Ameesh Baxi

ameeshbaxi@gmail.com

Abstract

The milestone report of the capstone project

MileStone Report

Data Science Workshop

Contents

[Introduction: 2](#_Toc472974256)

[Problem to be solved: 2](#_Toc472974257)

[Project Client: 2](#_Toc472974258)

[Data Set: 2](#_Toc472974259)

[Data Fields: 2](#_Toc472974260)

[Anonymized Categorical Variables: 3](#_Toc472974261)

[Data Exploration: 3](#_Toc472974262)

[Dealing with data: 3](#_Toc472974263)

[Data Wrangling: 3](#_Toc472974264)

[Data Story: 3](#_Toc472974265)

[Modelling: 3](#_Toc472974266)

[Approach: 3](#_Toc472974267)

[Developing Models on Local Machine: 4](#_Toc472974268)

[Sampling Training Data: 4](#_Toc472974269)

[The Right Model: 4](#_Toc472974270)

[Developing Various Models: 4](#_Toc472974271)

[Sampling Test Data: 5](#_Toc472974272)

[Validating the Classifier with Test Data: 5](#_Toc472974273)

[Visualizing Different Data Set With Different Classifier: 5](#_Toc472974274)

# Introduction:

This is a Capstone Project Milestone Report document outlining the problem, the data-set and the approach to solve the problem. The document will discuss the various approaches and predicts the best classifier for this problem. However, please look at the project jupyter notebook for implementation and detailed results or different classifiers.

# Problem to be solved:

The project is to predict whether a mobile ad will be clicked or not. Click-Through-Rate (CTR) metric is used for evaluating ad performance and CTR systems are widely used by internet economy. The problem and solutions are applicable to current internet economy. The problem provides quite a good learning opportunity with a typical setup in real word with lots of data and lots of features.

# Project Client:

The project is completed Kaggle competition. There is already a solution available for the project and reward has been awarded to the winning team.

The project client is myself to see if I am able to apply advanced Machine Learning technologies and Data Science methodologies learned in this workshop to arrive to a solution that would match up to the top teams in the competition. Since the solution is available, there is a certainty to solution that one should arrive.

As mentioned before, the main aim here is to apply learning of this course to a real life problem to 1) refine the learnings further 2) demonstrate the learnings and 3) be ready to apply the learnings to next projects.

# Data Set:

The data-set is available [here](https://www.kaggle.com/c/avazu-ctr-prediction/data). The data-set contains training and test data. The data-set is in csv format. The zipped training data, is about 1 Gb in size and test data-set is about 118 Mb. The data-set is well defined with some of the categorical features anonymized.

## Data Fields:

The data-set contains the following fields.

* id: ad identifier
* click: 0/1 for non-click/click
* hour: format is YYMMDDHH, so 14091123 means 23:00 on Sept. 11, 2014 UTC.
* C1 -- anonymized categorical variable
* banner\_pos
* site\_id
* site\_domain
* site\_category
* app\_id
* app\_domain
* app\_category
* device\_id
* device\_ip
* device\_model
* device\_type
* device\_conn\_type
* C14-C21 -- anonymized categorical variables

## Anonymized Categorical Variables:

Most of the fields are self-explanatory around mobile ad being clicked or not. However, there are anonymized categorical variables. It would have been good to get an idea about these variables. Not knowing the name or significance of these fields, any data engineering could not be applied to these fields and they have to be taken as they are given.

# Data Exploration:

## Dealing with data:

The training data-set in csv format is about 6 Gb. The test data-set is about 650 Mb. The work computer is only has 8 Gb RAM. For Exploration, the training data-set was converted into an sqlite database. The data will be explored using different sql queries.

## Data Wrangling:

The following cleaning-up and / or modification had to be done before the data-set could be utilised.

1. Ensure that there is no missing value.
2. The data format was not really usable. For exploration purpose, the date was converted into Pandas datetime format. For modelling purpose, the date was converted into ordinal format.
3. Some of the id fields were in hex format. They were converted into Integer format.
4. Id filed of the data-set will not be used.
5. Click column represented the classification if an ad was clicked or not.

## Data Story:

Please visit [this link](https://github.com/ameeshbaxi/data_science_workshop/blob/master/data_story/data_story.ipynb) to see the data story on data exploration of this data-set. The highlights of the data story are given below.

