POLITECNICO DI TORINO

DEPARTMENT OF CONTROL AND COMPUTER ENGINEERING

Master Degree Thesis

Click-Through Rate Prediction

Supervisor
Paolo Garza

Candidate
Klodiana Cika

Internship Tutor Reply
Giorgia Fortuna
Elena Facco

July 2021

Alla mia famiglia.

Contents

List of Tables

List of Figures

Abstract

Nowadays the internet has drastically changed the advertising industry and as technology advances and new platforms are released it continues to advance and change. The success of any advertisement campaign lies in reaching the right class of target audience and eventually convert them as potential customers in the future. In the last few years we can notice a considerable growth of the market of online advertising, which becomes very important, and provides a major source of advertising revenue.

In order to measure a campaign effectiveness there exist a variety of metrics and one of them is *Click-Through Rate (CTR)*: a ratio showing how often people who see an advertisement end up clicking it. CTR prediction means capturing user's dynamic and evolving interests from their behaviour sequence and answering to the question: How likely is the user to click on the advertisement? For performing prediction different techniques of Machine Learning are exploited.
Machine Learning is a subfield of Artificial Intelligence. While AI is the broad science of mimicking human abilities, machine learning is a specific subset of that trains a machine how to learn. The first approach to Machine Learning theory was done in the '60s for the American defence industry, in order to understand if the machine can be capable of independent reasoning.

This research presents a CTR prediction system that analyzes several factors to predict if an advertisement will receive a click or not. The analyzed dataset in the paper is the *Avazu dataset* that collects 10 days of user behaviour data in order to build and test prediction models. Firstly a detailed analysis is performed in order to better understand the user and extract the important features. Secondly a dashboard is created for the purpose of visualizing the numerous data in a more efficient way. Moreover before proceeding with building the models and the training phase, a pre-processing step is needed with the aim

of preparing the data.
This work is done in collaboration with *Machine Learning Reply* which is a pioneering company in the field of Machine Learning and Artificial Intelligence.

Acknowledgments

Throughout the internship and the writing of this paper I have received a great deal of support and assistance.

First of all I would like to thank my professor *Paolo Garza* whose feedback in each step of the work was crucial and extremely helpful.

I would also like to thank my tutors, Elena Facco for her patient support and guidance. I want to thank you for your valuable advice and for providing me the right tools that I needed to choose the right direction. *Giorgia Fortuna*, thank you for enlightening and supporting me during my stay at the company. I feel extremely blessed that I belong to the great family of *Machine Learning Reply*.

Moreover, I would like to thank Umberto Maggio who supported me during these years and who helped me a lot with the typesetting aspects of this thesis with LaTeX. Thank you, I couldn't be more grateful.

Chapter 1

Introduction

1.1 Background

Due to the evolution of technology and the increase of internet users, the market of advertising undergoes significant changes. Online marketing becomes very important and provides a key source of the revenue. Advertisers finance web publishers in order to advertise any products or services through digital approach. The revenue increases each time a user clicks on an advertisement and therefore, web publishers are focused on the rate of clicks on page views.

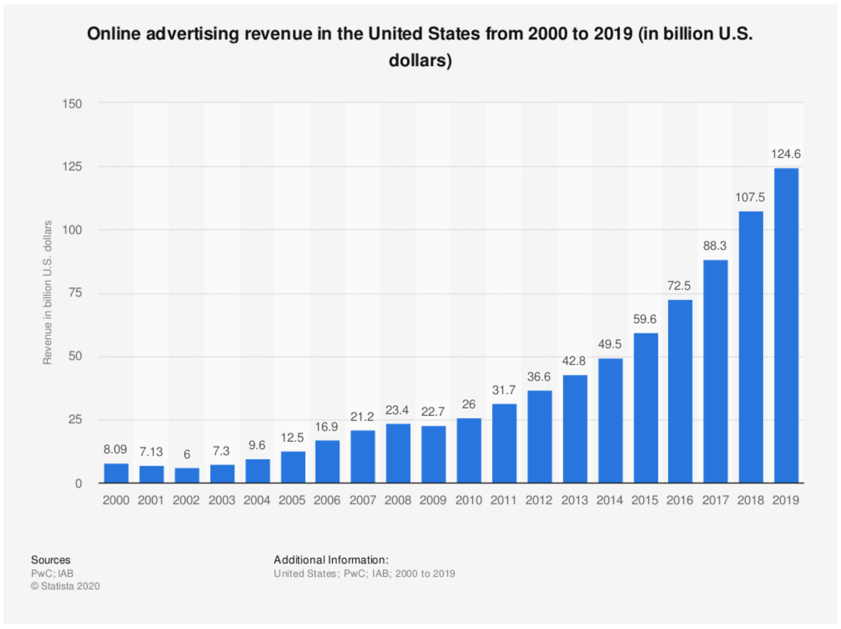


Figure 1.1. Online advertising revenue in the United States from 2000 to 2019 [2].

By analysing the data about online advertising revenue in the United States 1.1 we can observe that in 2000 we have a revenue of 8.09 billion U.S. dollars and then it grew by more than 100 billion U.S. dollars in 2019 up to 124 billion U.S. dollars. This is an indicator showing the incredible growth of online marketing. Globally the number of internet users increased from only 413 million in 2000 to over 3.4 billion in 2016. The market adapted to the changes and moved its focus on online marketing. There is a wide range of billing form such as CPM(Cost per mille), CPC(Cost per Click) or PCP(Pay per CLick), CPV(Cost per View) ect, and also different metrics are proposed for measuring the efficiency of an advertisement and its relevance with respect to a given user.

1.2 CTR

Click-through rate is the ratio showing how often people who watch an advertisement or a free product listing end up clicking it. It is obtained by calculating the number of times a click is made , divided by the number of times the advertisement is shown, expressed as a percentage.

$$CTR = \frac{NumberOfClicks}{NumberOfImpressions} \times 100\% \qquad (1.1)$$

It is a metric which measures the performance and the success of an advertisement and indicates weather the target audience is achieved. A high CTR is an indicator that the users find an advertisement helpful and relevant and it is very important because not only influences the final revenue of the whole platform, but also impacts user experience and satisfaction.

CTR prediction means answering to the question: How likely is the user to find the advertisement useful and click on it? It means capturing user's dynamic and evolving interests from their behaviour sequence. It plays a key role in modern online services.

1.3 Motivation and Goal of the thesis

The motivation to work on this project comes from the fact that nowadays online advertising is becoming more and more important and the market is progressing towards a user-centric one. Studying user behaviour and moreover predicting it has become essential for companies, not only for revenue purposes but also user satisfaction. The goal of the thesis is to analyse a large dataset, construct a model and than develop predictive techniques about user's behaviour. The aim of the event is to build Advertisement CTR prediction models which is the key problem in the area of computing advertising. Increasing the accuracy of Advertisement CTR prediction is critical to improve the effectiveness of precision marketing.

This thesis is organized in 6 chapters. The rest of the chapters are organized as follows: Chapter 2 is about Data Exploration. Chapter 3 regards Data Pre processing, Chapter 4 introduces the algorithms used in the paper. Moreover Chapter 5 and 6 describe how the performance of algorithms can be measured and the results are reported.

Chapter 2

Data Exploration and Visualization

In this chapter an accurate analysis on the provided dataset will be performed in order to understand which are the data of interest. The dataset is composed of more than 6GB of data. The analysis will start on a smaller subset of records retrieved in a random way by sampling. The training dataset provides the tag that specifies whether a user clicks an advertisement, where 1 indicates yes, and 0 indicates no. The aim is to provide the predictive click probability of each ad sample in the testing dataset.

2.1 Dataset Overview

The data set (after being masked) contains the advertising behavior data collected from 10 consecutive days and it is composed of 24 columns where we have information about site, application, advertisement and anonymous columns. The data analysis will be performed in a subset composed of 4042895(10% of the original data) rows randomly sampled.Table1.1 contains information about the columns.

id	identifier
click	target label 0/1
hour	format : YYMMDDHH
banner_pos	banner position
site_id	id of the site
site_domain	domain of the site
site_category	category of the site
app_id	application id
app_domain	application domain
app_category	application category
device_id	unique id of the device
device_ip	ip address of device
device_model	device model
device_type	device type
device_con_type	device connection type
C1	anonymized categorical variable
C14->C21	anonymized categorical variables

Table 2.1. Columns of the dataset.

The data is very clean because there are very few Nan/NULL, rows that are correctly dropped. In this case we deal with a classification problem on imbalanced data as the proportion between the two target categories are 20% of clicked advertisements(1) and 80% of not-clicked advertisements(0).The overall CTR of the dataset is 17%.
We can divide our data into groups based on how we can compare them and the types of aggregation that we can perform on the data. In the dataset we have 2 types of data: categorical and numerical . A categorical variable is a variable that can be compared only by equality. We say that 2 categorical values are or not equal. The type of aggregation we can perform are distributions, frequency, mode and we can group if there is a taxonomy over the values. Categorical features can take values from fixed ranges, for example: gender,country ect. Numerical data is a data type in numerical form. We can compare numerical data by equality and also extract information about the magnitude of the difference. In the data set we mostly have categorical data as numerical column we only have 'hour' column.

We classify the attributes in the dataset into four sets and one target variable 'click':

- Device specific attributes
- Site and app specific attributes
- Ad specific attributes
- anonymous data

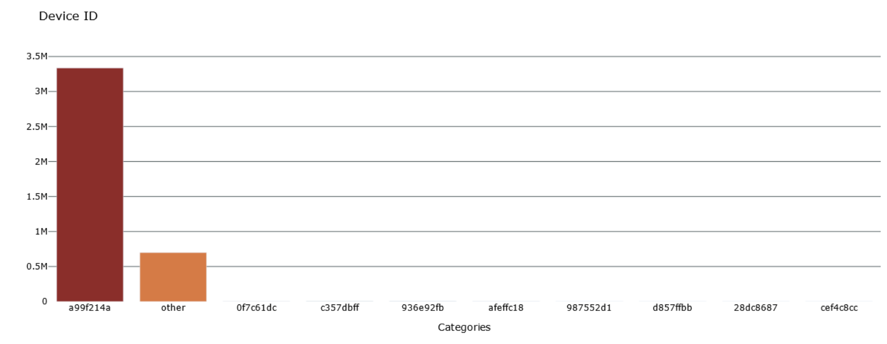
2.2 Data Analysis

Firstly, we start the exploratory analysis with the study of device specific attributes. The columns that contain information about the device are : 'device_id', 'device_ip', 'device_model', 'device_type', 'device_conn_type'.
'device_id' column contains 504285 different categories and this only in 10% of the dataset. We have a very high number of unique values and it is complex to visualize them. The same for the columns 'device_ip' and 'device_model' that respectively contain 1636808 and 6340 unique values. For these columns we visualize only 10 most representative categories and group the remaining ones under 'other'.
When it comes to 'device_ip' essentially we can deduce before visualizing it that we will have many different values and this due to the nature of an IP address, it is local and same device may have different values. We start by visualizing the distribution of the data and the click rate for each category. This visualization will help understand in a glance the important features.

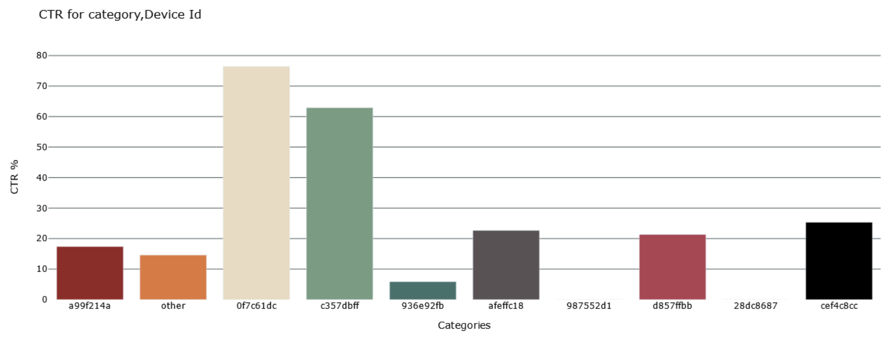
In this exploratory analysis it is represented the distribution of the data followed by the CTR rate for each category where the same color is used to represent the same category in order to make it easier to understand the data.

Device Data

In 2.1(a) device Id column is represented we can notice that one category has the most values. Following 2.1(b) CTR rate for each category. We notice that id ='0fc61dc' compares very few times but nearly every times provides a click. The same for category 'c357dbff' with a lower CTR.



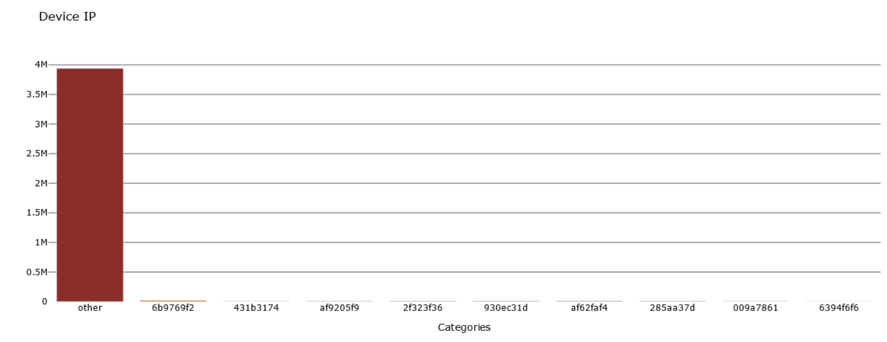
(a)



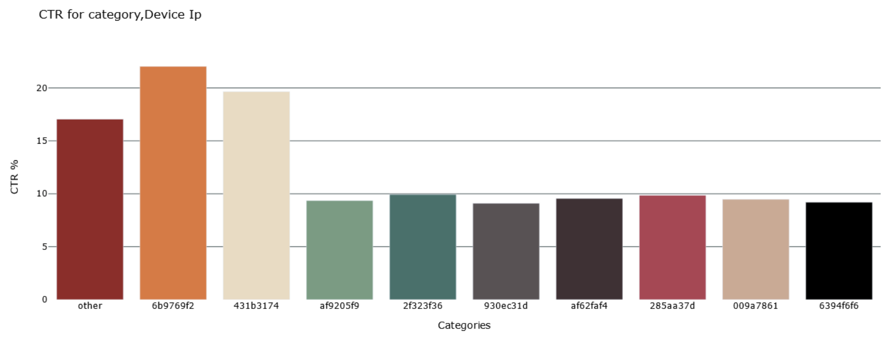
(b)

Figure 2.1. (a) Device ID data distribution, (b) CTR for Device ID.

In device IP column 2.2(a) as expected we have a wide range of values. Almost 90 percent of data is under 'other' category this means that this column presents well distributed examples. In 2.2 (b) we observe that the remaining categories that were not grouped have high CTR % although they compare few times. Observing category '6b9769f2' has the highest CTR in the visualized data meaning that the few times that compared it mostly resulted in click.



