Predicting Advertisement Clicks Based on Context

Ashok Vardhan Kari, Karthik Shivkumar, Vamshidar Reddy Chaulapally

***Abstract*— Context ads are the best way to target users with goods and services. Currently, Avito uses general statistics on ad performance to drive the placement of context ads. Their existing model ignores individual user behavior, making it difficult to predict which ad will be the most relevant for (and earn the most clicks from) each potential buyer.**

Avito.ru is a Russian classifieds website that sells a wide variety of products. The company has shared some of its data to as challenge for modelers. The objective is to predict whether an individual (buyer or a seller) will click on an ad based on the context of the ad and the behavior of the individual.

We will also attempt to answer a few other questions based on the data such as targeted advertising and user profiling.

***Index Terms*— Contextual ads, Targeted advertising, Regional Advertising, User classification, Marketing Attribution.**

I. INTRODUCTION

In Russia, if you're looking to sell a tractor, a designer dress, a vintage lunchbox, or even a house, your first stop will likely be Avito.ru. As the largest general classified website in Russia, Avito connects buyers and sellers across the world's biggest country.

Sellers are highly motivated to place ads on Avito, hoping to gain attention from the site's 70 million unique monthly visitors. There are three different types of ads available to sellers on Avito: regular, highlighted, and context.

Context ads are the best way to target users with goods and services. Currently, Avito uses general statistics on ad performance to drive the placement of context ads. Their existing model ignores individual user behavior, making it difficult to predict which ad will be the most relevant for (and earn the most clicks from) each potential buyer.

With the data Avito has made available, we will be attempting to answer some commonly asked questions about online advertising.

A. Predict Ad Click

Given the data about user behavior, context of ads and other variables, we will attempt to predict whether an ad will be clicked on. Once a certain predictive power is obtained, it would be possible to invest in only those ads that yield a good return.

B. Relationship between Ad performance and user variables.

Here we will attempt to ascertain if a correlation exists between ads and users. Specifically, ad variables and user behavior variables. This will allow marketers to identify the type of ads that are most effective for the different types of customers.

C. Improving Ad Performance

Here we will attempt to establish if having a user logged in significantly improves the performance of advertisements. The underlying intuition being that, if a user is logged in; there is more user information available. Given this user information, it is possible to optimize ads.

D. Factors driving Ad Popularity

By identifying the key factors/variables that directly affect ad performance, it is possible to improve future advertising. For example, it would be possible to identify the variable that makes an ad successful and improve ads. Similarly, it would also be possible to identify factors that cause an ad to fail and remove them from newer ads.

E. Relationship between ad Performance and Position

The idea that the position of an ad on a web page affects its performance is well established. Here, we will attempt to establish what are the factors that affect this and if they have different effects on different ads.

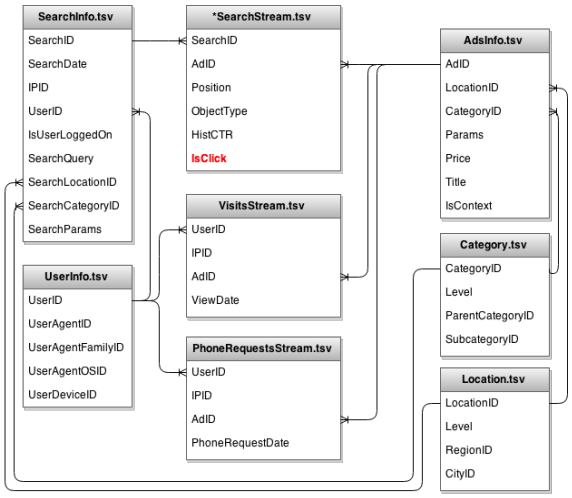
F. Other Prospective Analysis

We assume that as we progress through our work, we will stumble upon more questions that we could attempt to answer. Some of these would possibly more valuable and more insightful. If that is the case, we will include these analysis in the final report.

II. DATA DESCRIPTION

There are 8 relational datasets. All these files were encoded in UTF-8 and stored into tab-separated format (.tsv).

There is also an SQLite database (database. SQLite) alternative with all data available. Relationships between the datasets are captured in the following schema:



trainSearchStream.tsv, testSearchStream.tsv

These two files are the main datasets. trainSearchStream is a random sample of previously selected users' searches on avito.ru during at least 16 consecutive days from April'25 until the target impression.

Regular ads are shifted down constantly as new ads come in. (Normally, a visitor's search results are sorted by the time an ad is submitted to Avito). Each line in the file describes one "impression" (an ad that is shown to a particular user based on a search). testSearchStream shares
the same format, except the target variable field "IsClick" is omitted.

VisitsStream.tsv, PhoneRequestsStream.tsv

These are samples of users' visits to non-contextual ad landing pages and the corresponding phone request (if one occurred). Each ad's landing page shows the hidden seller's phone number. To be able to contact the seller, the user needs to click the request phone button:

Consequently, a user's phone request event could be considered a proxy for a user's response to the advertisement. We believe that clicking the phone request indicates a high level of interest in the ad.

SearchInfo.tsv

Details of Search by different users. Date and time they searched at, query they searched for, location they searched from, category they searched about and filters they applied.

AdsInfo.tsv:

Information about each ad such as their type, parameters, price etc. Note that both Params from AdsInfo.tsv and SearchParams from SearchInfo.tsv shares same dictionary (keys and values). The Params are semi-structured to reflect the nature of the search and the product. For example, the Params for clothing might be gender, size, color, brand; while the Params for houses might be size, # of bedrooms,
# of bathrooms, etc.

