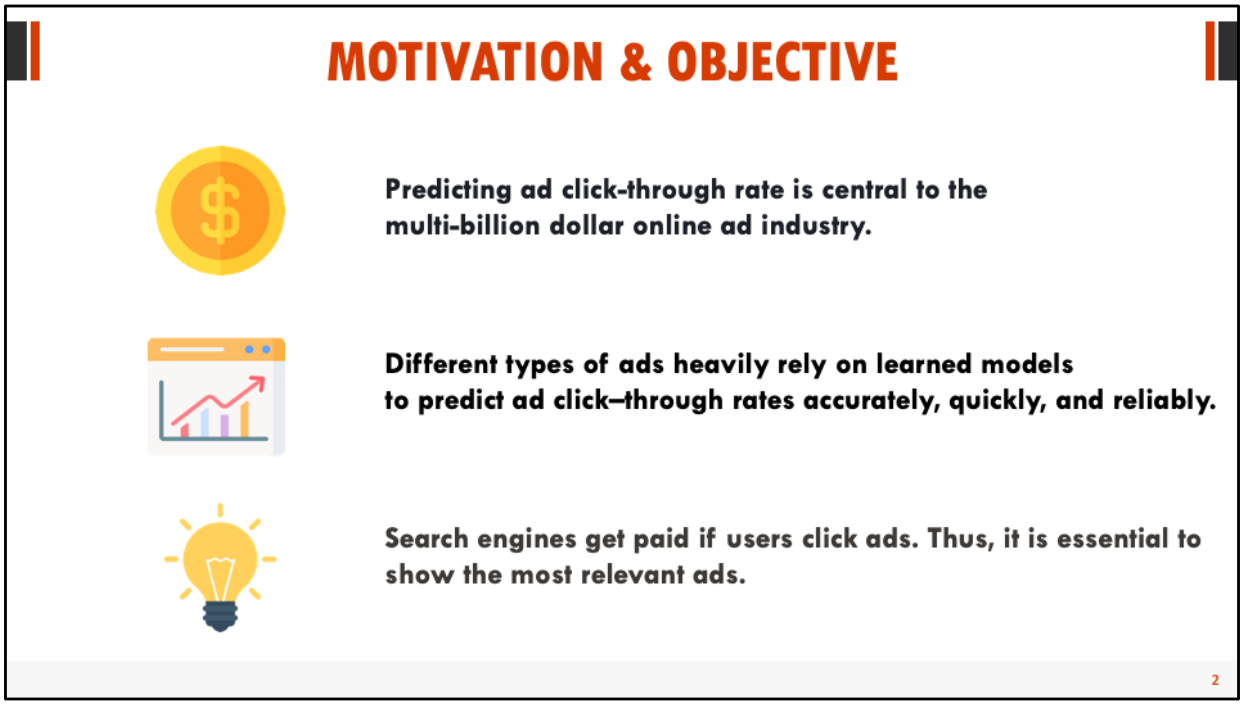

Click-through rate is a very useful metric for ranking and pricing ads in the internet marketing world. It is one of the greatest applications of machine learning. Sponsored search advertising, contextual advertising, display advertising, and real-time bidding auctions have all relied heavily on the ability of learned models to predict ad clickthrough rates accurately, quickly, and reliably.

Many proprietary search engines owned by Google, Microsoft, Yahoo etc., have effectively tackled the economic model underlying the prediction of ad CTR, which works in accordance with cost-per-click (CPC) advertising system where several ads, bidded by advertisers, are selectively picked and ranked by the product of the CPC and CTR (revenue). So the business objective for these companies centralizes on the balance between maximizing profit (and thus the CPC) and user satisfaction (and thus the CTR).