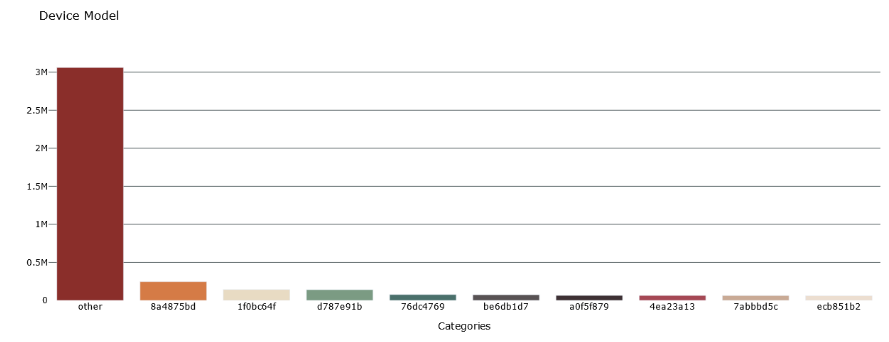
(a)



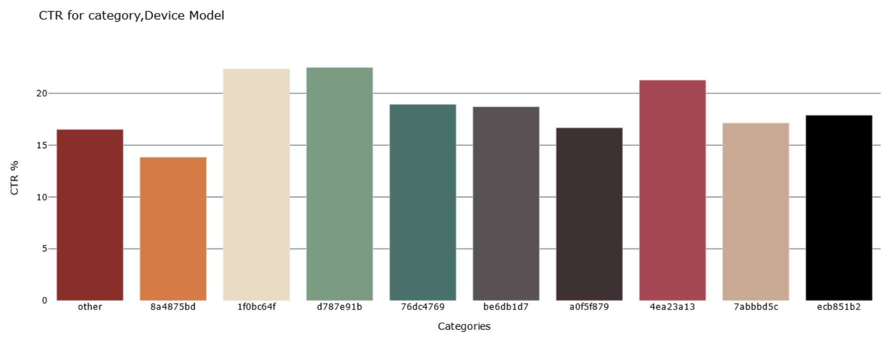
(b)

Figure 2.2. (a) Device IP data distribution, (b) CTR for Device IP.

In 2.3(a),2.3(b) we see that the column 'device_model' basically contains different categories but also here we have 'other' as principal category with a good CTR %. This also can be due to the type of data we are looking at. We have various types of device models and with the advancing of technology this market is growing very fast and in short periods of time is providing diversity and variation in the device models.



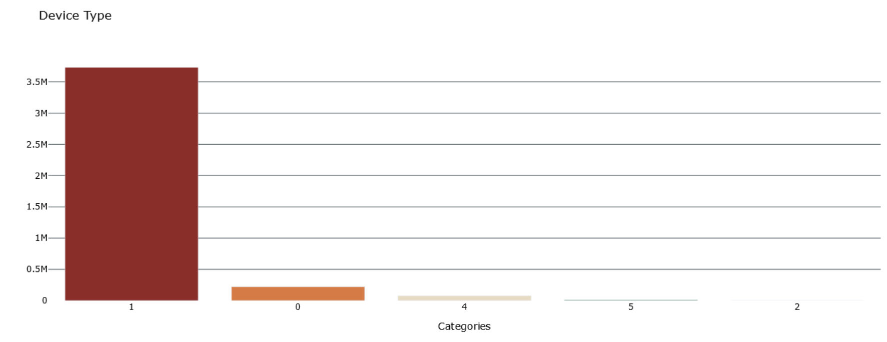
(a)



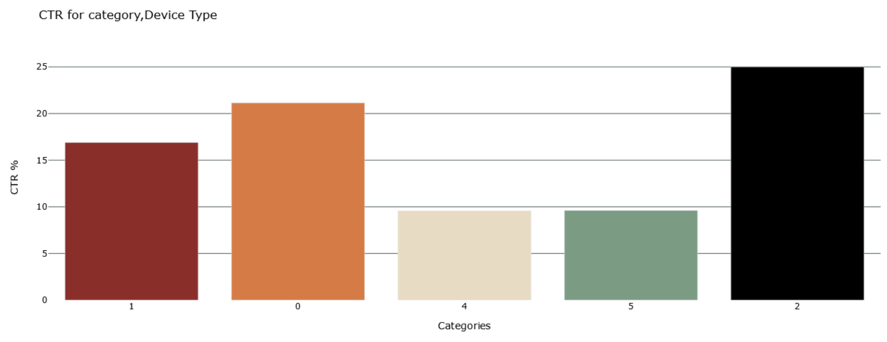
(b)

Figure 2.3. (a) Device model data distribution, (b) CTR for Device model.

Moreover, follows the visualization of the columns where no grouping under 'others' was performed. In 2.4 we can observe the 'device_type' data distribution and CTR % for each category of the column. In this case we can effectively represent all categories. We can notice that device type 2 actually is the one with the highest CTR% but if we consider the histogram we can observe that it does not compare much because we see device type 2 only 4 times in 10% of data set rows.



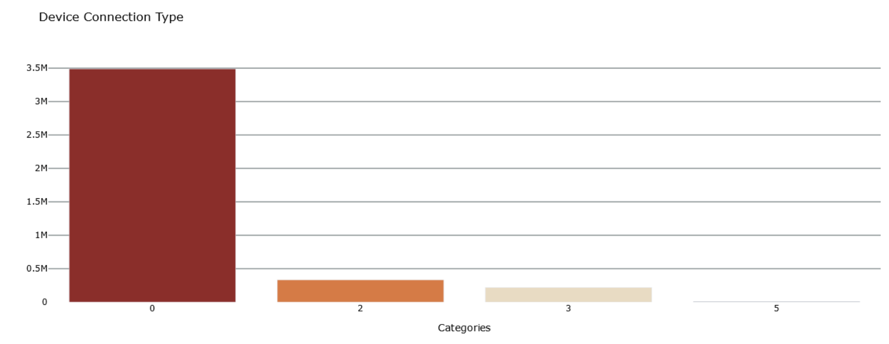
(a)



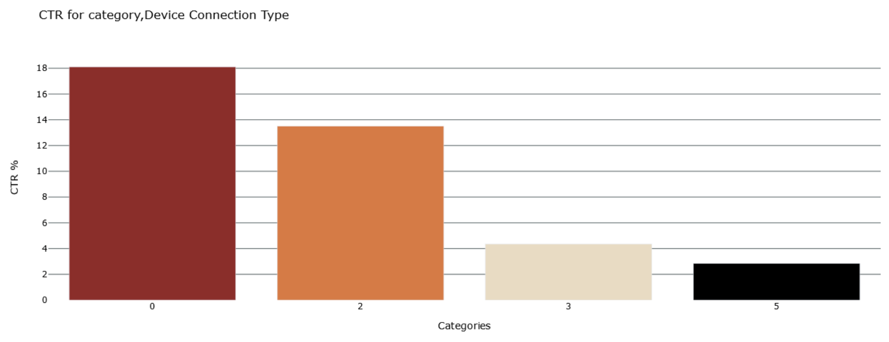
(b)

Figure 2.4. (a) Device type data distribution, (b) CTR for Device type.

Lastly, in device data is represented the connection type where it can be noticed that type '0' is the dominant one with a CTR slightly above the mean of the entire dataset. This also could be explained by the nature of a connection type. Nowdays, in the market Wifi connection type is the one most used and maybe this category is about wifi.
we can observe that type 2 has a high CTR for the number of times that compare.



(a)

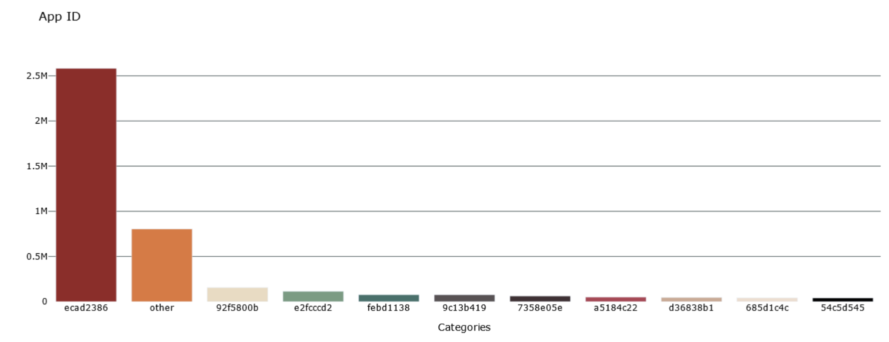


(b)

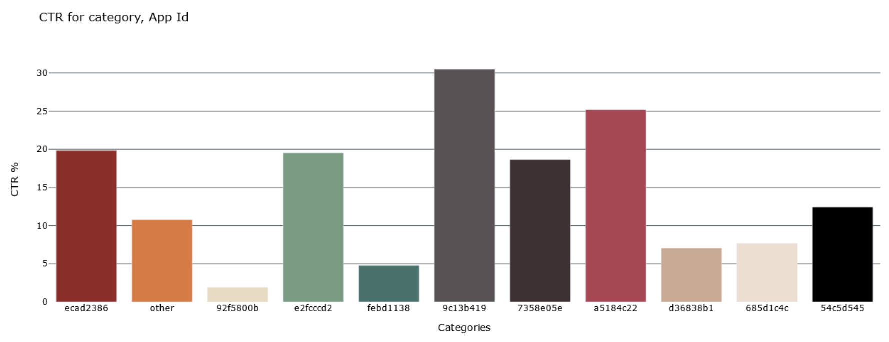
Figure 2.5. (a) Device Connection type data distribution, (b) CTR for Device Connection type.

App related data

The same analysis is made for app related columns. 'app_id','app_domain','app_category' have respectively 4894 311, 28 unique values. In a first analysis we can perceive that the data related to the app does not differentiate very well as we have a single category of 'app_id' that most compares in the data and as a consequence also domain and category. But we continue by visualizing it in order to make sure about the intuition.
For the first two columns we group under others in the same way as for device columns due to high number of categories. 2.6 (a) and 2.7(a) contains the data distribution and 2.6(b) an 2.7(b) contains the click rate for these columns.



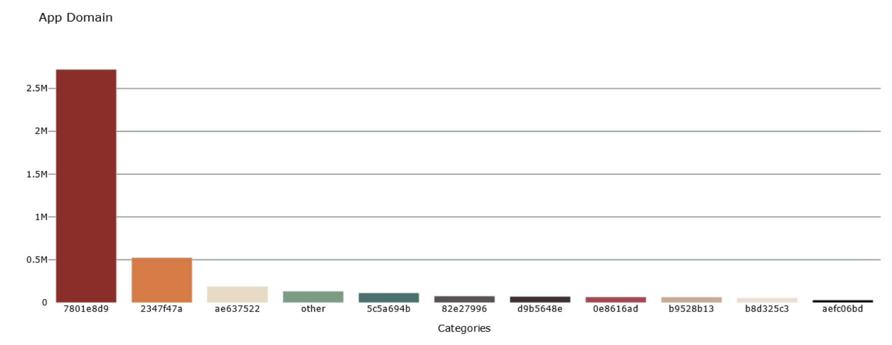
(a)



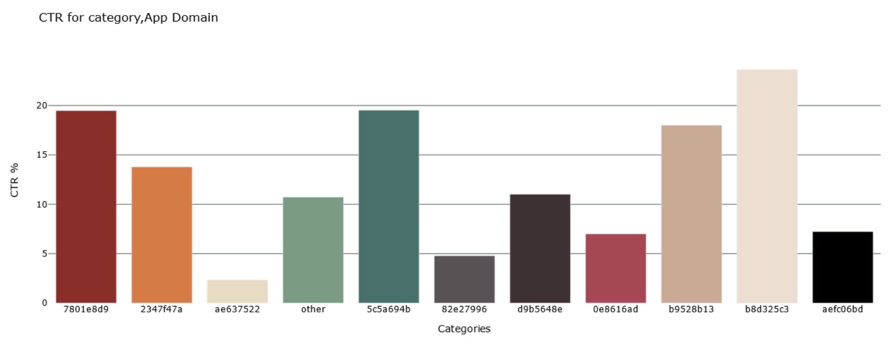
(b)

Figure 2.6. (a) App ID data distribution, (b) CTR for App ID.

we can observe that the 10% of the dataset mostly belong to a certain app with a good CTR % above the average as shown in above charts. This confirms the deduction that was previously made.



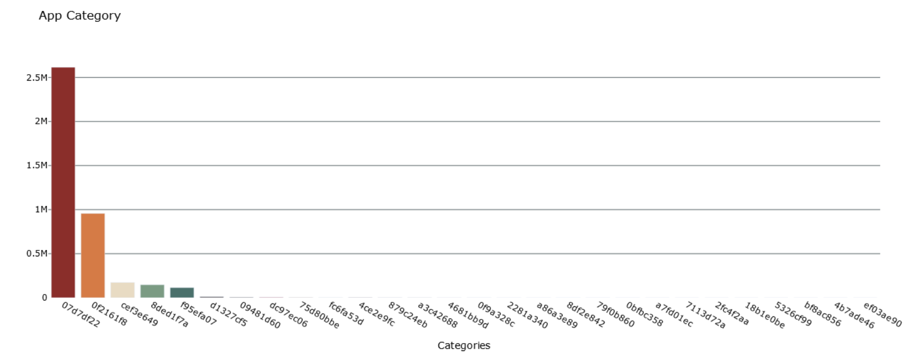
(a)



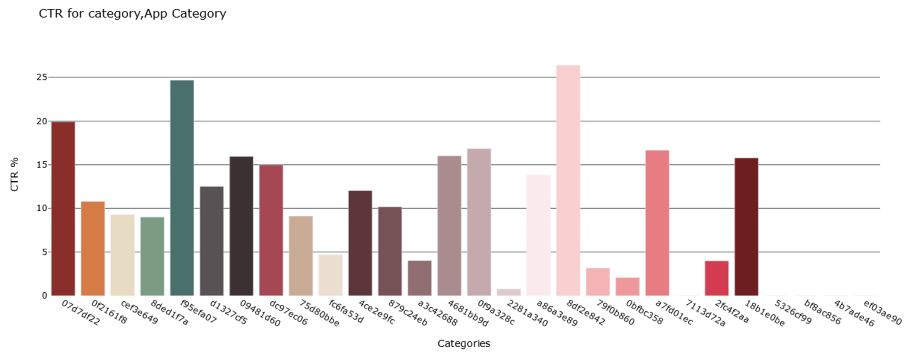
(b)

Figure 2.7. (a) App domain data distribution, (b) CTR for App domain.

Furthermore, 'app_category' column is represented in 2.8 where also in this case as expected by analysing app id one category is the dominant one. It would have been very interesting looking more into details of this column and understanding how the various applications are classified in different categories. We can add that in these data we can see the trend mentioned above , meaning some categories that compare very few times have high CTR so they could be exploited for being invested more and analysing the data afterwards.



(a)

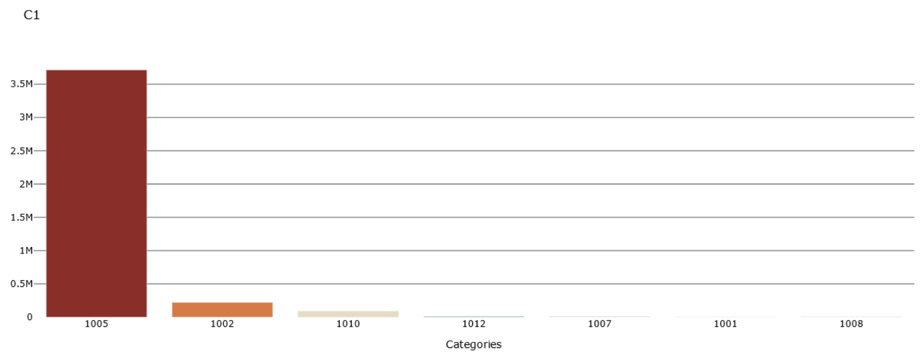


(b)

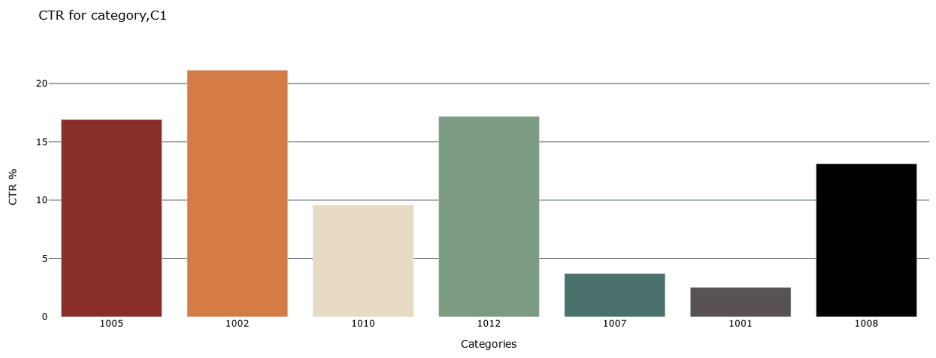
Figure 2.8. (a)App Category data distribution, (b) CTR App Category.