UserInfo.tsv:

Anonymized user information such as user identifier, user's browser family, user's OS and user's device type.

Location.tsv

Geo tagging information. Data includes city, region and country information.

Category.tsv

Ad category information consisting of parent category/ sub category.

Data Splitting

The dataset contains sample of users and their behavior. For each user, one target impression between time point A (May, 12) and time point B (May, 20) was selected randomly. The task of this competition is to provide the probability that a user will click on the selected target ad, given all the information generated by the user from the Start time point (April, 25) until the target impression.

Note that some users from the testSearchStream may not have any historical information in VisitStream, PhoneRequestsStream and trainSearchStream as some of them may be new registered or had no activity within Start - Test event time interval.

III. RELATED WORK

The marketing ecosystem is extremely diverse and is ever growing. There are different aspects to the multiple techniques to optimize marketing. They can range from bidding on keywords on search engines and bidding on ad spaces to forming affiliates and promotions to target specific customers. The domain is so diverse that "https://www.cabinetm.com" releases a periodic report on how the domain has progressed and who the major players are.

Some major players like Google AdSense, Facebook Insights, hotjar provide multiple services along lines very similar to this project. Researchers at Google [1] developed a system that processes training data in a streamed fashion, which was applied to sponsored search. However, our objective is to connect the various dimensions of online marketing and present it in a neat and succinct manner. The idea is developing a prototype that will aid in marketing decision making by showcasing the inter-connectedness of various marketing aspects. For example, changing the position of the ad might result in more clicks, but what is the effect on the type of customer?

Traditionally, online advertising falls into two categories: sponsored search [18] and contextual advertising [27]. Sponsored search advertising displays advertisements onto the result page of a particular query submitted to a search engine. Google is the first to incorporate click feedback into the model [3]. In order to estimate CTR for new ads, Richardson et al. [28] explored different types of features (ads, terms, and advertisers) to train a logistic regression model. Considering the rapid growth of data volume, Ciaramita et al. [9] employed online learning using the perceptron algorithm. Graepel et al. [15] addressed the same problem by using a scalable Bayesian algorithm. Contextual advertising refers to ads placement within the content of regular web pages. A large body of work is devoted to learning the relevance of the displayed ads to the page content. Features identifying both semantic and syntactic relationship of words between the Web page and ads are included in [6]. Semantic relationships of words between Web pages and ads are modeled by hidden classes in [26]. Murdock et al. [23] developed a system that uses features derived from statistical machine translation models, aiming to learn a "translation" of vocabularies between ads and target pages. Chakrabarti et al. [8] applied logistic regression to learn the match between ads and Web pages. Bai et al. [2] proposed a supervised version of latent semantic indexing to map the query document pair to a ranking score. In order to learn the ranking function, genetic programming is adopted by Lacerda et al. [20]. Karimzadehgan et al. [19] proposed a stochastic algorithm to learn a ranking function that directly optimizes IR evaluation metrics, which are usually not differentiable.

IV. PROPOSED APPROACHES

This section introduces the proposed approaches. We want to buil a reporting platform, which can be used to monitor the a performance, finding the most important factors that drive a popularity. Intuitively finding relationship between ad performanc and user variables.

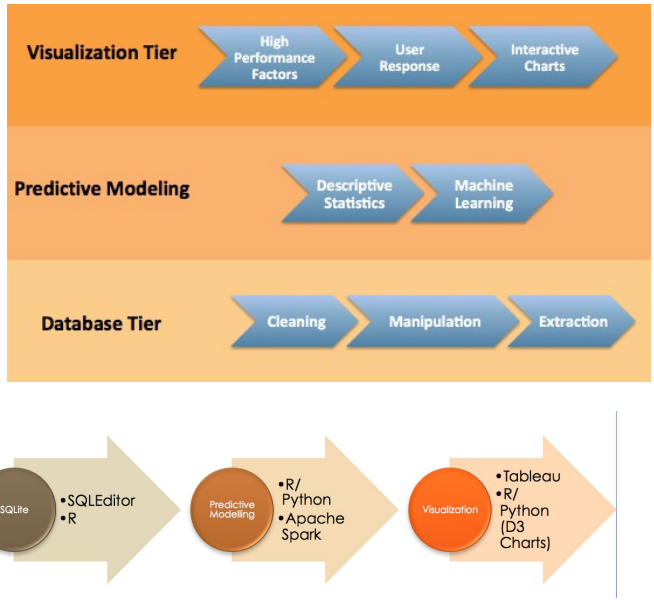
A. Tools

We have a SQLite database with all the data, so for hosting this we'll use an open source SQLite database. SQLEditor will be used to extract and manipulate information in the database. For creating predictive models we're planning to use R/ Python. The database we're provided with is 40GB. If we face latency issues we're planning to leverage Apache Spark for distributed computing.

For visualization standalone Tableau will be used. For interactive purposes we leveraged shiny charts with R.