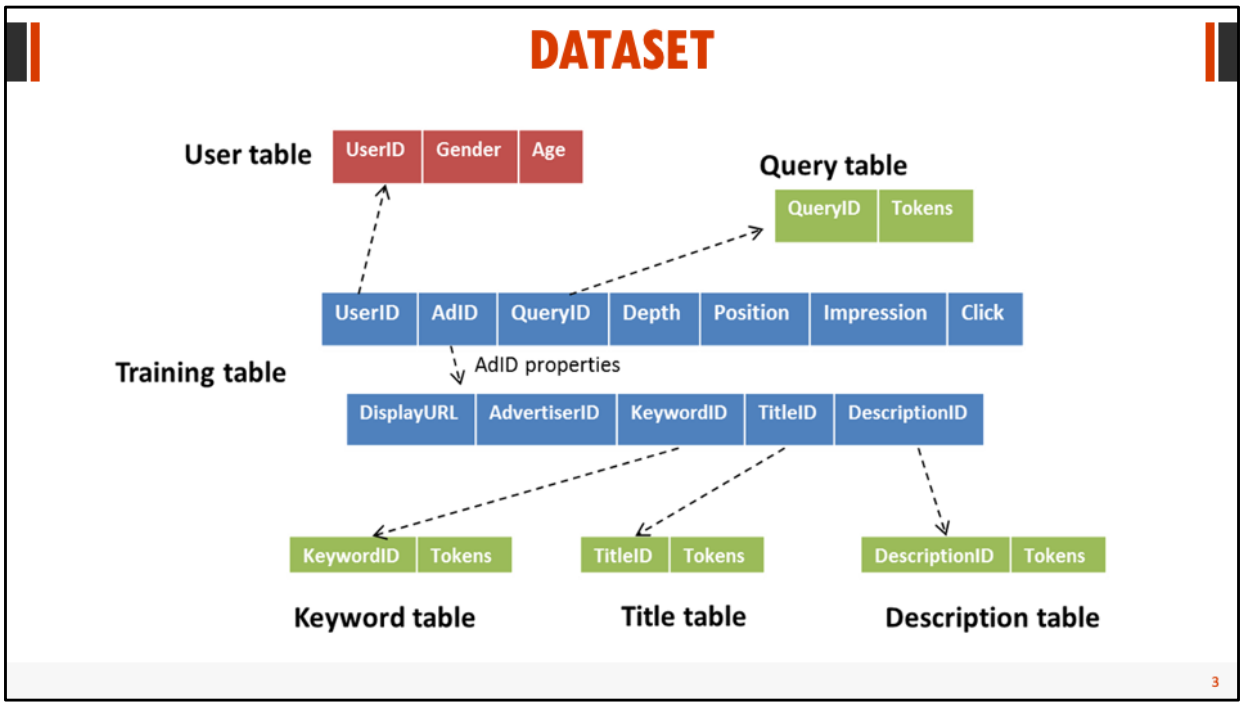

**Motivation**

Search advertising is a multi-billion dollar internet industry that has served as one of the most lucrative stories in the domain of machine learning. It relies extensively on the ability of learned models to predict ad click–through rates (CTR) accurately while promoting authenticity and low latency.

**Project Goal**

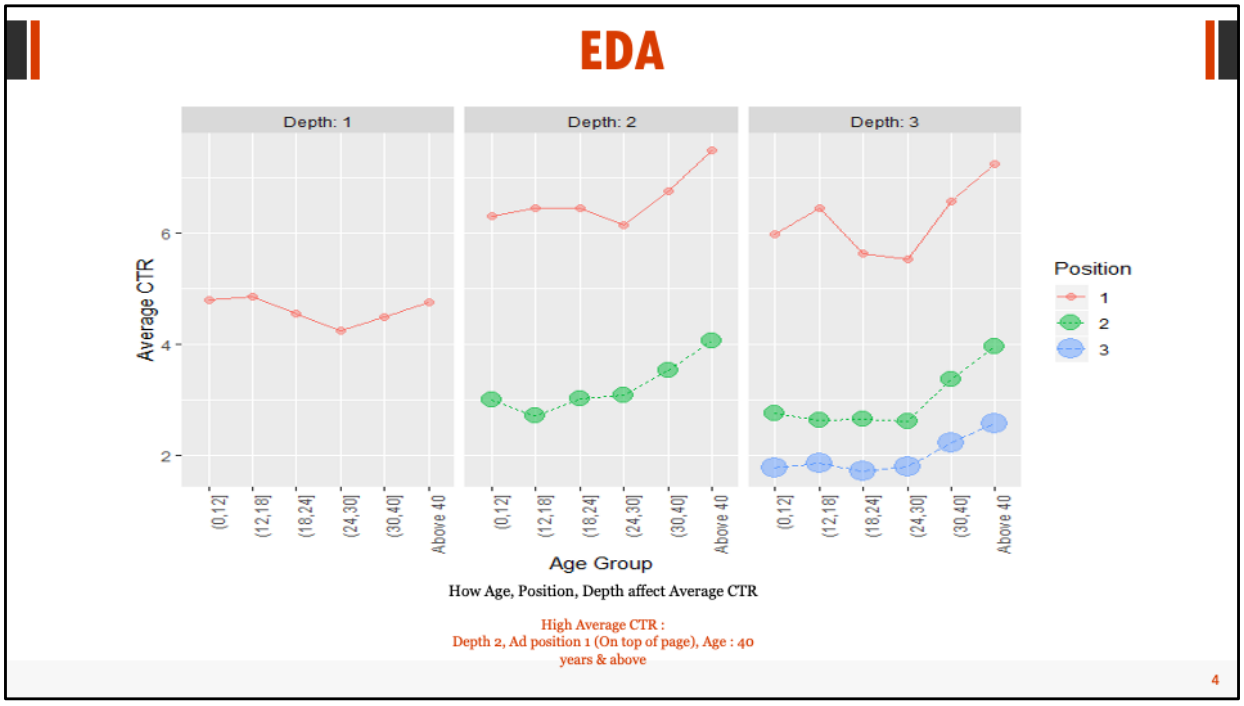
The mathematical objective of our project solely revolves around finding pCTR, which is the probability that a certain ad is clicked while being conditioned on the occurrence of the ad (AdID), user (UserID) and relevant context (P(Click | AdID,UserID,Context)).