Anonymous data

Secondly, we continue the exploratory analysis with the anonymous data. Being that also the column names are masked it can lead us to think that these data are about the user. However we continue with columns C1, C15, C16, C18 that contain an acceptable number of categories for representation.



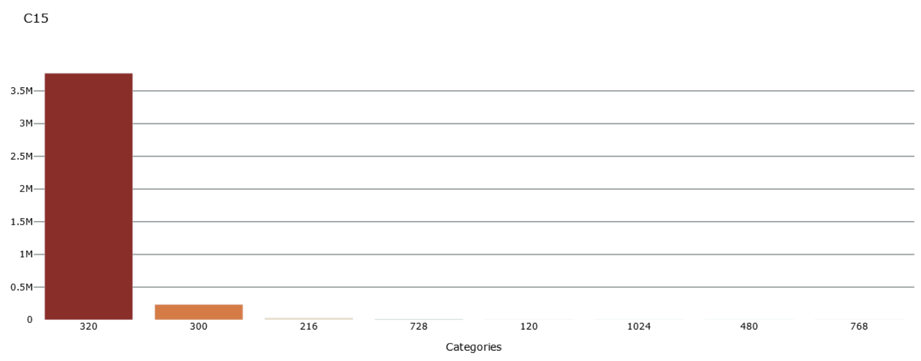
(a)



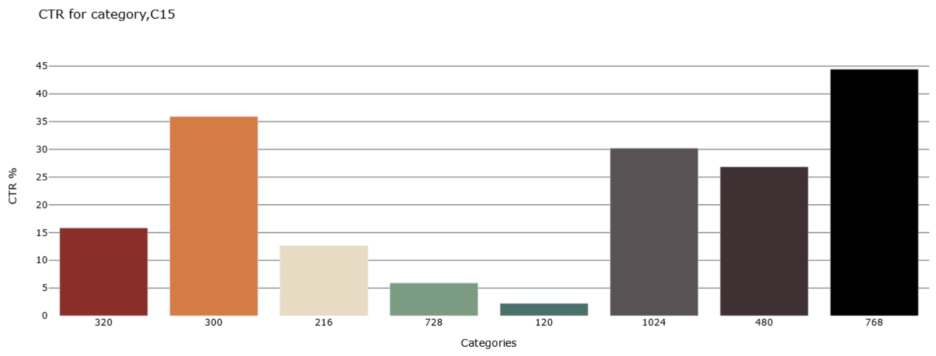
(b)

Figure 2.9. (a) C1 data distribution, (b) CTR for C1.

Column C1 has more or less the same behaviour of device_type column and this could lead us to think that maybe is some kind of advertisement related data but we cannot be sure about the intuition. We have a main category 1005 that we mostly find in the dataset with a CTR near the mean of the dataset. We can notice that category 1002 has the highest CTR% =21%.



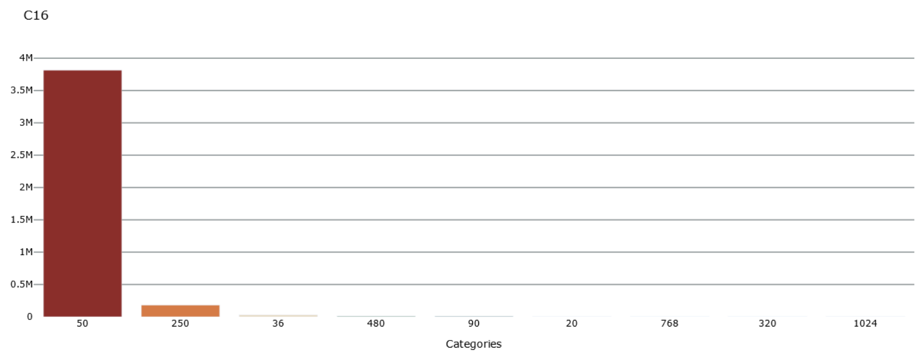
(a)



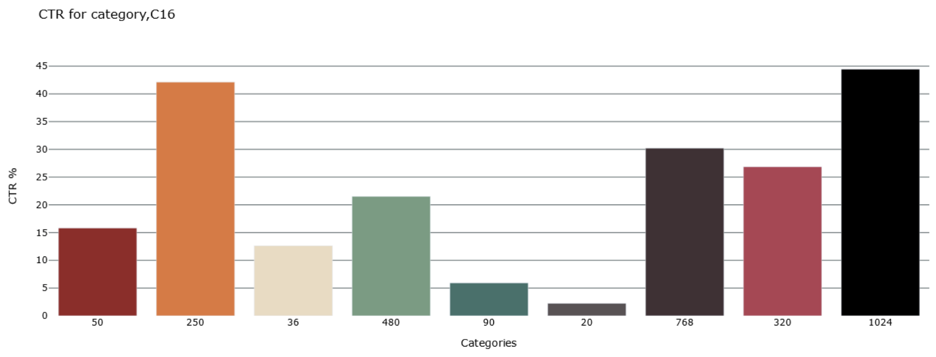
(b)

Figure 2.10. (a)C15 data distribution, (b) CTR for C15.

Also C15 column 2.10and C16 2.11 column present the same comportment as C1 column by having one category as the principal one with a discrete CTR and we notice the same trend where other categories that appear very few times have a high CTR. Maybe investing more in those categories would be a good idea in order to understand weather the trend holds up with the data. Surprisingly by funding more in these categories could result in a higher overall CTR mean.



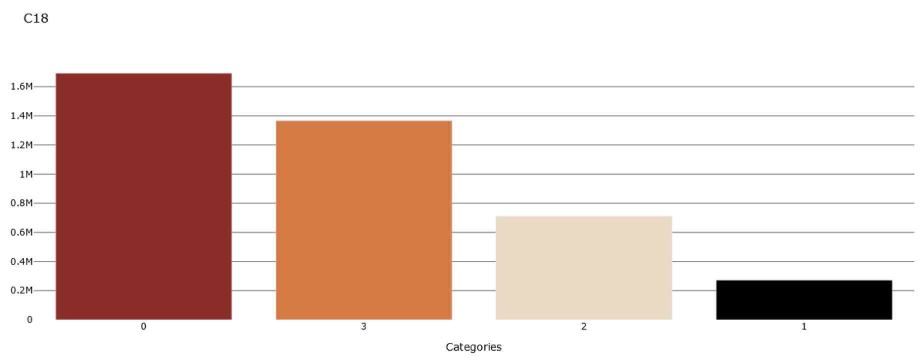
(a)



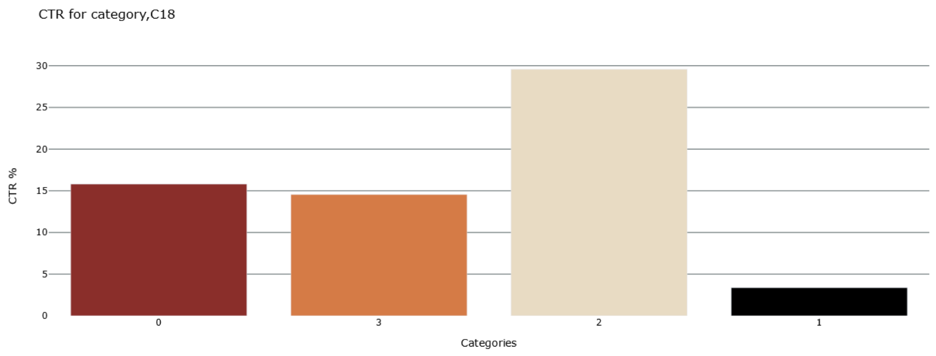
(b)

Figure 2.11. (a)C16 data distribution, (b) CTR for C16.

On the other side, column C18 presents a behaviour more well distributed. Very few categories , 2 are the mos appearing ones, 0 and 3, and the difference of CTR between these categories is very small. Moreover, we notice category 2 comparing less than 0 and 3 but having a higher CTR.



(a)

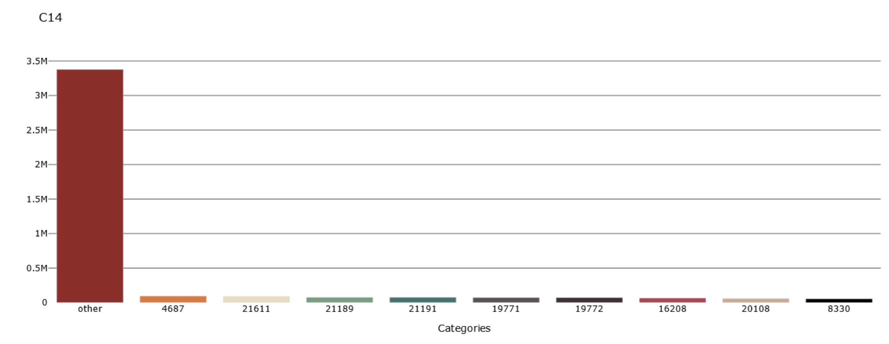


(b)

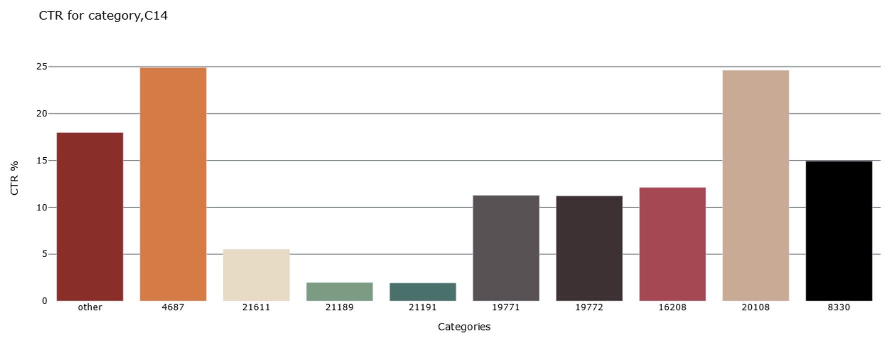
Figure 2.12. (a)C18 data distribution, (b) CTR for C18.

In these columns is difficult to make some hypothesis as there is no indication about the type of data that is being taken into consideration. It is not known weather we are dealing with an ID or with columns that may include some hierarchy or other. Moreover, it is proceeded with the analysis of the other columns where is found the need of the grouping under others strategy for properly visualizing them.

C14 2.13 has a similar behaviour as ip and model related columns where other is the category that most appears. The other categories though have a good CTR for the number of times that they appear. Maybe this data is similar to those columns being that they present more or less the same trend.



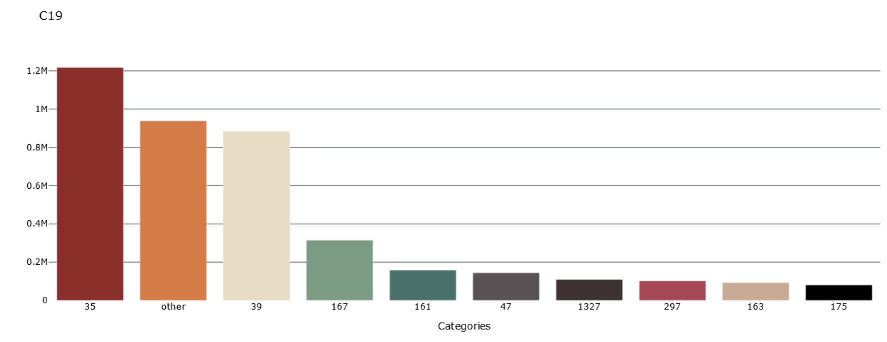
(a)



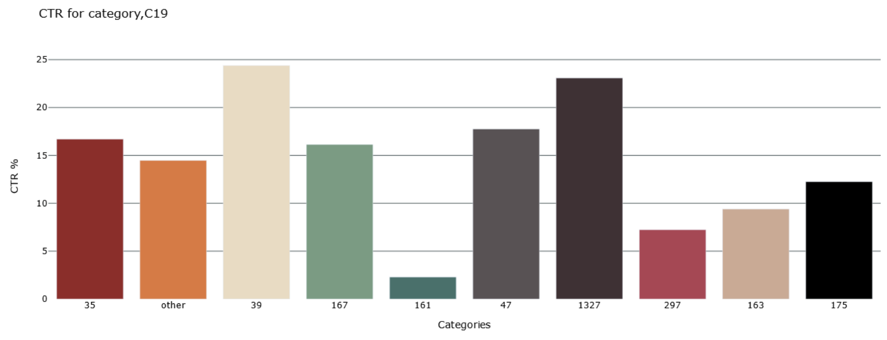
(b)

Figure 2.13. (a)C14 data distribution, (b) CTR for C14.

Moving on to C19 column 2.14 we notice a comportment as app related columns where we have one preeminent category and after that 'other' category. The CTR is presented very good and seems very well distributed. We notice category 39 that has the higher result in terms of CTR.



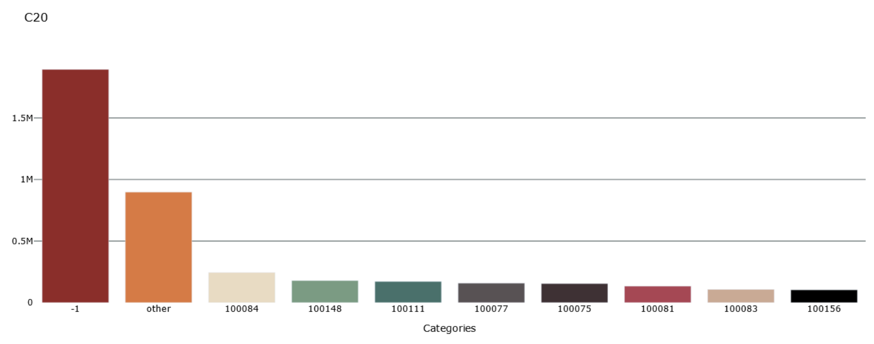
(a)



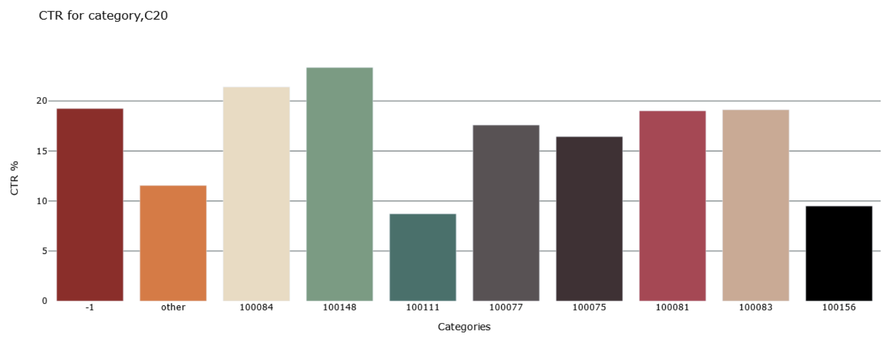
(b)

Figure 2.14. (a)C19 data distribution, (b)CTR for C19.