V. SYSTEM DESIGN

A. Early Stage, High Level Design



B. EXPERIMENT AND VISUALIZATION

Extracting Data from SQL database
We have a database of 40GB which we received from Avito. We're given 8 tables in the DB, we had to do multiple joins to make our data analysis ready.
Here is a snippet of code that pulls data from multiple tables and stores as a dataframe named "Loc_Reg_Time" in R.

```
Loc_Reg_Time <- fetch(db,"SELECT Location.RegionID, SearchInfo.IsUserLoggedOn, trainSearchStream.IsClick,
                  Category.ParentCategoryID, SearchInfo.SearchID, VisitsStream.ViewDate FROM SearchInfo
                  INNER JOIN Location ON Location.LocationID=SearchInfo.LocationID
                  INNER JOIN Category ON SearchInfo.CategoryID=Category.CategoryID
                  INNER JOIN trainSearchStream ON trainSearchStream.SearchID=SearchInfo.SearchID
                  INNER JOIN VisitsStream ON VisitsStream.UserID=SearchInfo.UserID
                  WHERE trainSearchStream.IsClick=1
                  LIMIT 1000000")
```

We've about 6 queries like these and each of them took on an average of 5-6Hrs to run.

Predict Ad Click
Predicting the click of an ad based on user behavior, context of ads and other variables. Here we're enlisting different models and different combinations of features chosen to build those models.

Feature Engineering
We've taken two different combinations of features to build models.

1. HistCTR + Position
2. Position + HistCTR + CategoryID + Price

Feature	Description
HistCTR	History Click Through rate of an ad
Position	Position of ad on a webpage
CategoryID	Ad Category
Price	Price of the item listed in the ad

Machine Learning Models
We built couple of different models to predict this.

Logistic Regression:

Logistic regression (Sayad, n.d.) predicts the probability of an outcome that can only have two values (i.e. a dichotomy). The prediction is based on the use of one or several predictors (numerical and categorical). A linear regression is not appropriate for predicting the value of a binary variable for two reasons:

a. A linear regression will predict values outside the acceptable range (e.g. predicting probabilities outside the range 0 to 1)
b. Since the dichotomous experiments can only have one of two possible values for each experiment, the residuals will not be normally distributed about the predicted line.

$$\log \frac{p(x)}{1-p(x)} = \beta_0 + x \cdot \beta$$

Beta is calculated as

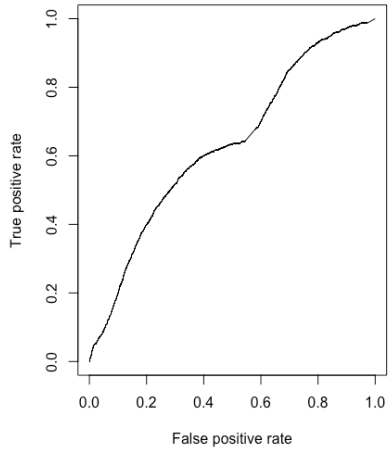
$$p(x;b,w) = \frac{e^{\beta_0 + x\cdot\beta}}{1+e^{\beta_0 + x\cdot\beta}} = \frac{1}{1+e^{-(\beta_0 + x\cdot\beta)}} \qquad 4$$

Feature Combination 1 (HistCTR + Position)
A generalized logistic regression model is built in R.

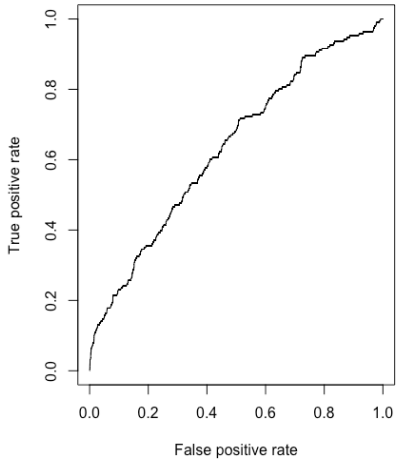
AUC (Area under the curve)
When using normalized units, the area under the curve (often referred to as simply the AUC) is equal to the probability that a classifier will rank a randomly chosen positive instance, higher than a randomly chosen negative one (assuming 'positive' ranks higher than 'negative'). This can be seen as follows: the area under the curve is given by (the integral boundaries are reversed as large T has a lower value on the x-axis)

$$A = \int_{\infty}^{-\infty} \mathrm{TPR}(T)\mathrm{FPR}'(T)\,dT = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(T' > T) f_1(T') f_0(T)\,dT'\,dT = P(X_1 > X_0)$$

Output for feature combination 1
AUC - 0.6316261



Feature Combination 2 (Position + HistCTR + CategoryID + Price)
AUC - 0.6327004



Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set

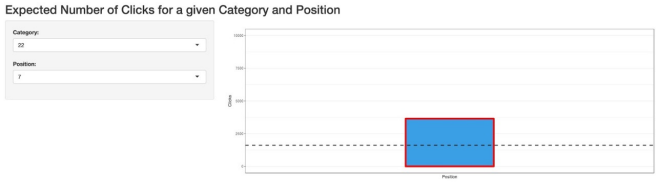
$$\hat{y} = \frac{1}{m}\sum_{j=1}^{m}\sum_{i=1}^{n} W_j(x_i, x')\, y_i = \sum_{i=1}^{n}\left(\frac{1}{m}\sum_{j=1}^{m} W_j(x_i, x')\right) y_i.$$

Outputs

	Feature Set 1	**Feature Set 2**
RMSE	0.07681	0.07942
RSquared	0.0026	0.0036

The models did not perform very well because the data is highly skewed. We have a ratio of 1000:6 (0/1) binary classification of information. For every thousand ads there are only 6 clicks. So the models built on this data did not give us any high accuracy but decent enough results.