Thus, accurately predicting the probability of click (pCTR) of ads is critical for maximizing the revenue and improving user satisfaction.




Each of the 155,750,158 observations in the data identifies an instance of a search session which describes an impressed/displayed ad (AdID) by a user (UserID) who issued a particular search query (QueryID) with a certain number of ad attributes such as the depth (number of ads shown), position, title, landing page, description, keywords and the advertiser of the ad (AdvID).

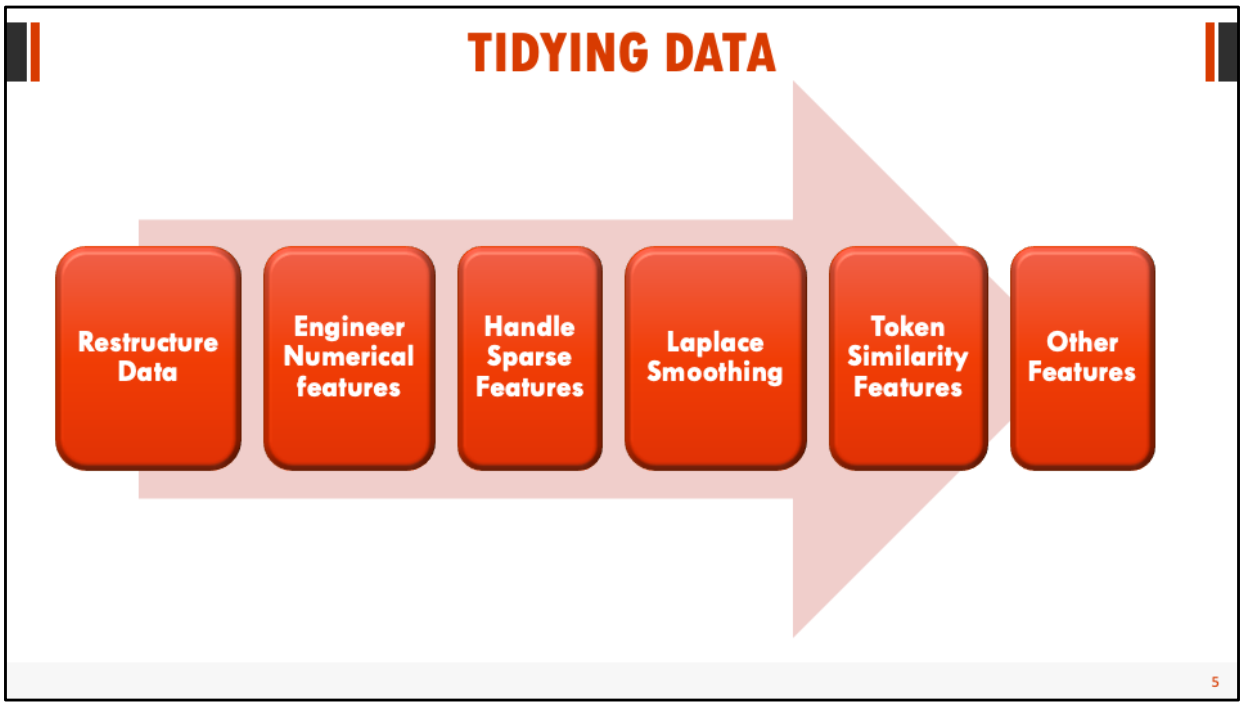
The fields, Impression (number of search sessions in which the ad was shown to the user) and click (number of times the ad has been clicked among the above impressions), are critical for our problem to obtain the target variable CTR by performing their division (#clicks / #impressions). Moreover, this dataset is accompanied by five additional data files, each line of which maps an ID to a list of tokens corresponding to the query, keyword, ad title, ad description and user information (age, gender), respectively.