C20 column 2.15 is also roughly distributed as C19 by having the dominant category -1 followed by others. also here CTR is presented good and distributed among visualized categories. Maybe these two columns contain some kind of id or domain data but related to advertisement.



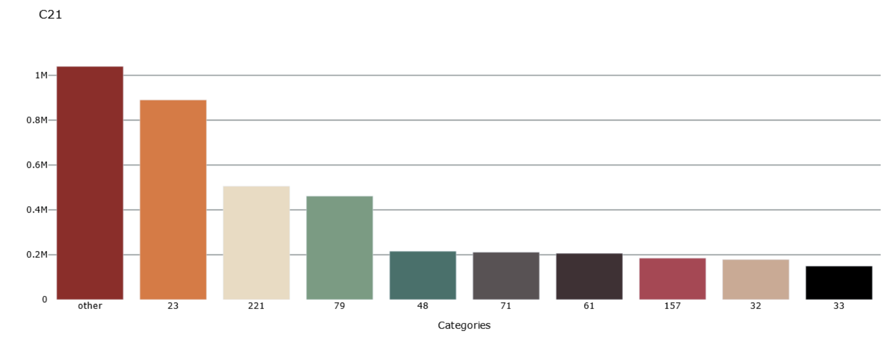
(a)



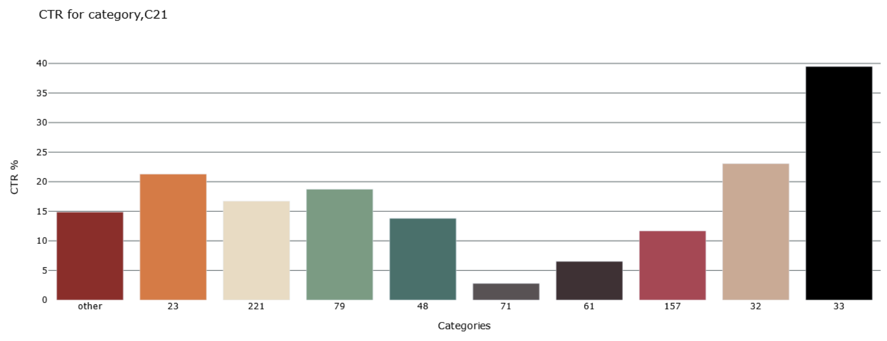
(b)

Figure 2.15. (a)C20 data distribution, (b)CTR for C20.

Lastly C21 2.16 column is presented where most appearing category is 23 with a good CTR % and we notice category 33 with the highest CTR in the column. the 'other' groups a lot of categories and these data present a CTR of 15%.



(a)



(b)

Figure 2.16. (a)C21 data distribution, (b) CTR for C21.

Site related data

Thirdly, site related data is taken into consideration. As it can be observed in figures 2.17 , 2.18 , 2.19 this data , in confront of app related data contains more categories but as always we can notice the relation between them.

Site ID column presents one site id as principal one meaning that the data was extracted mainly from one site and following is others category.

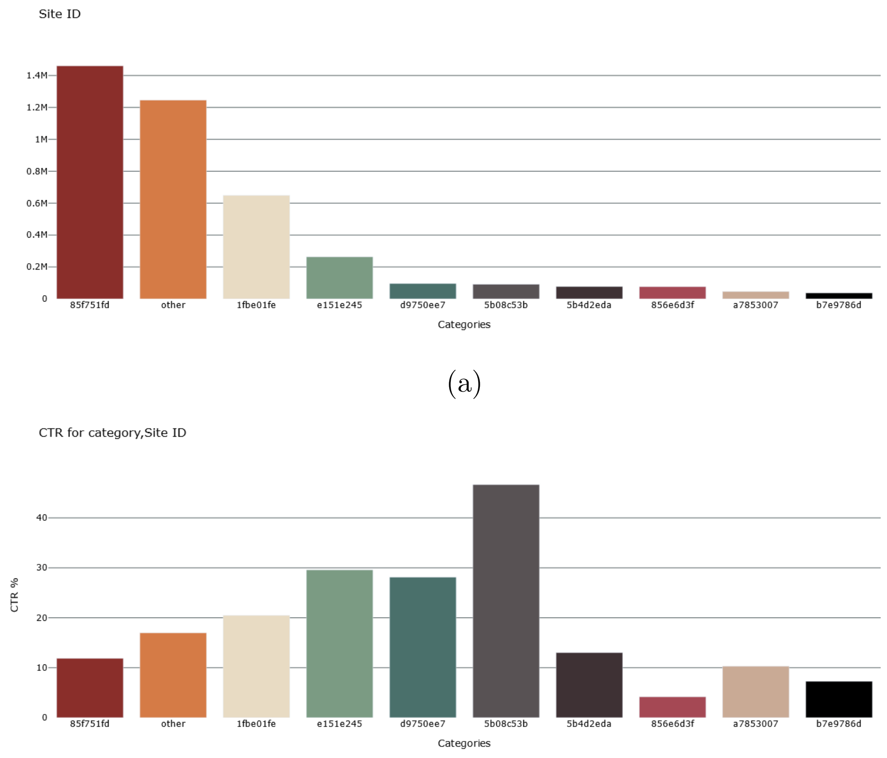
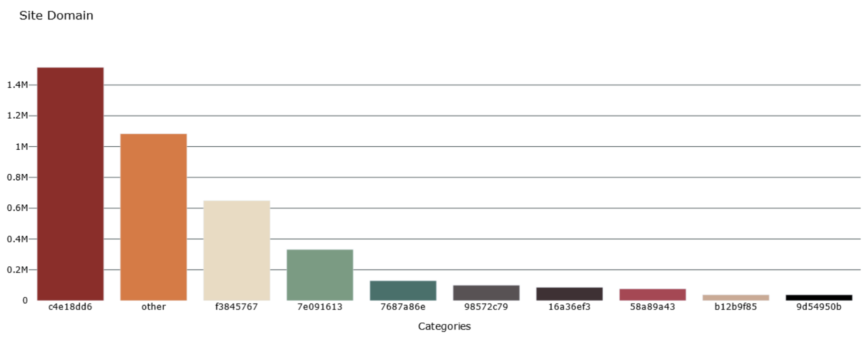
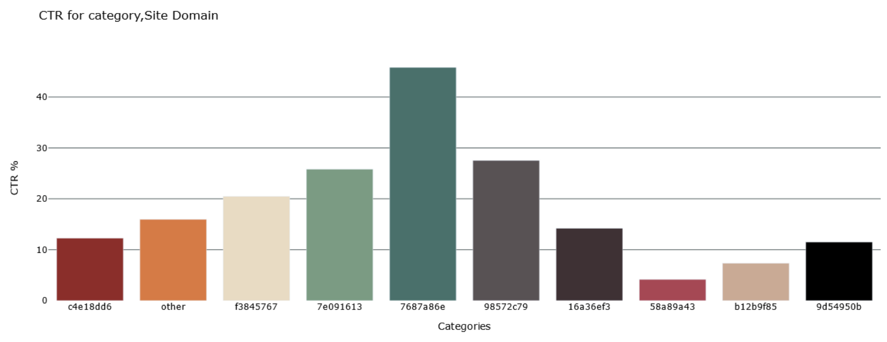


Figure 2.17. (a)Site ID data distribution, (b) CTR for site id.

Site domain due to the hierarchy nature that contains with the id column also presents one category as the dominant one. We can notice though that ’7687a86e’ category has the highest CTR in the data.



(a)



(b)

Figure 2.18. (a)Site Domain data distribution, (b)CTR for site domain.

For the site category we have a different trend with respect to app category column. Here the most representative category has a low ctr , slightly above 10% .

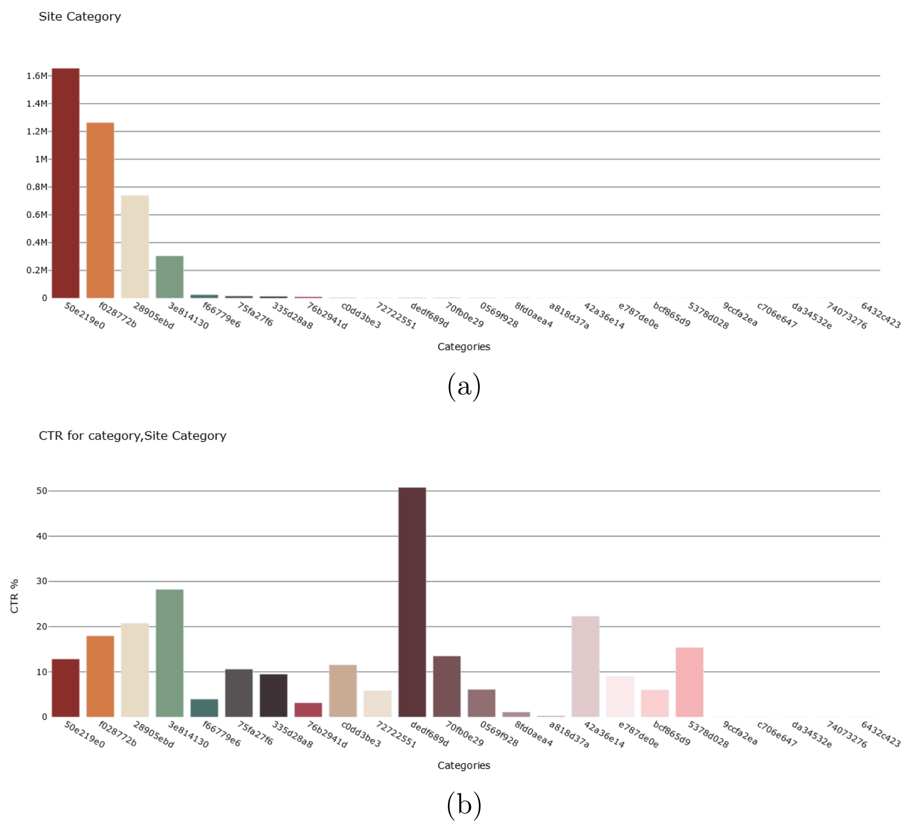


Figure 2.19. (a)Site category data distribution, (b)CTR for site category.

Temporal and Advertisement related data

Lastly, we continue with the ad specific attributes. These columns, 'hour', 'banner_pos', give us information about the position of the advertisement in the website and the hour in which the advertisement was shown and/or clicked. Before proceeding with the study we do some Feature Engineering in the column 'hour'. Essentially from column hour we extract 2 columns: 'hour_of_the_day', 'day_of_the_week'. In this way we can carry out a more complete analysis by calculating and trying to extract a trend based on the hour of the day and the day of the week.

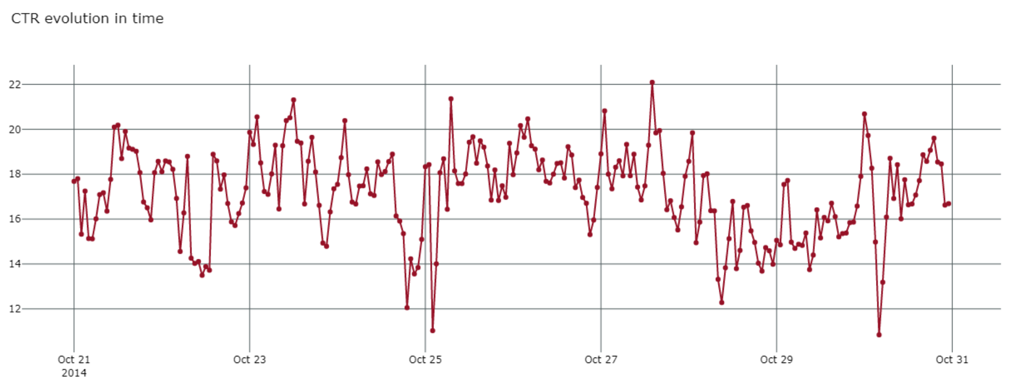
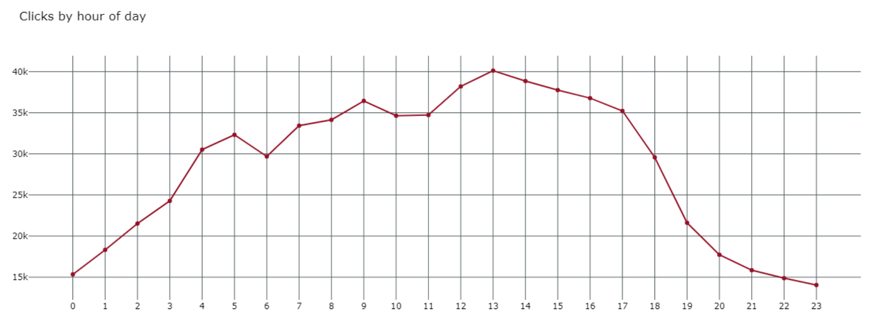
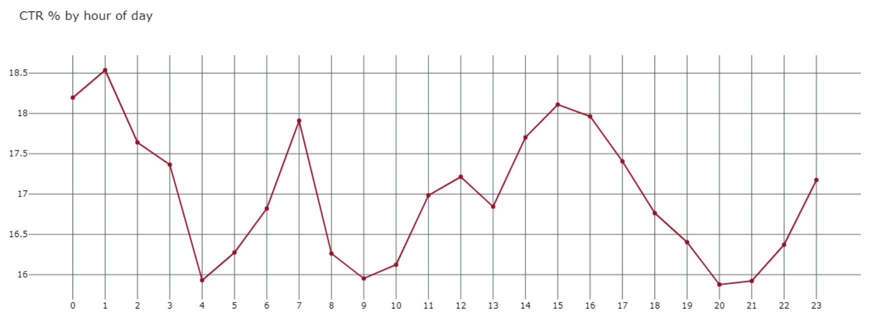


Figure 2.20. CTR% in time.

In Figure 2.2 the time evolution CTR% is shown. The maximum value of CTR is in October27 at 2pm and the minimum value is in October 30 at 4 am.We can note various peaks but Furthermore, we can notice the Click trend and the CTR% by hour of day in 2.22.



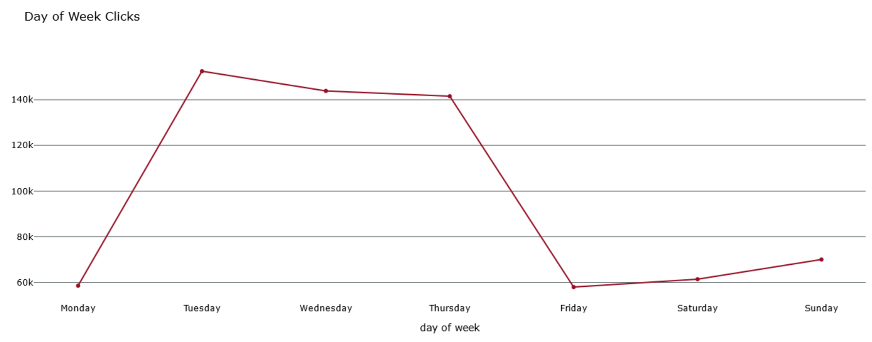
(a)



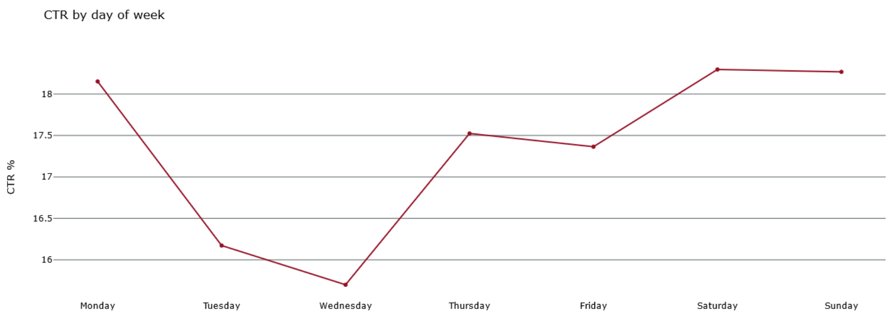
(b)

Figure 2.21. (a)Clicks by hour of day, (b) CTR% by hour of day.