Visualization of No.of Clicks to a given category and Position built using Shiny.



This is an interactive graph with values for Category and Position ranging from 1-8.

We sketched the plan for answering the second question using Tableau, which is to evaluate the performance of an ad based on the user behavior on the website.

Cost Sensitive Learning

Classification, Simple models work best in case of large amounts of data, such as logistic regression model or a simple decision tree

Cost-Sensitive Learning is a type of learning in data mining that takes the misclassification costs (and possibly other types of cost) into consideration. In case of an unbalanced data set, where the number of instances of one class maybe many more than the number of instances of another class, a cost sensitive learning algorithm will heavily penalize the algorithm for misclassification. Since our data had far fewer instances of clicks, a cost sensitive algorithm was an approach we dabbled with.

We also tried a synthetic generation technique. This did not yield a good result as there was not enough diversity in the existing data points. The idea behind a synthetic generation technique is to artificially creating data points that are similar to observation that are few in number. This creates a larger set for the model to sample from.

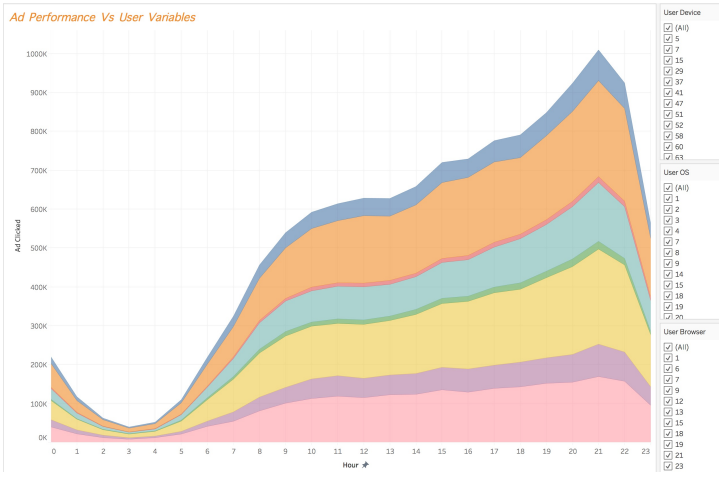
Since our classification algorithm didn't yield very good results, we tried an alternative approach. We tried to predict the probability of a click. This allowed us to use a regression model with continuous value of probability ranging from 0 to 1.

Platform based user profiling

Given the size and diversity of data, one of the possible exploratory paths can be to see how costumer visiting the website from different platforms perform. The intuition behind such a thought process is that, it is possible that users with expensive devices, might be more likely to buy more expensive products while users with economical hardware might be more likely to buy cheaper products.

A similar possibility is that users of some platforms have an affinity to buy certain category of products while users of another platform are likely to buy a different set of products.

The drop down will allow for the exploration of clicks in various categories based on the device being used, the operating system and the browser.



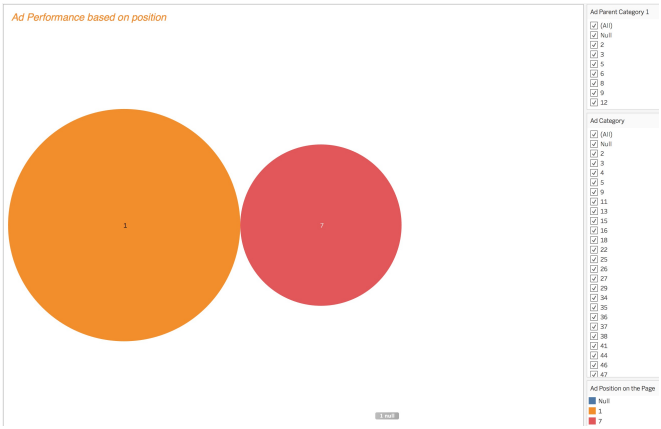
This plot talks about the total number of clicks received throughout the day. As shown in the plot, we can see a dip in the number of clicks received during early hours of the day (2AM-4AM) and significant rise towards the end of the day. Each color in the plot represents different category of an ad.

Ad Performance based on position

Another exploratory path can be to see how costumer visiting the website clicks on ads positioned differently. The intuition behind such a thought process is that some users prefer ads being non-intrusive while other users are likely to click on ads if they "stumble" on it.

Another set of user might prefer ads that anchor themselves on the left or right of the content

Some categories of ads might perform better in different locations. This can also be explored by pivoting the data accordingly.



This plot particularly shows only Position 1 and 7 because all contextual ads are placed either at position 1 or 7. Size of the bubble tells you the total number of clicks received at each positon, and it tells you that ads placed at position 1 received more number of clicks than at position 7.

This type of insight will allow for advertisers to choose a position for the ad that they think is most likely to get clicks. The intended visualization to present results is as follows.

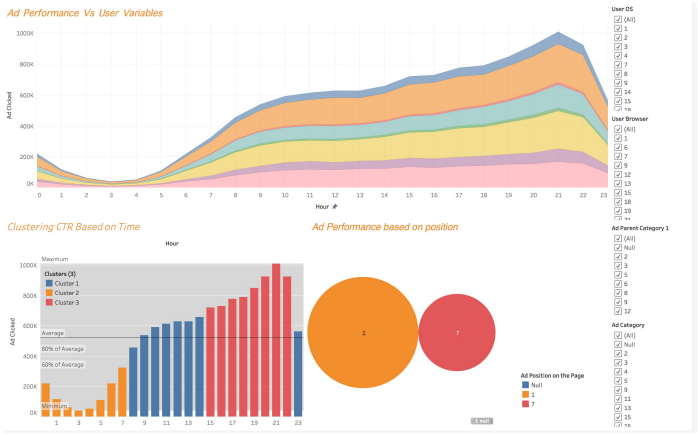
The check boxes on the right will allow for the exploration of clicks in various categories along various positions. This will give the total number of clicks you can expect when an ad placed at a certain positon.