EDA has been an essential part of the project in deriving value insights of various features and their coherent relationship on the CTR.

Few of the important insights are :

- As observed in the above picture, the 3 features Depth, Position and Age seem to have a correlation with the Average CTR.
    - Users above the age of the 30 years are more likely to click an Ad as compared the younger counterparts.
    - Ad's in position 1 have a higher chance of clicking than the others and also more the Ad's higher the chances of the clicking.
- We also found that females above the age of 30 seem to have an higher chance of clicking an Ad than the males of the same group.
- Advertisers regardless of the CTR used the same median word count in Ad Title and Description.
- High CTR Advertisers have higher mean number of impressions, frequent advertisers and Ads have higher average and median CTR.
- Positive correlation between ID's and impression seem to suggest that ID's may contain time information.

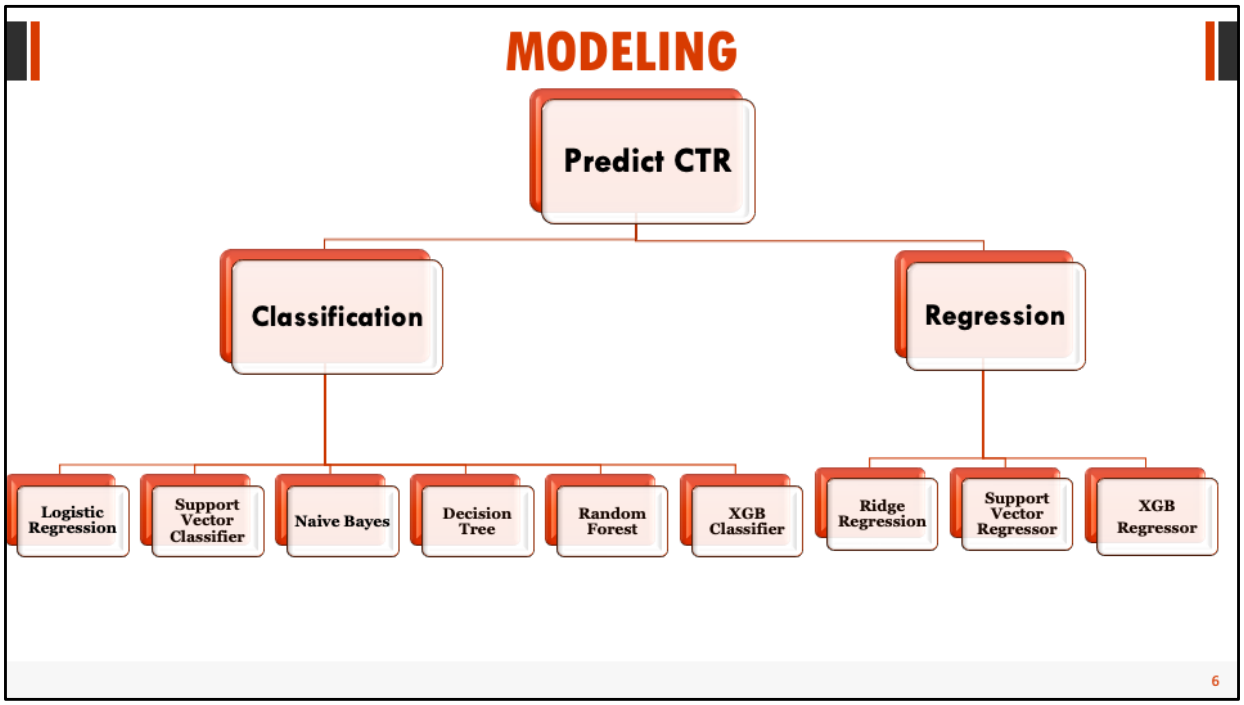



Data restructuring was the crux for modelling the data as a binary classification and regression problem. Impression and Clicks were the important features of the data, which were used to feature engineer as positive samples (#Clicks) and negative samples (#Impression - #Clicks) represented by 1 and 0 respectively. CTR - #Clicks / #Impressions was feature engineered to model as a regression problem.