We can observe by 2.21 that the behaviour is quite regular. The highest number of clicks is presented during business hours and it is associated with a CTR% near to the mean of overall data.
What can be noticed is the 1:00 am hour where we have few clicks but high CTR meaning that in that few advertisements were showed but there was effectively an interest by the users due to the fact that the CTR is high. Moreover we can notice that 20:00 and 21:00 have a very low number of clicks but also a very low rate of clicks, so maybe in this case in that hour advertisements could be showed in less frequency because there is no high interest.
Moreover we perform the analysis by day of week.



(a)

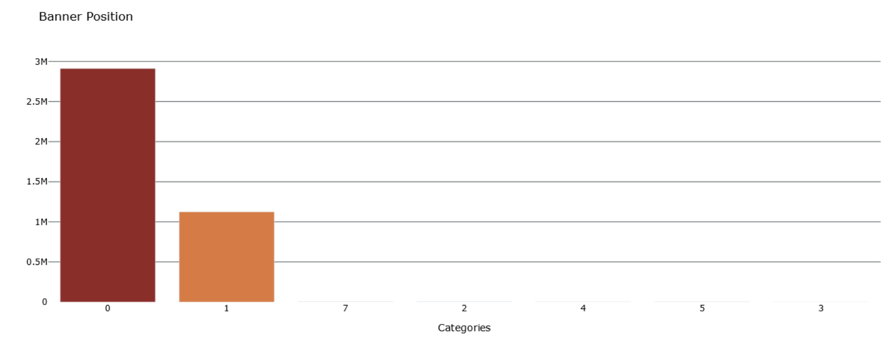


(b)

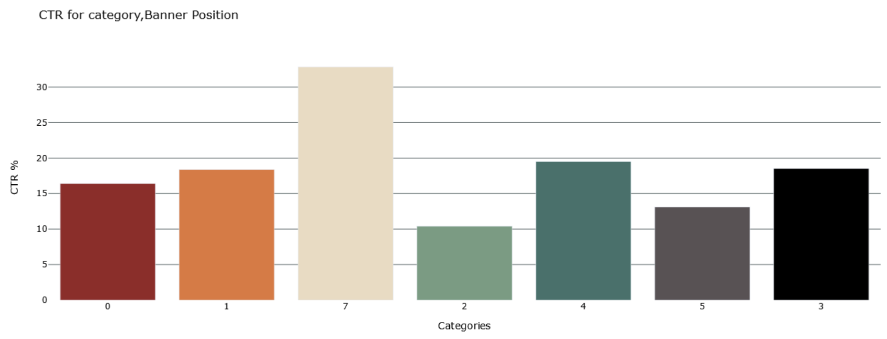
Figure 2.22. (a)Clicks by hour of day, (b) CTR% by hour of day.

By analysing the data based on the day of week we notice 2 things. Firstly, during work days we have a high number of clicks but compared to impressions result in low CTR. Different is Monday where actually we have very few clicks but also few impressions due to high CTR. Maybe this day could be exploited more in order to show more advertisements and after check the trend.
Secondly we notice that in the weekend we have very few clicks with respect to the work days but high CTR so these days could also be exploited more.

Lastly, we analyse banner position column in figure 2.23. We notice that position 0 and 1 are the important ones and they have an average % of CTR while position 7 has a high rate.



(a)



(b)

Figure 2.23. (a)Banner position data distribution, (b)CTR for banner position.

2.3 Dashboard

After the exploratory overview of the data, the need of having a tool that could assemble all information was needed. So a dashboard was created for the purpose of delivering data in the most efficient way . Information Visualization makes the data more natural for the human mind to comprehend and therefore makes it easier to identify trends, patterns within large data sets. For computational purposes the dashboard was created by using a subset of 10000 rows.The dashboard is implemented in python by using plotly dash library.

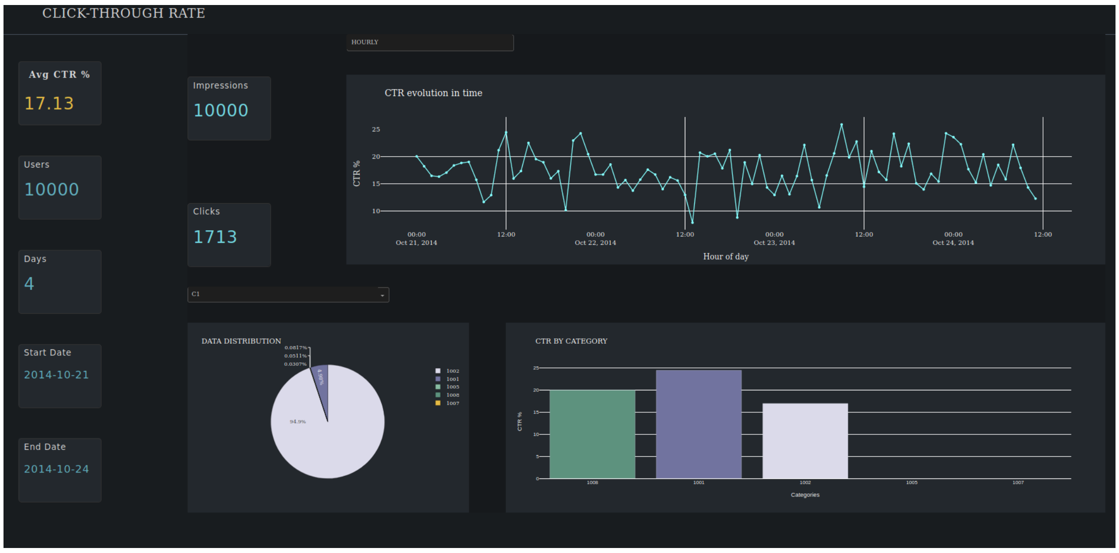


Figure 2.24. Dashboard Avazu data set.

The dashboard is divided in 3 macro-sections and it is interactive. The user can interact by the mean of dropdown menus in order to go through the entire data set.
The first macro-section contains global information about the data. The second section contains temporal visualization. By interacting with the dropdown menu we can switch such visualization. The third section contains information about each column of the dataset and the respective data distribution and the CTR% of 10 most appearing categories inside that column.The same charts represented in chapter 2.

Chapter 3

Data Pre-Processing

Before feeding the data to the Machine Learning algorithms it is extremely important to convert their format into an understandable one. This passage assures the minimization of errors and deeply affects the performance of the Machine Learning algorithms.These algorithms deal with numerical numbers so every feature of categorical type should be converted.Moreover, often datasets introduce further complexities due to the fact that data is not clean or processed. There is a wide variety of Data pre-processing techniques, and based of the type of algorithms used for making predictions many of them can be exploited. In this study the focus is in the Supervised Learning, Classification problem.[14]

In this paper we perform the below pre-processing techniques:

- Split the data set
- Handling categorical variables
- Scaling numerical features
- Dimensionality reduction

All these steps were performed by the mean of a pipeline which takes in input the raw data, it scales,transforms and splits it and gives in output data that are ready to be fed to the models.
Pipelines are very powerful and widely used in the data science field. We can see it as a sequence of blocks where the output of a block is the input of another block.

Pipelines are a scikit-learn module that allows you to automate of the workflow, creating extremely sophisticated algorithms from the combination of basic library objects.[18] In this chapter a detailed description of each step is provided.

3.1 Splitting the data set

Before proceeding with the pre-processing we split the dataset into 2 parts. Essentially we use a part of dataset(training set) in order to fit the model and train it, and we use the other part(testing set) as a test for measuring the performance of the algorithm. The splitting of the dataset is done by using the function of sklearn *train_ test_ split* that return X_train, X_test and y_train and y_test. [19]

The dataset in our case was divided in:

1. Training set (90%)
2. Testing set (10%)

Both parts of the dataset will go through pre-processing but in a different way. We perform pre-processing after splitting for the aim of maintaining the total division between the training phase and the testing phase. In order to achieve an accurate result we must treat the test set as unseen data. y_train and y_test contain 'click' column data for both parts of the subsets. They will be used in the following way:

1. y_train is used in the fit method for training
2. y_test is confronted with y_predicted(the result we obtain after predicting from test set)

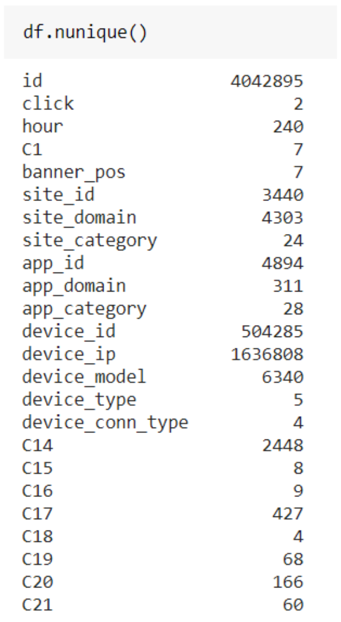
3.2 Handling categorical Variables

Categorical variables need some pre-processing as the format that they adapt is not suitable for being fed to the algorithms. There are many ways of how these variables can be managed and there exist many type of encoding methods but there is the need to be careful when using with them. If ,for example, we encode a feature with *OrdinalEncoder*[17] we are supposing and adding an order inside the column. The encoder basically associates a number to each category but this kind of conversion is not suitable in our case.[5]

The categorical features of *Avazu* data set ,as described in 2, compose the biggest part of the data and that is why the pre-processing step is crucial. Exploring the data it was noticed that columns like 'site_id','app_id','app_domain' ect. were very difficult to visualize as they contain an elevate number of categories (hundreds, thousands).Even if we correctly encode these data , for example with *OneHotEncoder* [11], we still would not have a good result. OneHotEncoder creates a new column for each category of the feature and this will lead in extremely high number of columns ,high dimensionality.
The first step is to reduce the number of categorical values inside a column and after, perform a transformation for converting into right format. Different techniques were studied in order to deal with this problem.

3.2.1 Value Counts

"value_counts()" is a function of pandas library in python. It takes in input a column and gives in output all the different categories of that column and the corresponding number of times that the category appeared in the data. The main idea for reducing the number of categorical values is to take only 10 most representative.
As shown in Figure 3.1 some columns have even more that 10000 categorical values. The columns that will be processed by this transformation are: site_id', site_domain', app_id' ,app_domain', 'device_id', device_ip', device_model', C_14', C_17', C_19', C_20', C_21'.



```
df.nunique()
```

```
id                  4042895
click                     2
hour                    240
C1                        7
banner_pos                7
site_id                3440
site_domain            4303
site_category            24
app_id                 4894
app_domain              311
app_category             28
device_id            504285
device_ip           1636808
device_model           6340
device_type               5
device_conn_type          4
C14                    2448
C15                       8
C16                       9
C17                     427
C18                       4
C19                      68
C20                     166
C21                      60
```

Figure 3.1. df.nunique().

After carefully studying the value counts a mapping was built {column_name : number_of_categorical_values_to keep} for the purpose of associating each feature with the number of unique categories that we want to keep untouched,for example {site_id : 10}. The function returned in output the transformed dataset where all the categorical values inside the column are now truncated into 10 most representatives + all the others grouped under 'others' value. (So site_id after the function has only 11 different values).

3.2.2 Custom Transformer

Another technique was exploited,a custom one that was build on top of the data that is being analysed.After careful considerations, this one was chosen in order to encode the dataset. The goal of the technique is: Instead of concentrating in the categorical values let's concentrate on the target column and the relationship of each categorical value with the CTR.

The main idea is that inside my column I will have, not my categories but a quantity that represents my categories. What we want to have in output is data that is good for training an algorithm that will later predict, but also we would like to have columns that do not contain an elevate number of categorical values. The principle is to take the overall CTR mean and categorize it into 5 levels:

- mean_CTR + 0.04 **very high**
- mean_CTR + 0.02 **high**
- **mid**
- mean_CTR - 0.02 **low**
- mean_CTR - 0.04 **very low**
-

The thresholds were chosen in an experimental way. The percentiles could have been chosen but it was observed that the results were good with these thresholds. This classification already introduces a kind of order to the categorical values and this means very close to numerical value. The idea is that instead of labeling very high,high,mid,low,very low I can immediately set 5, 4, 3, 2, 1 and transform my categorical variables into numerical ones. The columns that received this transformation are the ones with a very high number of categorical values: site_id', site_domain' ,app_id', app_domain', device_id', device_ip', device_model', C_14', C_17', C_19', C_20', C_21'.

The essence of the technique is:

- Take a column
- Take a categorical value inside that column
- Calculate CTR for that variable
- Classify the value

In output I will have for each categorical column at most 5 categories and a mapping {categorical value : CTR level} that I will consecutively use for prediction phase.
There is the necessity to distinguish between 2 different phases. In the training phase the aim is to create a dictionary that contains the mapping {categorical value : CTR level}, in the testing phase the aim is to check the dictionary for finding the current categorical value and replace it with the CTR level. It is very important to make this distinction because in order to calculate the CTR levels we need 'click' column, but as explained in 3.1 the test set is used as unseen data, without a 'click' column. If by mistake we would include that column the performance would result very high but it is being predicted

what already is known. In practice, analysing column 'site_id' for '543a539e' category the steps are the following ones:

- Take '543a539e' category inside 'site_id' column
- Calculate the CTR for '543a539e' (clicks/impressions) :CTR= 0.69
- The overall mean value of CTR in the dataset is 0.17
- CTR('543a539e') $> 0.17 + 0.04 =>$ This category is classified as very high
- Replace all '543a539e' values with 5

This process will go in loop for all the categories inside the 'site_id' column until it contains only values 1, 2, 3, 4, 5 that represent the magnitude of CTR.
This operation will be done in all defined columns with high number of unique categories.

3.3 Categorical variable Encoding

After the Custom Transformer, we still have the other categorical features with an acceptable number of values: 'C1, 'banner_pos, 'day_of_week', 'device_conn_type', 'C15', 'C18', 'C16', 'device_type'. We do not process these features with the Custom Transformer but we continue with *One-Hot-Encoding*, one of mostly used method in data analysis.

3.3.1 One-Hot-Encoding

One-Hot-Encoding[11] is a technique that basically creates what are called "dummy variables". Suppose we have a column named color that contains the values : 'red' 'green' 'blu' 3.1. After One Hot Encoding this column is split in 3 columns as showned in 3.2.

color
red1
green
blu

Table 3.1. Color Column.

color_red	color_green	color_blu
1	0	0
0	1	0
0	0	1

Table 3.2. One-Hot-Encoding.

One Hot Encoding is very simple and it gives great results but we need to take into account the increase in the dimensionality of the dataset.

3.4 Scaling numerical features

The only numerical column is 'hour_of_the_day' and the motif of scaling it is that we do not want a machine learning algorithm to consider values as lower or higher , regardless of the unit. We need to introduce normalization when handling numerical columns.

Different scalers such as *Standard Scaler*, *MinMax Scaler* , *Robust Scaler* can be exploited. In this paper *MinMax Scaler*[10] is used which scales and translates each feature individually such that it is in the given range, between zero and one.