1. From the data-set, the Click-Through-Rate is about 16.98%. The Click-Through-Rate is successfully clicked ad from the total data-set. As explained in the field, when Click value is greater than zero, an ad is clicked.
2. The training data-set contains about 40+ million entries with 22 features affecting the prediction.
3. With the large number of entries, only about 17% data is classified as positive class and remaining are in negative class. The negative class values are about 5 times more than positive class value. The data-set presents Class imbalance problem.
4. With various exploration, at times it seems that day and time may not have much impact. However, the volume of data on a particular day may have an impact. It would be hard to tell if there is a real impact or not without running through various models.
5. As discussed above, all features shall be considered for the modelling purpose.

# Modelling:

## Approach:

The local work machine has only 8 Gb RAM. The training data-set itself is 6 Gb in size. In order to work locally and explore different models, a data sample should be created. The various models / code will be developed using on one sample. Once, the code is ready, run these models on entire data-set and see if the models produce the right result.

## Developing Models on Local Machine:

### Sampling Training Data:

* After playing with cleaned up data-set, it was observed that the computer could handle about 80,000 data entries. This is about 0.2% of the total data-set. However, it represents large data-sample statistically.
* Also, during the modelling, there may be frequent restart of python notebook. Once the notebook is restarted, the sample should be loaded and available quickly for modelling purpose.
* First, the big csv file will be divided into chunks of 50,000 lines. For sampling, go through each bin / chunk and pick right amount of samples from each bin. This approach picks samples evenly from entire data-set.
* Once the sample is generated, the sample could be saved in a separate csv file. This csv file can be loaded quickly and easily whenever required.
* These steps could be written in a small python routine to generate and/or reload the samples.
* Also, the data-set presents class imbalance problem. In order to get the right balance, the negative class could be under sampled to achieve the right balance. The python routine could be extended to generate the ratio required for the modelling purpose.

The Right Model:  
The problem is of a classification to see if an ad will be clicked or not. The theory material covered that for large number of samples and features, the Random Forest Classifier seems to work the best. Also, it has been proved that for class-imbalance problem, the sample / data-set used for training purpose should be balanced.

So, for this problem, Random Forest Classifier with balanced sample should perform the best.

### Developing Various Models:

As mentioned above, from the theory, Random Forest Classifier with balanced sample should perform the best. However, the idea here is to prove this and apply the different techniques we learned during the course. The modelling will start with basic modelling and build up-on from there. The steps are given below.

* Start with simple **LogistincRegression** as a classifier and train and predict the performance after splitting the sample using **train\_test\_split method**.
* Apply **KFold** approach to train the classifier and see the improvement in the results.
* Try different **coefficients** with **KFold** approach and find the best coefficient for the classifier.
* Next is to use **GridSearchCV** to find optimized C value for **LogisticRegression** and check if it matches with KFold approach or not. **do\_classify** and **do\_optimize** functions from classification mini project.
* Once these routines are ready
  + For Samples with **50-50**, **33-66** and **20-80** positive-negative ratio
    - Find the best **LogisticRegression**, **SVM** and **RandomForest** Classifier.
* Once the best classifiers are worked out
  + Look at the **accuracy score** for test split, entire different sample sets.
  + Look at the **confusion matrix** results for each classifier with different data-set.
* Comparing accuracy scores and confusion matrix from different classifier should make clear that **RandomForest** classifier trained with **50-50 balanced** sample should perform **the best**.

### Sampling Test Data:

The Test Data ins @650 Mb in size. Similar to generating training sample, a test sample should also be generated. Generate the test sample using the following steps.

* Divide big test csv file into small chunks with each chunk containing 50,000 lines.
* While dividing the chunks, ensure that data is processed correctly. Use the routines developed for generating training sample.
* Once small files are generated, go through each small file and pick equal amount from each small file to generate a test sample.
* Save this test sample file into a separate CSV and load it quickly whenever required.

### Validating the Classifier with Test Data:

Once the test sample is generated described in steps above, use all classifiers to predict the Click-Through-Rate for test samples, generate accuracy score and confusion matrix.

The results should highlight that **RandomForest**  classifier with **50-50 balanced** sample should perform **the best.**

### Visualizing Different Data Set With Different Classifier:

#### Plotting the Data:

Visualizing the different classification on a plot will give a clear idea if a classifier is really working or not. In order to visualize the data, create X and Y data points by performing two component PCA transformation on the data set to find a 2D hyperplane. For each classifier plot X and Y 1) based on their classifier specified in training set and 2) based on predicted values from the classifiers.

#### Plotting the Reliability Curve:

As described in [this link](http://scikit-learn.org/stable/auto_examples/calibration/plot_calibration_curve.html), plot the Reliability Curve / Probability Calibration Curve for a 50-50 balanced training data set and compare the various classifiers.