Ad performance based on feature combination
Ad performance can depend on multiple variables or otherwise called features. The idea is that some factors might have a certain influence over performance but might perform differently in context of other variables.

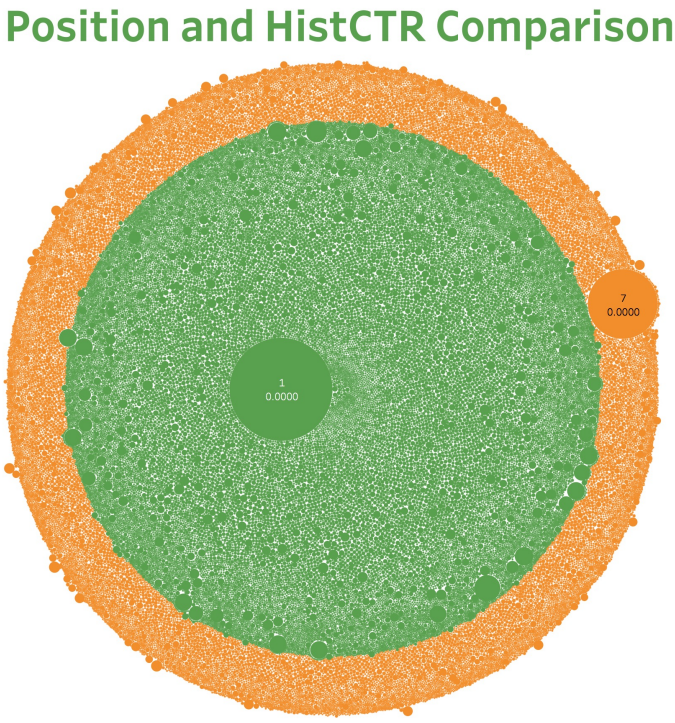
As an example, fishing rods might sell better in certain locations but the sale is also heavily dependent on the time of the year. Same is the case with umbrellas. People are more likely to buy umbrellas during the rainy season but only in locations that experiences rainfall.

The intended visualization to present results is as follows. The drop-down boxes on the left will allow for the filtering of data based on multiple variables, factors and features. This will allow us to draw granular insights based on past performances.

Building a Dashboard
We can build dashboards in Tableau by integrating multiple worksheets. In the dashboard mentioned below, we integrated all the worksheets explained above. Once you have them in place we can pivot data in different ways.

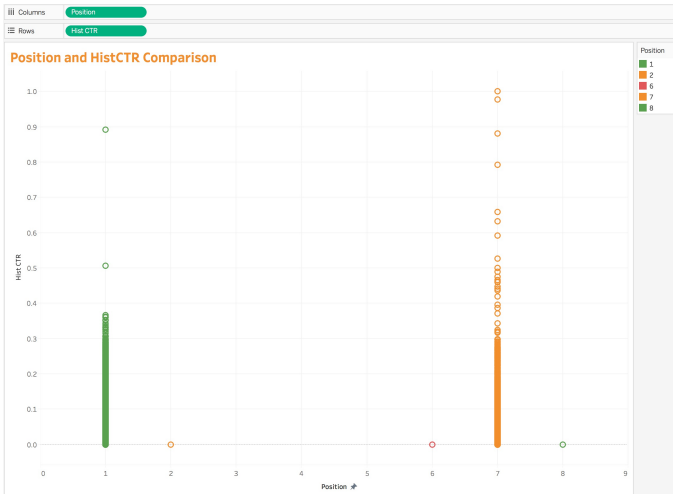


Visualizing 100 Million Data points
This is to show that how well one can visualize 100,000,000 data points in a single plot using Tableau. This graph talks about the total distribution of clicks received by all ads based on the position an ad is placed.



Inside the plot, “Orange” color represents Position 7 and “Green” color represents Position 1. And it conveys us that ads placed in position 1, received way more number of clicks compared to ads placed at position 7. It is basically understood that ads at position 1 are more likely to receive a click when compared to any other position on the page. But, we’re going to show another visualizatio which tells us a little bit of different story.

Position and HistCTR Comparison
HistCTR is basically historical Click Through Rate of an ad. The below plot tells us that, ads which have a HistCTR of 0.5 and above are more likely to receive “click” even when they are placed at positon 7.



X-Axis represent Position of an Ad Y-Axis represent HistCTR of an Ad

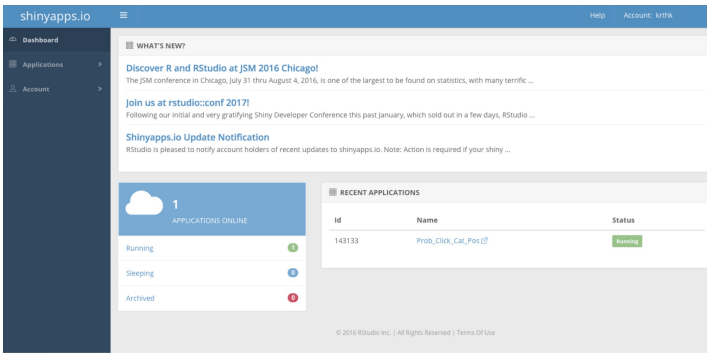
Shiny is solution provided by RStudio and its makers.