3.5 Dimensionality Reduction

Another study that we need to conduct is the one of *Dimensionality Reduction*.In both Statistics and Machine Learning, the number of features or columns is referred to as *Dimensionality* of data. Dimensionality reduction addresses the task of reducing the number of features in a dataset while keeping as much of the variation in the original dataset as possible. When we have a problem of classification as in our case click - no click decreasing the number of feature is very important. From the computational point of view it is faster to learn a classifier in low-dimension input. More importantly, has a regularizing effect that can help avoid over-fitting. The reason is that DR can remove two types of “noise” from the input: [1]

- independent random noise
- Unwanted degrees of freedom, which are possibly nonlinear

Some of the advantages of Dimensionality Reduction are:

- Increase in the overall performance of machine learning algorithms
- Avoids over-fitting
- Multicollinearity
- Removes noise in the data
- ect.

There is an extensive selection of dimensionality reduction methods. In this paper we study the Correlation.

3.5.1 Correlation

Correlation is a statistical quantity in order to measure the linear relationship between 2 variables.It’s a common tool for describing simple relationships without making a statement about cause and effect.It is useful because it can indicate a predictive relationship that could be exploited in the practice. Variance also could be used for determining the relation so why correlation?
The main idea is that Variance depends on the scale of the data,in this case it is not very suitable because maybe the same "degree" of relationship seams very high due to the

fact that data have high value. We would like to know the "degree" of the relationship independently of the scale so that is why we use Correlation.

In this study the Correlation is used in order to decrease the dimensionality of the data set. Fundamentally if 2 columns, A and B, are highly correlated this means that these 2 columns present the same information. This means that from column A I can predict column B . In this way we have redundant information inside the data set so we can drop one of them.

Pearson Correlation Coefficient

It is one of the most used in data analysis. It measures the linear relationship between 2 variables.

$$P_{xy} = \frac{\text{cov}(X,Y)}{\sigma_x \sigma_y} \tag{3.1}$$

This coefficient reflects the association and amount of variability between 2 variables and is and it is susceptible to the outliers. In this paper after the data pre-processing before feeding the data to algorithm we study the Pearson Correlation coefficient. The Matrix is showned in 3.2 .

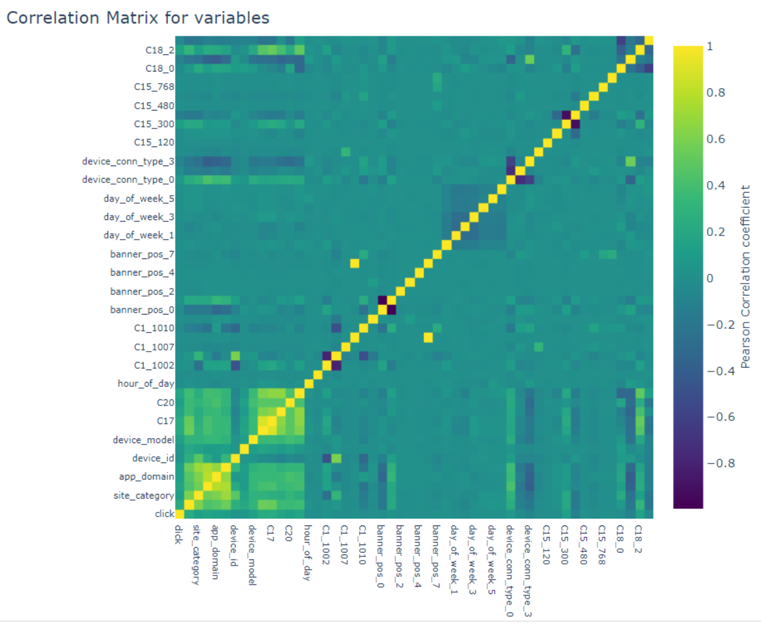


Figure 3.2. Pearson Correlation Coefficient.

Pearson Correlation Coefficient Matrix is a symmetrical matrix where we have the columns on both axis and M(i,j) = $P_{col1/col2}$. We can notice that in the diagonal we have only yellow color, this derives by the fact that we calculate P on the same column and as expected they have P=1.
Outside the diagonal if we encounter P>=threshold than one of the 2 columns must be dropped. This step is performed by a function that gives in output the columns to be dropped. The threshold chosen is 0.8. If two columns of the dataset have P>=0.8 this means that they are highly correlated and I am going to drop one of them as keeping both means adding a new dimension that essentially it is not giving any new information so it is useless.

Chapter 4

Machine Learning Algorithms

In this chapter a more complete examination of the used models is delivered. We have the baseline algorithms and the more complex one. In this paper we discuss a classification problem that means predicting a class label for a given input data: Given an example, classify if it will receive a click or not. Classification is one of the categories in supervised learning where algorithms learn from labeled data. After understanding the data,so after the training phase the algorithm will predict a label in the new unseen examples. For building the models the scikit-learn library of python was used. We start this chapter by introducing a historical point of view of Machine Learning and later on presenting the algorithms used in this project.

4.1 History

Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed for it.[9]. Machine Learning algorithms are built to make inference. The process of inference is done in 3 steps.[4]

- Observe a phenomenon
- Construct a model of the phenomenon
- Do predictions

Machine Leaning models follow the principle of *Pattern recognition*, extracting pattern and regularities from the data. The aim is:

Given pairs of examples (vector representation of an observation, class label),class labels are $\mathcal{Y} = \{-1, +1\}$, the algorithm will construct an association *vector representation of an observation* $\rightarrow$ *class label* by minimizing the error on unseen examples.
Once we have the vector representation of observations subsequently we need to find a function that best fits the data and separates the positive class from negative class. We can find this function by the means of *Interpolation* but it is not reasonable as it increases the complexity and it is not feasible. The idea is to search for regularities in the observations and generalize. Formally:

- We consider an input space $\mathcal{X} \subseteq \mathbb{R}^d$ and an output space $\mathcal{Y}$
- **Assumption:** examples $(\mathbf{x}, y) \in \mathcal{X}$ are *identically* and *independently* distributed with respect to an unknown probability distribution $\mathcal{D}$
- **Samples** We observe a sequence of m pairs $(\mathbf{x}_i, y_i)$ generated *identically* and *independently* by $\mathcal{D}$
- **Aim:** Construct a prediction function $f : \mathcal{X} \rightarrow \mathcal{Y}$ which predicts an output y for a given new $\mathbf{x}$ with minimum probability of error

Discriminant models directly find a classification function $f : \mathcal{X} \rightarrow \mathcal{Y}$ from a given class of function $\mathcal{F}$. The function should be the one having the lowest probability of error

$$\mathcal{L}(f) = \int_{\mathcal{X}\times\mathcal{Y}} l(f(x), y)\, d\mathcal{D}(x, y) \tag{4.1}$$

where l is the instantaneous loss defined as:

$$l : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^{+} \tag{4.2}$$

In classification the function is usually the misclassification error:

$$\forall(x, y); l(f(x), y) = \not\Vdash_{f(x)\neq y}$$

As the probability distribution $\mathcal{Y}$ is unknown the analytic form of the true risk cannot be driven so the prediction function cannot be found. Instead it is used the *Empirical Risk Minimization principal.*[20] ERM states:

Find f by minimizing the unbiased estimator of its generalization error $\mathcal{L}(\mathrm{f})$ on a given training set $S = (x_i, y_i)_{i=1}^m$

$$\mathcal{L}(f, \mathbf{S}) = \frac{1}{m} \sum_{i=1}^{m} l(f(x_i), y_i) \tag{4.3}$$

The first attempt to build learnable artificial model it begun in 19th century with the study and the representation of the biological neuron. In 1943 we have the first representation of a formal neuron by Mccullah & Pits shown 4.1

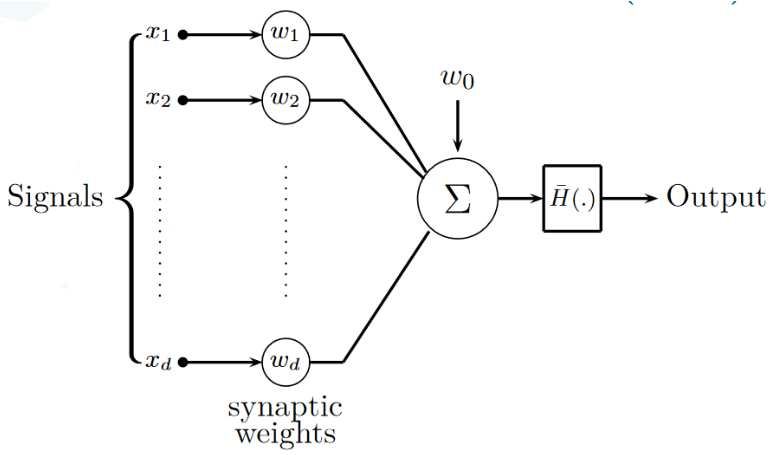


Figure 4.1. Formal Neuron.

In 1958 we have the first machine that was able to do predictions. The experiment of Rosenblatt, *Perceptron* was the first one that was able to predict by taking in input a photo weather it was a woman or man.
Perceptron has a linear prediction function:

$$h_w : \mathbb{R}^d \to \mathbb{R}\mathbf{x} \to \langle \bar{w}, \mathbf{x} \rangle + w_0 \tag{4.4}$$

The aim is to find the weights **w** in order to minimise the distance between the misclassified example and the decision boundary.

Perceptron parameters:

- **Objective function:**

$$\mathcal{L}(\mathbf{w}) = -\sum_i y_i(\langle \mathbf{w}, x_i \rangle + w_0) \tag{4.5}$$

- **Update rule: Gradient Descent**

$$\forall t \geqslant 1, \mathbf{w}^t \leftarrow w^{(t-1)} - \eta \nabla_{\mathbf{w}^{(t-1)}} \mathcal{L}(\mathbf{w}^{(t-1)}) \tag{4.6}$$

4.2 Baseline algorithms

A baseline is a simple model that provides reasonable results on a task and does not require much expertise and time to build. Because it is simple, a baseline is not perfect but help us put a more complex model into context in terms of accuracy. We analyse *Random Forest*, *Logistic Regression* and *SGD Classifier*.

4.2.1 Random Forest

Random Forest is an ensemble Machine Learning algorithm that it is made of decision trees and combines their output [12]. It uses *Bagging* and a slightly different splitting rule from decision trees. we first introduce the concept of *Bagging*:

Bagging

Assume you have a dataset $\mathcal{D}$ that contains m features. From this dataset you construct m datasets $\mathcal{D}_1, \mathcal{D}_2, ...\mathcal{D}_m$ of the same size of $\mathcal{D}$ with replacement. For each dataset you run a classifier (decision tree) $h_j(x)$. The final classifier will be the mean of all weak learners

$$h(\bar{x}) = \frac{1}{m}\sum_{i=1}^{m} h_i(x) \tag{4.7}$$

The aim of bagging is to reduce the **Variance** of the classifier. An algorithm error is seen as the sum of two terms; *bias* and *variance.* In order to decrease the error we want to reduce the variance.

Before diving into Random Forest an overview of a decision tree is needed.

Decision Trees

Decision trees start with a basic question,from there, you can ask a series of questions to determine an answer. The core is that we build a tree that will divide the space in 2 parts with similar labels. Than we use a threshold t in order to check if a feature value $\geqslant$ t ,than it will go in one child node otherwise into the other one. Ideally all positive points fall into one child node and all negative points in the other. The algorithm will

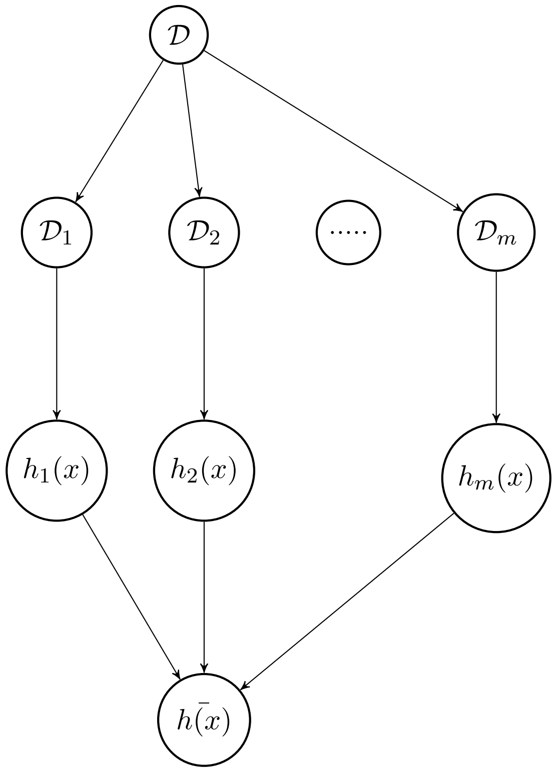


Figure 4.2. Bagging.

continue until all leaves are pure so we have the same label If this is the case, the tree is done. If not, the leaf nodes are again split until eventually all leaves are pure (i.e. all its data points contain the same label or cannot be split any further (in the rare case with two identical points of different labels). For the purpose of splitting there are 3 widely used methods: entropy, gini, and twoing.[6]

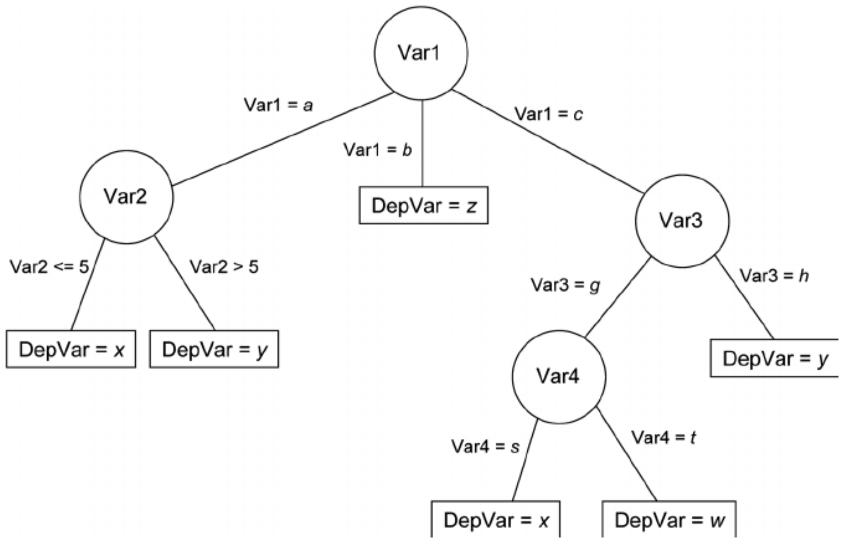


Figure 4.3. Decision Tree.

Gini index

Gini Index / Gini impurity, is a quantity that determines and compute the probability that a feature selected in a random way is misclassified. Gini index takes values in the range from 0 to 1, where 0 means that there is no impurity or only one class exists and all observations belongs to that class. 1 specifies that there is some random distribution of observations across distinct classes. The value of 0.5 of the Gini Index shows an equal distribution of elements over some classes.
Gini Index can be expressed as:

$$GINI\_index = 1 - \sum_{i=1}^{N}(p_i)^2 \tag{4.8}$$

Where Pi denotes the probability of an element being classified for a distinct class.

Random Forest

Random Forest is very powerful due to the fact that it is a bagged decision tree algorithm. The difference of Random Forest with respect to decision trees is that every time a split is done it will be done in a random subset k$\leqslant$ d(dimension of the dataset) so you don't take all the features.In this way each classifier will work on randomly selected features so they all will be different and make different errors. Afterwards with Random Forest I will take the mean of all those classifiers. One of the advantages of random forest is that it introduces a degree of uncertainty.