The data composed largely of categorical variables which were very useful in our model. For each categorical variable we computed the average CTR as an additional one dimensional feature. The categorical variables were to be utilized and the best way was to convert them to binary features. The huge amount of categorical variables led to the generation of a Sparse Matrix.
The binary classes were under represented. In order to deal with the imbalance of the values Additive smoothing technique had to be used over the categorical variables. The dataset contained descriptive data which was utilized by using cosine similarity between TF-IDF vector of the tokens.

Apart from the numerical features in the data we added a few additional such as the number of the tokens in each of the descriptive features as well as the sum of the TF-IDF value's of the tokens for each of the descriptive data.

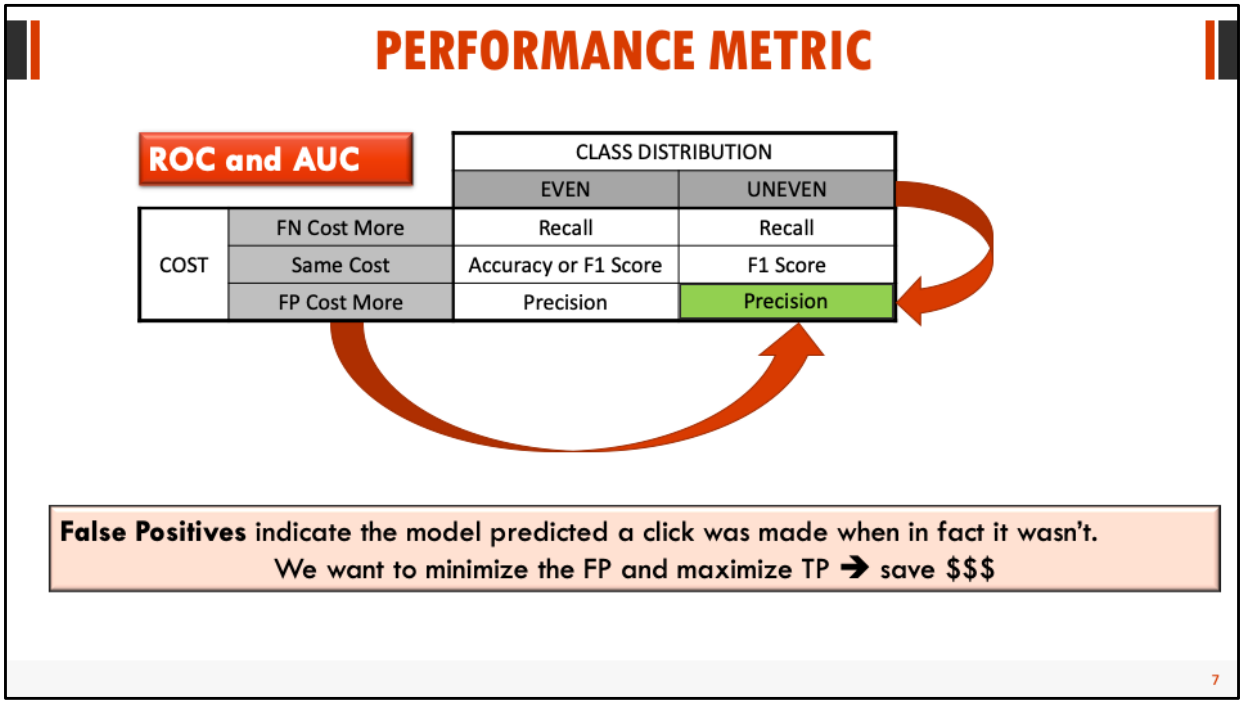
Predicting CTRs for new ads is extremely important and proved very challenging in from our work. We adopted an approach for predicting the CTRs of new ads by using the other ads with known CTRs and the inherent similarity of their keywords. The similarity of the ad keywords establishes the similarity of the semantics or functionality of the ads.