Creating Webapp using Shiny

With the use of Shiny, it was possible to create dynamic visualization's that could be hosted online. Shiny is solution provided by the makers of RStudio. The advantage of using a service like Shiny to create webapps is that the visualization can be scaled, resized and re-oriented based on the device on which it is viewed. Given that both the raw data and models can stored online, the webapp can become truly accessible online.

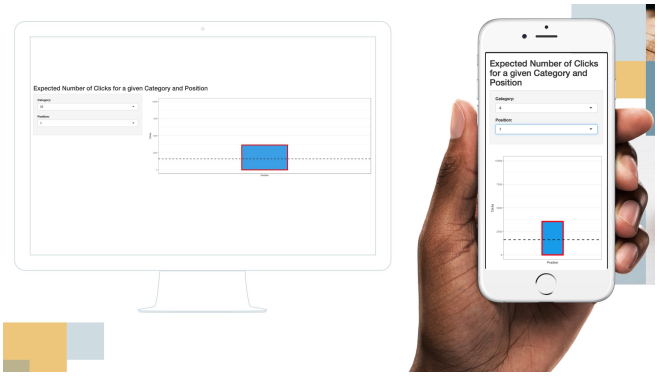
The entire data set (stored as flat files) and the models (built locally) were stored on GitHub. The objective behind doing this was that the shiny webapp could access the data and model. This would allow the webapp to make predictions based on values that are fed to it.

For the creation of the Webapp, Shiny requires that a UI script and server script be uploaded. Once this is done, an authentication is done and the WebApp is active. It can be accessed publicly using the URL.



The concept that allows for the making a shiny Webapp is a reactive variable. A reactive variable is different from a regular variable in that it re-captures the data every time the app is interacted with. Hence changing a value in a dropdown, will cause the variable to refresh and recapture the value. Once this happens, the server script runs the relevant section of code again to create a new output.
The shiny the predicts the number of clicks that can be expected, given a certain category and certain position can be found at this link:
https://krthk.shinyapps.io/Prob_Click_Cat_Pos/

Shiny can host the server script and UI script.
This is done using two scripts in R. ui.R and server.R
Shiny uses "reactive" variables.
Variable is recaptured with every interaction.



VI. APPENDIX

The drop-down boxes on the left will allow for the exploration of clicks in various categories based on the device being used, the operating system and the browser.

A. Task Assignment for each member

Task	**Major**	**Member**
Data Manipulation	Vamshi	Ashok/ Karthik
Descriptive Statistics	Ashok	Vamshi/ Ashok
Predictive Modeling	Ashok	Karthik/ Vamshi
Visualization	Karthik	Ashok/ Vamshi

B. Schedule

Date	**Due**
9/30/16	PROJECT PROPOSAL
10/28/16	PROJECT
12/2/16	PRESENTATION
12/9/16	REPORT SUBMISSION

REFERENCES

[1] Scot Brinker, "Infographic: Marketing Technology Landscape 2016" CabinetM, 2016. (https://www.cabinetm.com/marketing-technology- landscape-infographic-2016)

[2] *"What is retargeting?"*, AdRoll. (https://www.adroll.com/getting-started/retargeting)

[3] *Account Based Marketing*, AdRoll., Sept 2016. (https://www.adroll.com/resources/guides-and- reports/account-based-marketing-guide)

[4] *Avito Context Ad Clicks*, Avito.ru, Kaggle, June 2015.

[5] https://www.kaggle.com/c/avito-context-ad-clicks)

[6]https://en.wikipedia.org/wiki/Receiver_operating_characteristic#Area_under_the_curve