Essentially:

- Sample dataset into m datasets of the same size
- For each dataset train a decision tree by randomly choosing k features $\leqslant$ d ($\mathcal{D}$ dimension)
- Construct the final classifier

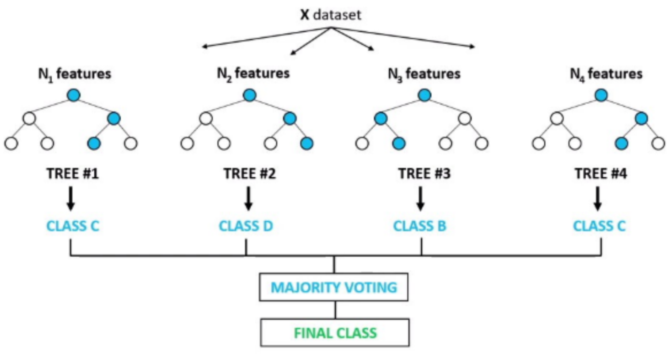


Figure 4.4. Random Forest.

The algorithm works in 4 steps

- Select random samples from a given dataset
- Construct a decision tree for each sample and get a prediction result from each decision tree
- Perform a vote for each predicted result
- Select the prediction result with the most votes as the final prediction

Random forest uses gini importance coefficient for calculating the feature importance. Gini importance is also known as the total decrease in node impurity. This is knowing how the algorithm change and how does it affect in the accuracy when dropping a feature.

4.2.2 Logistic Regression

Logistic regression algorithm is based in the concept of probability. It uses a linear function defined as the 'Sigmoid function' or also known as the 'logistic function' in order to divide the 2 classes. The main distinction of logistic regression with respect to perceptron is that firstly you model the data (both classes) with a probability distribution and divide the two distributions rather than the data itself. LR estimates P(y|x) by maximizing the log-likelihood. I want to know what is the probability the given x(an observation) it belongs to class y.
Likelihood responds to the question: "What does it make my data look like this". Assuming that each observation x is generated by some parameters Θ the aim is to find the parameters Θ for which model explains best observations.In statistics,*maximum likelihood estimation (MLE)* is a method of estimating the parameters of a probability distribution by maximizing a likelihood function, so that under the assumed statistical model the observed data is most probable.

$$P(y|x) = 1/1 + e^{-w^T+b}w, b = argmax_{w,b}\Pi P(y_i|x, w, b) \tag{4.9}$$

find w,b that best explains the label of my data For computational puposes take the log

$$argmax_w = \sum_i log(\frac{1}{1 + e^{(wTx)y}}) \tag{4.10}$$

Instead of maximizing we minimize

$$argmin_w = \sum_i log(1 + e^{(wTx)y}) \tag{4.11}$$

And than minimize by using the *Gradient Descent.* In a nutshell, take the gradient of the function , multiply by a small value α and subtract it from the function. It is guaranteed that you are moving in the direction small values and being that the function is convex it is assures that you have a global minimum(gradient is zero).

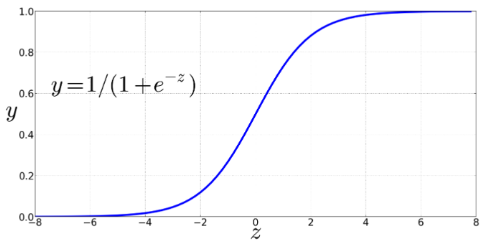


Figure 4.5. Sigmoid Function.

Logistic Regression belongs to the class of *Discriminative Classifiers*, this means that in confront with the generative models Logistic Regression tries to distinguish the classes by figuring out the features that neatly separates them

4.2.3 SGD Classifier

SGDClassifier implements a plain stochastic gradient descent learning routine which supports different loss functions and penalties for classification.[16] So it is a linear classifier optimized by the usage of *Stochastic Gradient Descent*. We can define various loss functions to the SGD Classifier and the result is one of the algorithms with the stochastic Gradient Descent, for example SVM,Logistic Regression ect.
In this paper with the SGDClassifier the logarithmic loss is defined so we train a logistic regression with stochastic gradient descent.

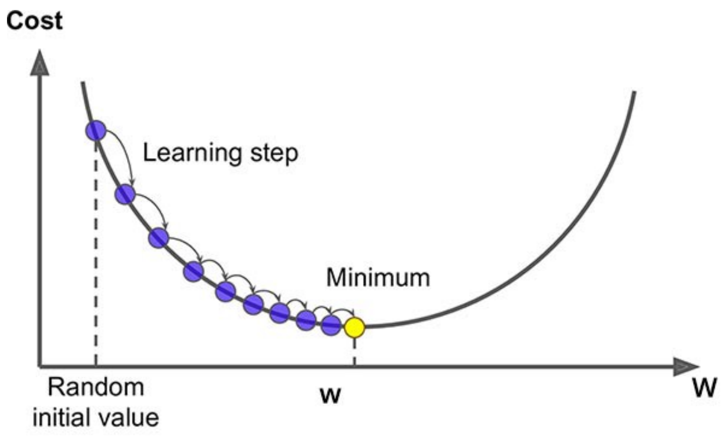
Stochastic Gradient Descent

Gradient Descent is the algorithm used in order to minimize the loss function. In Machine Learning Algorithms essentially we have a minimization technique. Once we have the loss function we than want to minimize it until finding the minimum value of it. In the minimum we have smallest error. Gradient Descent essentially works in different step

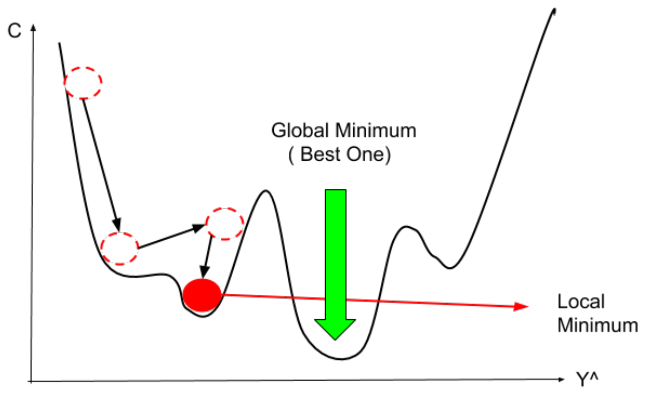
- Calculate the gradient
- Start in a random position
- Update the gradient each time
- Calculate the step sizes η : η = gradient * learning rate
- Update new parameters : new params = old params -step size
- loop until gradient close to 0

The learning rate parameter is crucial for convergence purposes. Larger learning rates make the algorithm take huge steps and in that case it would not get the minimum but pass it. So, it is advisable to take low values of the learning rate.
The Gradient Descent Algorithm is great but it presents a challenge.We have a huge amount of computation. The algorithm will continue until it has found a minimum. With Stochastic Gradient Descent we introduce the notion of randomness. Essentially *SGD* randomly picks one sample for each step and use that sample for calculating derivatives. It reduces the amount of terms computed by a very large factor and it is very useful when we have redundancy in the data. Frequently it is used mini-batch gradient descent that is sampling more data points at each step rather than just one point. The purpose is to

have a balance between the speed of stochastic gradient descent and the result of gradient descent



(a)



(b)

Figure 4.6. (a) Gradient Descent, (b) Stochastic Gradient Descent.

4.3 LightGBM algorithm

LightGBM algorithm takes advantage of tree algorithms and gradient boosting to achieve the result of a fast processing algorithm. One of the biggest advantages of lightgbm is that it is very robust when it comes to computation.

Boosting

Boosting handles the term of Bias. The aim is to reduce the bias and as a consequence reduce the classification error. It works with an ensemble of weak learners that will than produce a strong learner.

$$H(x) = \sum_{i=1} \alpha \times h_1(x) \tag{4.12}$$

Boosting is very powerful and widely used.[15]

A characteristic of LGBM is that it grows leaf-wise(best-first) while other algorithms grow level-wise(depth-first),a representation is shown in 4.7. This can lead to overfitting but you can control it by defining the depth for splitting.

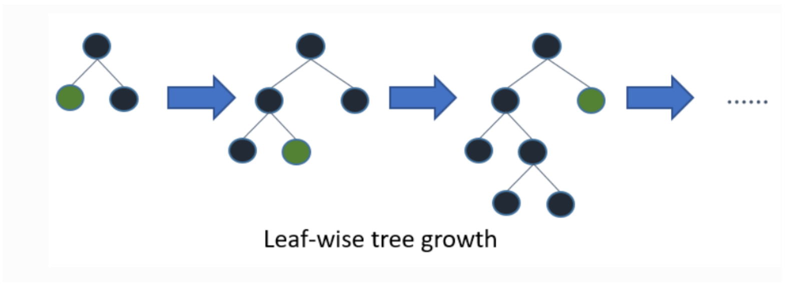


Figure 4.7. Leaf-wise Tree growth [7].

Gradient boosting decision tree (GBDT) is a widely-used machine learning algorithm, due to its efficiency, accuracy, and interpretability.[3]

Lightgbm

LightGBM buckets continuous feature (column) values into discrete baskets. This leads to a reduced memory usage and speeds up training phase. It is able to deal with very large datasets and actually it doesn't work very well with small datasets."Light" stands for the fact that it is very fast when it comes to computation.

In LightGBM algorithm 2 novel techniques with respect to GBDT. Those techniques are:

Gradient-based One-Side Sampling(GOSS)

The main idea is that instead of focusing on well learned data points we focus more on under-trained ones. The gradient is used as a metric in order to differentiate between well learned data points(small gradient) and not well learned ones(large gradient). When down sampling keep the data points with large gradient and randomly drop the ones with small gradient. Such a treatment can lead to a more accurate gain estimation than uniformly random sampling.It is demonstrated that GOSS will not lead to training accuracy loss and it will surpass the performance of random sampling. [8] When down sampling keep the data points with large gradient and randomly drop the ones with small gradient. Such a treatment can lead to a more accurate gain estimation than uniformly random sampling. It is demonstrated that GOSS will not lead to training accuracy loss and it will surpass the performance of random sampling. [8] Steps for GOSS calculation with parameters a and b:

- Sort based on absolute gradient value
- Keep top a $\times$ 100% data samples(large gradient)
- randomly drop b $\times$ 100% data samples(small gradient)
- Amplify small gradients by multiplying $\frac{1-a}{b}$ when calculating information gain

Exclusive Feature Bundling(EFB)

The aim is to reduce the number of features by taking advantage of the sparsity of large datasets. [8] Usually in real applications we notice that although there are a large number of features, we have many zeros. So we merge(bundle) them and save space. The purpose is to reduce the dimensionality of the dataset. We work with data that have many features which are mutually exclusive i.e they never take zero values simultaneously. Examples include the one-hot features (e.g., one-hot word representation in text mining). We can safely bundle such exclusive features. In order to keep the merge reversible we will keep exclusive features reside in different bins[8]

Steps for EFB proces:

- Collect information about the exclusive features that have overlapping values
- Sort by number of non-zero values, descending

- loop over features and based on t(threshold) assign a bundle or create a new one

Chapter 5

Evaluation of Machine Learning Algorithms

As stated in the above chapters evaluating Machine Learning algorithms is one of the most essential part of a project. A model may give you satisfying results when evaluated using *accuracy score* but may give poor results when evaluated against other metrics such as *logarithmic loss.* In this paper we study in deep 4 metrics.

- Accuracy Score
- Confusion Matrix
- ROC_AUC Curve
- Log Loss

5.1 Accuracy Score

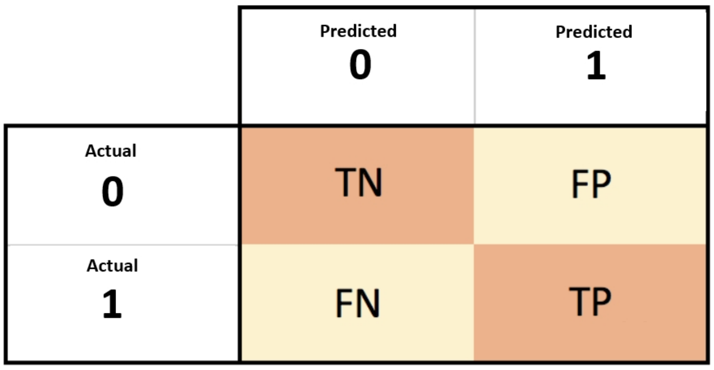
Accuracy Score is the ratio of number of correct predictions to the total number of input samples.

$$Acc\_score = \frac{NumberOfCorrectSamples}{TotalNumberOfInputSamples} \tag{5.1}$$

This metrics works well when dealing with balanced classes, which is not our case.It works well only if there are equal number of samples belonging to each class. A case in which the accuracy score is not uitable for measuring algorithm performance is the following one: Consider that there are 98% samples of class A and 2% samples of class B in our training set. Then our model can easily get 98% training accuracy by simply predicting every training sample belonging to class A.This kind of classification algorithm is referred to as *Dummy Classifier*.

5.2 Confusion Matrix

Confusion Matrix is a performance measurement for machine learning classification problems where output can be two or more classes. It is a table with 4 different combinations of predicted and actual values.By definition a confusion matrix C is such that $C(i,j)$ is equal to the number of observations known to be in group i and predicted to be in group j.



	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

Figure 5.1. Confusion Matrix.

- **True Positive (TP)**:
  It was predicted a positive value and the true value is actually positive. It was predicted that an advertisement will be clicked and it was actually clicked.

- **True Negative (TN)**:
  It was predicted a negative value and the true value is actually negative. It was predicted that an advertisement will not be clicked and it was actually not clicked.

- **False Positive (FP)**:
  It was predicted a positive value but the true value is actually negative. It was predicted that an advertisement will be clicked but it was actually not clicked.

- **False Negative (FN)**:
  It was predicted a negative value but the true value is actually positive. It was predicted that an advertisement will not be clicked but it was actually clicked.

From the Confusion Matrix we can obtain 2 very important metrics, **Recall** and **Precision**

$$Recall = \frac{TP}{TP + FN} \tag{5.2}$$

$$Precision = \frac{TP}{TP + FP} \tag{5.3}$$

Based on the type of application under study, there might be the need to increase recall or precision.
For example consider the case of a pandemic situation. In this scenario , if we want to predict weather someone is infected or not we want to have a high recall meaning that I do not want to predict that someone is not infected when it actually is. I would rather predict that is infected and deal with a false positive.
In our case we are more interested to have a high precision meaning that I am interested in having no false positives, so I do not want to predict that someone will click and than actually not click on the advertisement as it will result in cost increase.

5.3 ROC Curve and AUC

When we need to check or visualize the performance of an algorithm that deals with a classification problem, we use the *AUC (Area Under The Curve) ROC (Receiver Operating Characteristics)* curve. It is one of the most important and most used evaluation metrics for checking any classification model's performance.

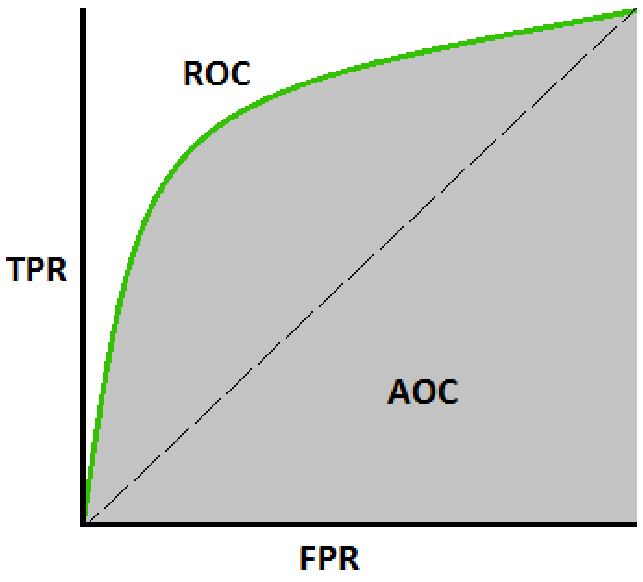


Figure 5.2. ROC_AUC Curve.