The CTR prediction problem could easily be modeled as a regression task by training the model to use the continuous valued CTR = click/impression as target. We then regressed over the numerical response CTR using various informative features such as the titles, the descriptions and the user profiles among others. On the other hand, to model this as a classification problem, we had to unroll and expand the data corresponding to the number of impressions in the original file. The number of clicks in the original file determine the number of tuples which will carry a 1 in the clicks field. The others were assigned to 0. We used the number of an ad's impressions and the number of the ad's clicks to calculate the click-through rate based on the prediction results of the test data, i.e., for every impression, the user may or may not choose to click an ad, which can be represented as 0 (a user does not click the ad) and 1 (the user clicks the ad). Therefore, the model essentially predicts the probability with which the ad will be clicked by a particular user which is effectively its Click-Through Rate.

Due to the flexibility in modeling this hypothesis as both a regression as well as a classification problem, it bridges gap between diverse set of models based on their underlying assumptions of the data and eventually allowed us to assess the causal nexus of features as well as demarcate interpretable and theoretical models that posits a good performance such as KNN, SVM, Naïve Bayes from those models that tend to perform better on a more practical standpoint at the cost of interpretability like XGBoost, Random Forests among others.

We analyzed models in an effort to better understand the hidden insights or patterns using both Python and R as as some tools tend to have an edge over the others because of more efficient library implementations leading to improved performance while others aim to be faster at the cost of performance.




The goodness of the predictions is evaluated by the area under the ROC curve (AUC), which is equivalent to the probability that a random pair of a positive sample (clicked ad) and a negative one (unclicked ad) is ranked correctly using the predicted click-through rate. That is, an equivalent way of maximizing the AUC is to divide each instance into (#click) of positive samples and (#impression-#click) negative samples, and then minimize the pairwise ranking loss of those samples using the predicted click-through rate. We found AUC to be favorable for evaluating the performance of the hypothesis as our objective is concerned only with the order/ranking of the highest performing ads rather than the real values of their probabilities where metrics like logloss would be a better fit to evaluate with. Consequently, this suffices to evaluating the holistic performance of a classifier without any need to choose best performing thresholds. Additionally, ROC is insensitive to changes in class distribution or skewness. For instance, a dummy classifier that always outputs 0 or 1, may erroneously achieve high accuracy but the AUC value centers around 0.5 suggesting a performance equivalent to a random classifier.

It was an interesting challenge to appropriately evaluate regression using AUC by making use of an efficient but course-grained approach of adopting an algorithm proposed by Tom Fawcett in an effort to calculate AUC from a model that outputs uncalibrated scores, which essentially happens to be the engineered numerical response variable, CTR. Initially, the samples are sorted in the decreasing order of CTR and for each successive sample, the trapezoidal area is computed by cutting it into vertical slices at every change of CTR resulting in an approximate AUC value for a regressed hypothesis with clicks and non-clicks in the place of True Positive rate and False Positive rate.