[7] http://www.saedsayad.com/logistic_regression.htm

[8]https://github.com/diefimov/avito_context_click_2015

C. R Codes

Extracting Data from SQL tables

```
# Loading required Packages
options(sqldf.driver = "SQLite") # as per FAQ #7 force SQLite
options(gsubfn.engine = "R")
library(parallel)
library("RSQLite")
library("sqldf")
library("caret")
library("data.table")
#connect to the database
db <- dbConnect(SQLite(), dbname="database.sqlite")
#List of tables
dbListTables(db)

# Runs the query, fetches the given number of entries and returns a
# data.table
fetch  <- function(db, query, n = -1) {
  result <- dbSendQuery(db, query)
  data <- dbFetch(result, n)
  dbClearResult(result)
  return(as.data.table(data))
}

Cat_Pos_All <- fetch(db,"SELECT AdsInfo.AdID, AdsInfo.CategoryID, trainSearchStream.IsClick,
                     trainSearchStream.Position, Category.ParentCategoryID FROM AdsInfo
                     INNER JOIN trainSearchStream ON trainSearchStream.AdID=AdsInfo.AdID
                     INNER JOIN Category ON Category.CategoryID=AdsInfo.CategoryID
                     WHERE trainSearchStream.ObjectType=3 AND trainSearchStream.IsClick=1
                     LIMIT 1000000")

write.csv(Cat_Pos_All, "Cat_Pos_IsClickAll.csv")

Cat_Pos <- fetch(db,"SELECT AdsInfo.AdID, AdsInfo.CategoryID, trainSearchStream.IsClick,
                  trainSearchStream.Position, Category.ParentCategoryID FROM AdsInfo
                  INNER JOIN trainSearchStream ON trainSearchStream.AdID=AdsInfo.AdID
                  INNER JOIN Category ON Category.CategoryID=AdsInfo.CategoryID
                  WHERE trainSearchStream.IsClick=1 AND trainSearchStream.ObjectType=3
                  LIMIT 1000")

format(object.size(Cat_Pos), units = "Mb")
Parent_Pos_IsClick <- merge(Cat_Pos, ParentCat_Pos, by.x = "AdID", by.y = "AdID")
table(Cat_Pos$IsClick)

Loc_Reg_Time <- fetch(db,"SELECT Location.RegionID, SearchInfo.IsUserLoggedOn, trainSearchStream.IsClick,
                    Category.ParentCategoryID, SearchInfo.SearchID, VisitsStream.ViewDate FROM SearchInfo
                    INNER JOIN Location ON Location.LocationID=SearchInfo.LocationID
                    INNER JOIN Category ON SearchInfo.CategoryID=Category.CategoryID
                    INNER JOIN trainSearchStream ON trainSearchStream.SearchID=SearchInfo.SearchID
                    INNER JOIN VisitsStream ON VisitsStream.UserID=SearchInfo.UserID
                    WHERE trainSearchStream.IsClick=1
                    LIMIT 100000")
```

```r
split_date <- function(Avito) {
  Avito$Years = strftime(strptime(Avito$ViewDate, "%Y-%m-%d %H:%M:%S"),"%Y")
  Avito$Month = strftime(strptime(Avito$ViewDate, "%Y-%m-%d %H:%M:%S"),"%m")
  Avito$DayOfMonth = strftime(strptime(Avito$ViewDate, "%Y-%m-%d %H:%M:%S"),"%d")
  Avito$Hour = strftime(strptime(Avito$ViewDate, "%Y-%m-%d %H:%M:%S"),"%H")
  return(Avito)
}

Loc_Reg_Time <- split_date(Loc_Reg_Time)
write.csv(Loc_Reg_Time, "Loc_Reg_Time.csv")

HistCTR_Pos <- fetch(db, "Select trainSearchStream.HistCTR, trainSearchStream.IsClick, trainSearchStream.Position
                         FROM trainSearchStream LIMIT 100000000")
format(object.size(HistCTR_Pos), units = "Mb")
write.csv(HistCTR_Pos, "HistCTR_Pos.csv")
save.image()

HistCTR_Pos_Pct10 <- fetch(db, "Select trainSearchStream.HistCTR, trainSearchStream.IsClick, trainSearchStream.Position
                         FROM trainSearchStream LIMIT 1000000")
format(object.size(HistCTR_Pos_Pct10), units = "Mb")
write.csv(HistCTR_Pos_Pct10, "HistCTR_Pos_Pct10.csv")
save.image()
```

```r
# Kaggle Competetion (Avito Contextual Ad Click)
# I got a 0.05104 on the public leaderboard

# Loading required libraries
library("data.table")
library("RSQLite")
library("caret")
library("randomForest")
library("pROC")
library("ROCR")

# Connecting to Database
db <- dbConnect(SQLite(), dbname="database.sqlite")
dbListTables(db)

# Define constants to improve readability of large number
thousand <- 1000
million  <- thousand * thousand
billion  <- thousand * million

# Creating a query which fetches required records and returns as a data frome
fetch  <- function(db, query, n = -1) {
  result <- dbSendQuery(db, query)
  data <- dbFetch(result, n)
  dbClearResult(result)
  return(as.data.table(data))
}

# Select contextual Ads (OnjectType=3)
trainSearchStreamContextual <- fetch(db, "select Position, HistCTR, IsClick from trainSearchStream where ObjectType=3", 10 * million)
m <- nrow(trainSearchStreamContextual)

# Create stratified sample
sampleSize <- 1 * million #100 * million
sampleRatio <- sampleSize / m
sampleIndex <- createDataPartition(trainSearchStreamContextual$IsClick, p = sampleRatio, list=FALSE)
trainSearchStreamContextualSample <- trainSearchStreamContextual[as.vector(sampleIndex), ]

# Compare click-ratio in full set and sample to verify stratification
print(paste("Clickratio full dataset:", sum(trainSearchStreamContextual$IsClick)/m))
print(paste("Clickratio sample:", sum(trainSearchStreamContextualSample$IsClick)/sampleSize))

# Create stratified random split ...
trainSampleIndex <- createDataPartition(y = trainSearchStreamContextualSample$IsClick, p = .80, list = FALSE)

# ... and partition data-set into train- and validation-set
trainSearchStreamContextualTrainSample <- trainSearchStreamContextualSample[as.vector(trainSampleIndex),]
