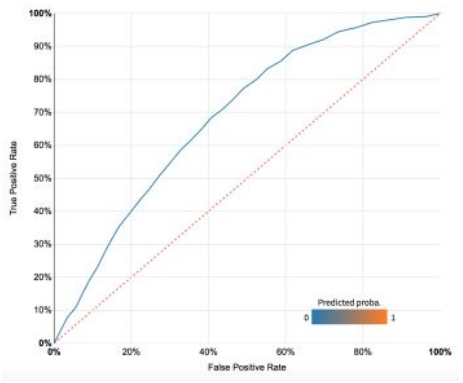
2. Data visualization

Before launching into the implementation of our cleaning protocol of the data and realizing algorithms of prediction with Scala and Spark, we found interesting to use beforehand a tool of data analysis which we know : Dataiku. We were able to realize easily the transformations that we had planned on our data and to see if they were sensible. We were also able to test various algorithms of prediction by varying the parameters to have an idea of the algorithm which we thought optimal for the set of data. This approach allowed us to have an axis of development for our project Spark.

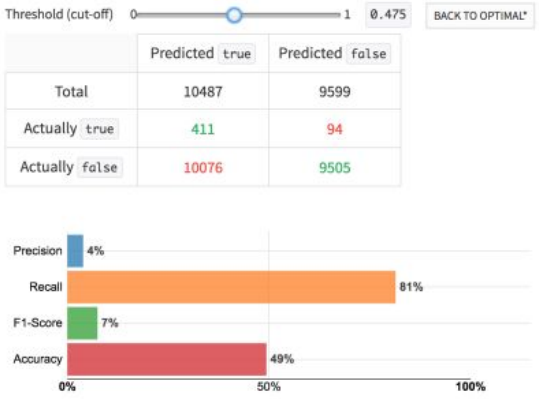
You can see below, the confusion matrix and roc curve which we had with a linear regression and a random forest on Dataiku:

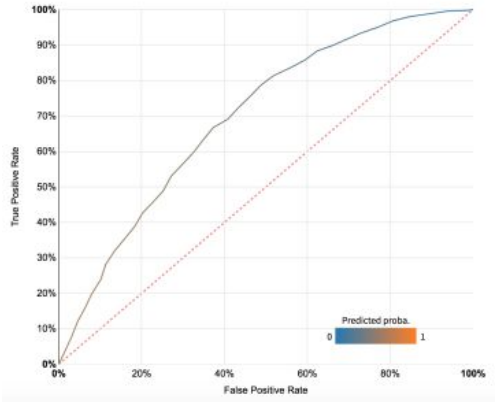
Linear Regression





RandomForest





3. Training process and Algorithms used

To do our predictive model we tried to implement three different supervised classification algorithms. To train our model we decided to split our data set into 70% for the training part and 30% for the testing part.

Our goal was to get the highest recall possible, we didn't want to miss any click. Of course the precision was also important, but if we would have predicted all the rows to false (no click) we would have got a nice precision even if we predict nothing, that's why we were focus on the recall.

a. Logistic Regression

As Logistic Regression is one of the most popular supervised algorithm to predict categorical response we decided to implement it. To do this we used the column "**classWeightCol**" we created in the cleaning process. But with this algorithm gave us the same probability for all the rows (~50%) so our prediction was based solely on randomness.

label_indexed	prediction	probability	rawPrediction
1.0	0.0	[0.54498983317908...	[0.18044737640928...
1.0	0.0	[0.54498983317908...	[0.18044737640928...
1.0	0.0	[0.54498983317908...	[0.18044737640928...
1.0	0.0	[0.54498983317908...	[0.18044737640928...
1.0	0.0	[0.54498983317908...	[0.18044737640928...
1.0	0.0	[0.54498983317908...	[0.18044737640928...
1.0	0.0	[0.54498983317908...	[0.18044737640928...
1.0	0.0	[0.54498983317908...	[0.18044737640928...
1.0	0.0	[0.54498983317908...	[0.18044737640928...
1.0	0.0	[0.54498983317908...	[0.18044737640928...
1.0	0.0	[0.54498983317908...	[0.18044737640928...

After multiples tests, we tried many configurations like taking the same number of rows with label equal to 1 and label equal to 0, for our training dataset. But our predictions were always based on randomness, we were getting only 1 or 0 for all the rows of our prediction.

b. Decision Tree

As the Logistic regression was not a success for us, we decided to implement another supervised classification algorithm : decision trees. The power of decision trees is that they are easy to interpret and can handle categorical features so it was fitting our project. We obtained much more conclusive results than with the linear regression, but they were still mediocre, a low recall, and a precision almost nonexistent. That's why we decided to implement Random Forest, which is based on decision trees.

c. Random Forest

This supervised algorithm is based on the construction of multiple decision trees. It performs learning on multiple decision trees driven on slightly different subsets of data.
This algorithm gave us our best results, it maximized the recall (~60%) and get the highest precision possible (~3%). To get this we used the following configurations :

```
val randomForest = new RandomForestClassifier()
  .setLabelCol(label)
  .setFeaturesCol(features)
  .setNumTrees(50)
  .setImpurity("gini")
  .setMaxBins(350)
```

We chose to do our Random Forest on 50 trees because the time of execution is rapid, and results are interesting. For the impurity, as we have a classification algorithm we had the choice between “gini” and “entropy”, they gave us approximately the same results but the entropy is a little bit slower to compute. For the maximum number of bins used we have 350, because that the maximum number of features of our dataset.

4. Results & post mortem

a. Results

This is the best metrics we got on 1000 rows:

Execution time (training model)	97925 ms
Execution time (Prediction)	12227 ms
True positive	19
False positive	9
True negative	368
False negative	604
Threshold 1.0, Recall	67.85%
Threshold 1.0, Precision	03.04%

We know that the metrics are not so good, but we didn't find the configuration or the algorithm to get better metrics. In our prediction we missed 9 clicks, it's a lot as the recall can say, only ~68% of recall is not enough to have a good click prediction algorithm. We tried to change the configurations, the threshold for example, but we failed to get better results.

b. Post mortem

During our cursus at Polytech we acquired theoretical knowledge in ETL, Statistics, etc, and this project allowed us to have a first real experience in Data Science domain. We have confronted with the real issues and the problems of an analysis of a voluminous set of data.

With Dataiku, whom we had already used for projects, we had results more decisive than those of our projects. We are disappointed to fail to get similar results with Spark and Scala, it is maybe understandable by a lack of experience in Data Science. We took a while before really knowing what we are going to make in term of development. We had theoretical knowledge bug we had difficulty to apply it.