A machine learning classification model can be used to predict the actual class of the data point directly or predict its probability of belonging to different classes. This probability gives us more control over the result. We can determine our own threshold to interpret the result of the classifier. In a binary classification problem this threshold is 0.5 . By having the opportunity of changing the threshold we are able to decrease or increase based on the problem we are dealing.

Setting different thresholds for classifying positive class for data points will provide different results when thinking about the confusion matrix. And one of these thresholds will probably give a better result than the others, depending on whether we are aiming to lower the number of False Negatives or False Positives. The ROC Curve summarizes this information, meaning that we change our threshold and each time we save the results about True Positive Rate and False Positive Rate. AUC is the area under ROC Curve. The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes. AUC is used in order to compare the performance between algorithms. If an algorithm has a greater AUC with respect to another one than this algorithm is better in terms of prediction.

5.4 Logarithmic Loss

Log Loss is the most important classification metric based on probabilities.It quantifies the accuracy of a classifier by penalising false classifications. Log loss will become higher whenever an algorithm does a lot of mispredictions and viceversa. Log Loss is defined as:

$$-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} log p_{ij} \tag{5.4}$$

where:

- N is the number of samples
- M is the number of labels
- y_{ij} is a binary indicator of whether or not label j is the correct classification for instance i
- p_{ij} is the model probability of assigning label j to instance i

In our case , so we have M=2 as the variable y can take only 2 values: y=1(click) and y=0(not click) the function takes the form :

$$-\frac{1}{N}\sum_{i=1}^{N}(y_i log p_i) + (1 - y_i)(1 - log p_i) \tag{5.5}$$

Log loss is widely used as a measure to confront different algorithms performance. If in the same data set one algorithm has a lower value of log loss than another one than that algorithm is better.

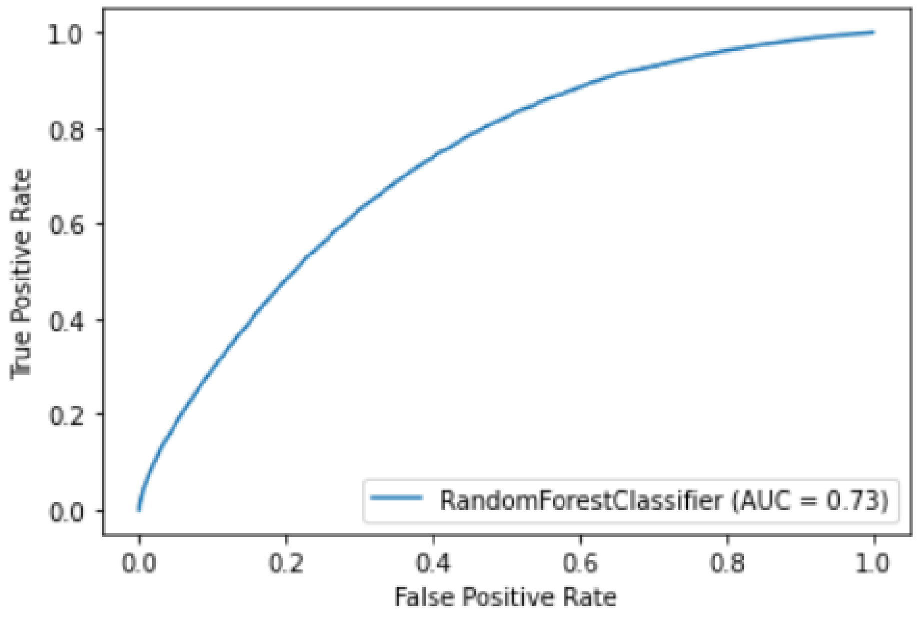
Chapter 6

Results and Conclusion

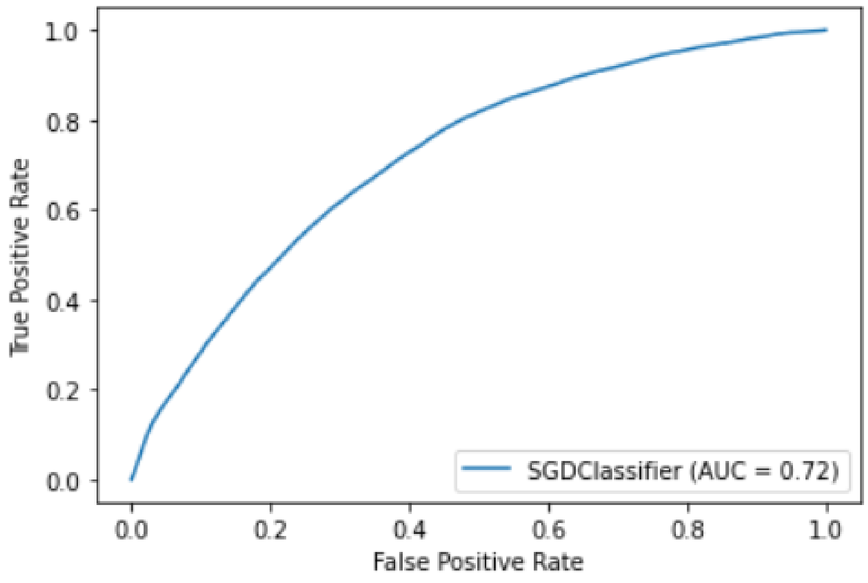
In this chapter the results of the algorithms described in chapter 4 are reported by focusing on the metrics presented in chapter 5. In this paper the focus is mostly on 2 metrics, the AUC(area under the curve) and Logloss(logarithmic loss). AUC essentially shows how good is the algorithm in distinguishing between click and not click classes while the logloss shows how much was the algorithm penalized when doing a misclassification.

6.1 Baseline algorithms

In a classification problem a model will calculate the probability for each example that belongs to a class. In order to distinguish between classes the threshold is 0.5 meaning that if the probability for a certain example is > 0.5 than it is a click otherwise it is not a click. By varying this threshold, calculating **False positive rate** and **True positive rate** each time we extract the ROC Curve. The Area under the curve (AUC) will essentially give the information about the performance of the algorithms. Closer to 1 better the algorithm. In figure 6.1 is shown the ROC Curve about the 2 baselines algorithms used in the paper. We can notice that the Random Forest algorithm presents an AUC=0.73 and the Logistic Regression algorithm presents an AUC= 0.72. Both algorithms perform in a good way. Furthermore, we take into consideration the logloss metric. This metric will give an idea on how much is the algorithm mistaken the probability that it is assigning to misclassified examples. For Random Forest Classifier we have a logloss= 0.53 meaning that it is mistaking by a very large amount the misclassified examples while for SGD the logloss=0.41 which is acceptable and way better that random forest.



(a)

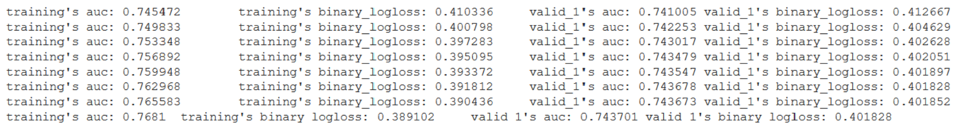


(b)

Figure 6.1. (a)Random Forest AUC, (b)SGD AUC.

6.2 Lightgbm

The lightgbm algorithm, as described in chapter 4 is very powerful and highly performing. In this algorithm we also have the possibility to introduce a validation set. The purpose is to validate the result while training. In the figure 6.2 we can observe the results. In the training set we achieve a maximum value of AUC of 0.7681 and a minimum value of logloss of 0.3891 while in the validation set we achieve an auc of 0.74 and a logloss of 0.40.



```
training's auc: 0.745472       training's binary_logloss: 0.410336     valid_1's auc: 0.741005 valid_1's binary_logloss: 0.412667
training's auc: 0.749833       training's binary_logloss: 0.400798     valid_1's auc: 0.742253 valid_1's binary_logloss: 0.404629
training's auc: 0.753348       training's binary_logloss: 0.397283     valid_1's auc: 0.743017 valid_1's binary_logloss: 0.402628
training's auc: 0.756892       training's binary_logloss: 0.395095     valid_1's auc: 0.743479 valid_1's binary_logloss: 0.402051
training's auc: 0.759948       training's binary_logloss: 0.393372     valid_1's auc: 0.743547 valid_1's binary_logloss: 0.401897
training's auc: 0.762968       training's binary_logloss: 0.391812     valid_1's auc: 0.743678 valid_1's binary_logloss: 0.401828
training's auc: 0.765583       training's binary_logloss: 0.390436     valid_1's auc: 0.743673 valid_1's binary_logloss: 0.401852
training's auc: 0.7681  training's binary logloss: 0.389102     valid 1's auc: 0.743701 valid 1's binary logloss: 0.401828
```

Figure 6.2. Lightgbm AUC.

Moreover, we can notice (figure 6.3) how the logloss decreases with the increasing of the number of iterations and how it stabilize in the case of the validation set.

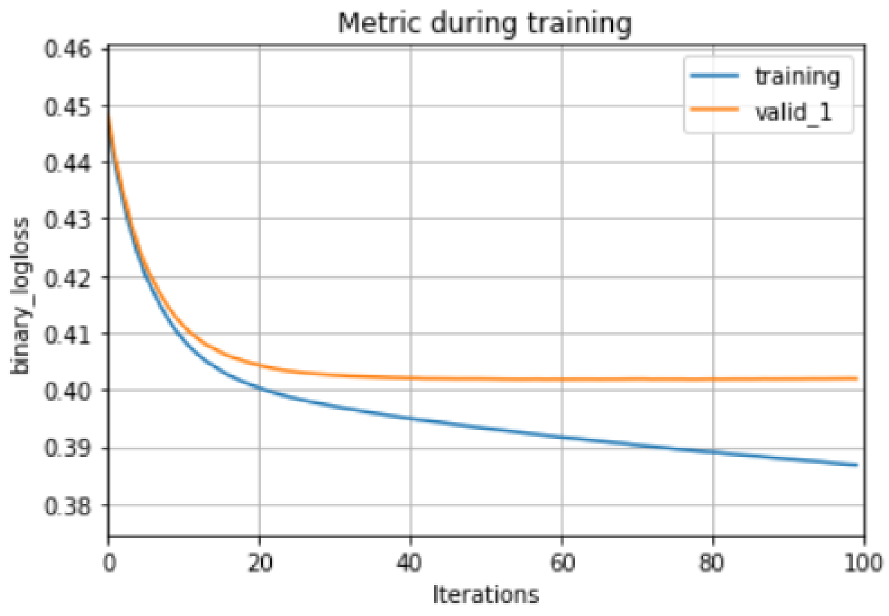


Figure 6.3. Logloss during training phase, train set and validation set .

Finally the ROC Curve is presented in figure 6.4 after making the predictions on the testing set. The curve is very stable and achieves an AUC of 0.74.

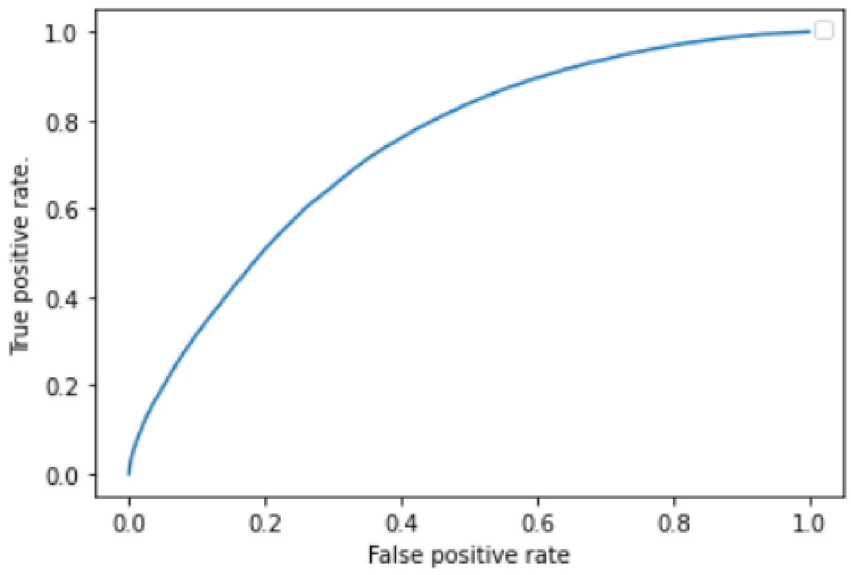


Figure 6.4. ROC Curve Lightgbm.

The best result is obtained from Lightgbm algorithm having the lowest value of logloss and highest of AUC. Following the Logistic Regression Classifier and the Random Forest Algorithm.
We notice that the RF algorithm has a bad performance when using logloss evaluation metrics but not so bad with auc. This is due to the fact that logloss metric also depends on the probability that the algorithm assigned to a misclassified sample. For example, if the algorithm predicted with high probability that a certain observation was going to be a click but it was not a click than the quantity of penalization is higher.
In this paper is proposed a model for CTR prediction with both good accuracy and generalization ability. The results are in line with what was expected at the beginning of the project.

6.3 Discussion and Future Work

CTR metric is widely used and not only on the advertisement area but also in other fields such as search engines in order to understand how well it correlates the content to user queries so an interesting future work could be in this area.
Furthermore, in this paper we discuss and predict the expected CTR for a new advertisement. We could exploit these models and data analysis in order to inform advertisers what is the trend of marketing and what could they improve in order to increase their CTR%.
Finally, expand the analysis on more than advertisers and advertisements. Collecting information over time about the general quality of an advertiser. a model which is time dependent and can keep up with the trend by updating the CTR of all the advertisements while collecting the data.

Bibliography

[1] Weiran WangandMiguel éA. Carreira-Perpinanan. «The Role of Dimensionality Reduction in Classification». In: (2014).

[2] Statista Research Department. *Online advertising revenue in the United States from 2000 to 2020.* 2021.

[3] Jerome H. Friedman. «GREEDY FUNCTION APPROXIMATION:A GRADIENT BOOSTING MACHINE». In: (2001).

[4] Trevor Hastie Robert Tibshirani Jerome Friedman. *The Elements ofStatistical LearningData Mining, Inference, and Prediction.* 2009.

[5] *Handling Categorical Variables.*

[6] M. Kamber J. Han and J. Pei. *Data Mining: concepts and techniques, 3rd ed. Waltham, MA: Morgan Kaufmann, 20.* 2011.

[7] *leaf wise tree growth.*

[8] Guolin Ke Qi Meng Thomas Finley Taifeng Wang Wei Chen Weidong Ma Qiwei Ye Tie-Yan Liu. «LightGBM: A Highly Efficient Gradient Boosting Decision Tree». In: (2017).

[9] *Machine Learning.*

[10] *MinMaxScaler.*

[11] *One Hot Encoder.*

[12] *Radom Fores algorithm.*

[13] *RandomizedSearchCV.*

[14] D. Kanellopoulos S. B. Kotsiantis and P. E. Pintelas. «Data Preprocessing for Supervised Leaning». In: (2006).

[15] Robert E. Schapire. «A Brief Introduction to Boosting». In: (1999).

[16] *SGD Classifier.*

[17] *Sklearn OrdinalEncoder.*

[18] *Sklearn Pipeline.*

[19] *Sklearn train_ test_ split.*

[20] V. Vapnik. «Principles of Risk Minimization for Learning Theory». In: (1992).