trainSearchStreamContextualValidationSample <- trainSearchStreamContextualSample[-as.vector(trainSampleIndex),]

# Build a logistic regression ...
model <- glm(IsClick ~ HistCTR+Position, data = trainSearchStreamContextualTrainSample, family="binomial")
summary(model)
varImp(model)
```

```r
# Loss-function to evaluate result
logloss <- function(y, yHat){
  threshold <- 10^(-15)
  yHat <- pmax(pmin(yHat, 1-threshold), threshold)
  loss <- -mean(y*log(yHat) + (1-y)*log(1-yHat))
  return(loss)
}

# ... and predict data on validation data-set
prediction <- predict(model, trainSearchStreamContextualValidationSample, type="response")
print(logloss(trainSearchStreamContextualValidationSample$IsClick, prediction))

pred1 <- prediction(prediction, trainSearchStreamContextualValidationSample$IsClick)
perf1 <- performance(pred1, measure = "tpr", x.measure = "fpr")
plot(perf1)
auc1 <- performance(pred1, measure = "auc")
auc1 <- auc1@y.values[[1]]
auc1

# Random Forest
rfModel <- randomForest(IsClick ~ HistCTR+Position, data = trainSearchStreamContextualTrainSample, ntree=50,
                        do.trace=2,replace=FALSE,verboseiter=FALSE)

summary(rfModel)
varImp(rfModel)

conf <- rfModel$confusion
conf

# Predicting on test data-set
rfPrediction <- predict(rfModel, trainSearchStreamContextualValidationSample, type="response")
postResample(rfPrediction, trainSearchStreamContextualValidationSample$IsClick)


#=====================   More Features   ========================

# Creating more models with more features
# Extracting required fields from multiple tables
trainSearchStream<-dbGetQuery(db, "SELECT trainSearchStream.SearchID,trainSearchStream.AdID,trainSearchStream.Position,
                              trainSearchStream.objectType, trainSearchStream.HistCTR,trainSearchStream.IsClick,
                              SearchInfo.SearchDate, SearchInfo.UserID,SearchInfo.CategoryID as SearchCategoryID,
                              AdsInfo.Price,AdsInfo.LocationID,AdsInfo.CategoryID FROM trainSearchStream,
                              SearchInfo, UserInfo, AdsInfo where trainSearchStream.SearchID = SearchInfo.SearchID
                              and SearchInfo.UserID=UserInfo.UserID and trainSearchStream.AdID=AdsInfo.AdID
                              and AdsInfo.IsContext=1 limit 100000")

#Number of PhoneRequest per user
NumPhoneRequest<-dbGetQuery(db,"select UserID, count(UserID) as NumPhoReq from PhoneRequestsStream group by UserID")

#Number of View per user
NumViews<-dbGetQuery(db,"select UserID, count(UserID) as NumView from VisitsStream group by UserID ")
```

```r
trainSearchStream[is.na(trainSearchStream)]<--1
#Convert Position and Price to numeric (train)
position <- as.factor(trainSearchStream$Position)
price <- as.numeric(trainSearchStream$Price)
isClick<-as.numeric(trainSearchStream$IsClick)
HistCTR<-as.numeric(trainSearchStream$HistCTR)
LocationID<-as.factor(trainSearchStream$LocationID)
CategoryID<-as.factor(trainSearchStream$CategoryID)
SearchCategoryID<-as.factor(trainSearchStream$SearchCategoryID)


#DataFrame for train data
finalData <- data.frame("isClick"=isClick,"Position"=position,"Price"=price, "HistCTR"=HistCTR,
                        "CategoryID"=CategoryID, "SearchCategoryID"=SearchCategoryID,
                        "UserID"=trainSearchStream$UserID)

finalData$UserID<-NULL

trainIndex <- createDataPartition(finalData$isClick, p = 0.7, list=FALSE)
data_train <- finalData[as.vector(trainIndex), ]
data_train[is.na(data_train)]<-(-1)
head(data_train)

data_test <- finalData[-as.vector(trainIndex),]
data_test[is.na(data_test)]<-(-1)
head(data_test)

# Building a random forest model
rfModel1 <- randomForest(isClick ~., data = data_train, ntree=50,
                         do.trace=2,replace=FALSE,verboseiter=FALSE)
summary(rfModel1)
varImp(rfModel1)

conf1 <- rfModel1$confusion
conf1

# Predicting on test data-set
rfPrediction1 <- predict(rfModel1, data_test, type="response")
postResample(rfPrediction1, data_test$isClick)

# Building a Logistic Regression Model
glmModel <- glm(formula = isClick ~ Position + HistCTR + SearchCategoryID + Price,
                data=data_train, family = binomial("logit"))
summary(glmModel)
round(varImp(glmModel))

# Predicting on test data-set
glmPrediction <- predict(glmModel, data_test, type="response")
print(logloss(data_test$IsClick, glmPrediction))

pred <- prediction(glmPrediction, data_test$isClick)
perf <- performance(pred, measure = "tpr", x.measure = "fpr")
```

```
conf1 <- rfModel1$confusion
conf1

# Predicting on test data-set
rfPrediction1 <- predict(rfModel1, data_test, type="response")
postResample(rfPrediction1, data_test$isClick)

# Building a Logistic Regression Model
glmModel <- glm(formula = isClick ~ Position + HistCTR + SearchCategoryID + Price,
                data=data_train, family = binomial("logit"))
summary(glmModel)
round(varImp(glmModel))

# Predicting on test data-set
glmPrediction <- predict(glmModel, data_test, type="response")
print(logloss(data_test$IsClick, glmPrediction))

pred <- prediction(glmPrediction, data_test$isClick)
perf <- performance(pred, measure = "tpr", x.measure = "fpr")
plot(perf)
auc <- performance(pred, measure = "auc")
auc <- auc@y.values[[1]]
auc

# Disconnecting Database
dbDisconnect(db)
```