# FEATURE SELECTION

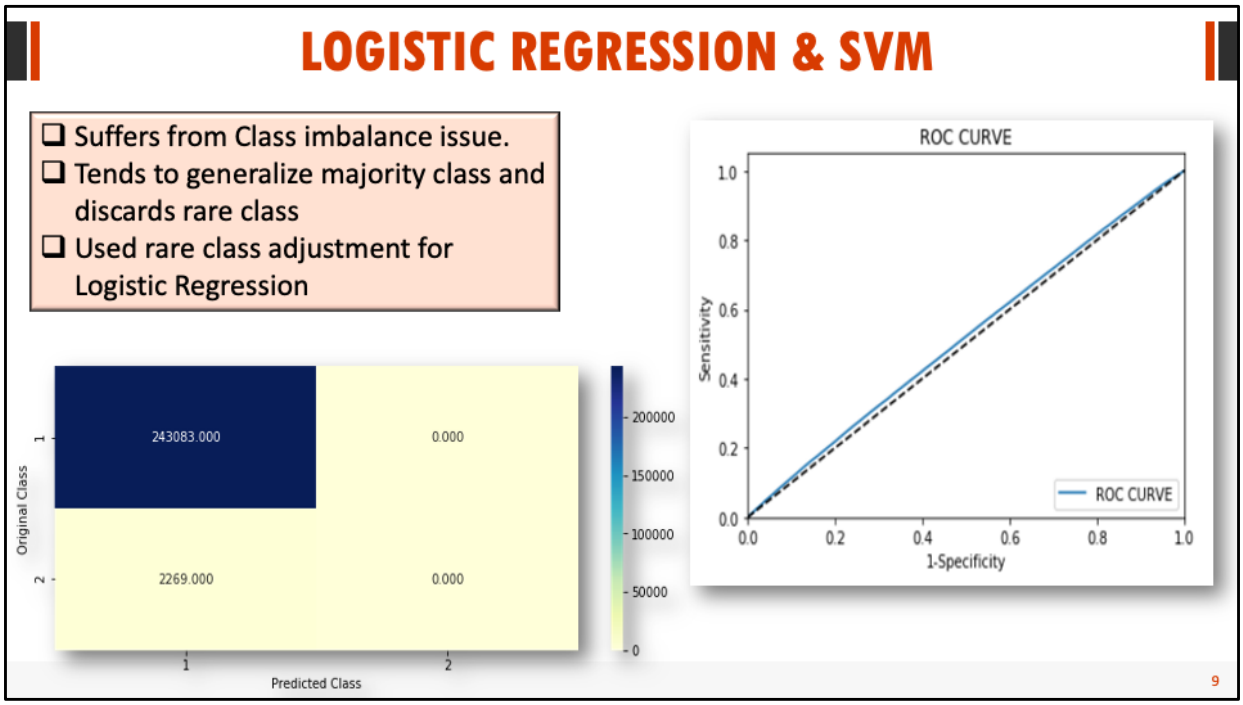
| Features | AUC |
|---|---|
| Sparse | - |
| Add pCTR | ↑ |
| Add aCTR | - |
| Add Group | ↑ |
| Add num_tokens | ↑ |
| Add age, depth, position, rposition | ↑ |
| Add num_impression | ↓ |
| Add num_occurs | ↑ |
| Add sum_idf | - |
| Add tokens_vector | ↑ |
| Add ID_val | - |
| Add mean_idf | - |

Performed Subset selection, forward and backward elimination and PCA to assess the resultant features and their performance on AUC.

8

A wide spectrum of methods were used to reduce the number of features in each model. The inherent relationship between the predictors and the response variable was found to be non-linear which led us to adopt various features selection techniques based on the tacit assumptions of the models. We performed multiple reruns of our evaluation on various subset of features for each individual model in order to delineate the causal inferences between the variables by better accommodating the assumptions the model holds about the data. We began by using fruitful but tedious pre-processing methods such as manually selecting features based on sufficient domain knowledge gleaned from our Literature survey as well as incorporating univariate selection where in we considered features that exceeded a certain threshold of correlation between the feature and the target although it had its own setbacks for only considering the features individually.

We then made use of automated step-wise methods like Forward and Backward selection in trimming out irrelevant features and finally adopted the least interpretable but effective method of PCA to analyze the variables with highest variance by standardizing them when needed. The extensive blend of approaches undertaken contributed to effectively summarize the effects of multiple default/engineered variables on AUC, although the variables are further factored down to accommodate per-model assumptions. For instance, since Naive Bayes exploits a strong independence assumption to efficiently handle a large dataset, we mainly tested on the raw categorical features and used a simple Backward selection method repeatedly to select relevant features.




Initially we considered logistic regression as it works well on large datasets/high dimensions and models our hypothesis by the following a probability distribution P(Click | x,w) using a maximum likelihood problem, where the clicking probability is taken as the ranking criteria. Although, Logistic regression was helpful in promoting interpretability using feature importance or hypothesis testing, it suffered from multicollinearity problem which significantly hurt the interpretability of the weights. To overcome this, we introduced polynomial or interaction features and calibrated the model to account for data imbalance.

These surprises have not adversely affected our workflow. If anything, they allowed us more flexibility and room to brainstorm model assumptions and disregard the use of models such as KNNs and Kernel LR as it is apparent that they perform very poorly on our dataset and are very expensive to train as their runtime complexities prove terrible for this problem. Linear models are also often not that good regarding predictive performance, because the relationships that can be learned are so restricted and usually oversimplify how complex reality is. Consequently, The interpretation of a weight can be unintuitive because it depends on all other features.

Looking at the data, we infer that the data is highly skewed in the direction of no clicks, i.e. Clicked column has more than 85% data with value 0. As only one AD will be clicked for every session, the percentage of Clicked AD is considerably lower leading to over sampled data and very high precision. SMOTE technique was used to solve this problem which balances the data for better attributes. This helped the AUC for Logistic regression to almost jump up to 0.6. The SVM is very expensive to train for multiple iterations to analyze the effect of smoothing imbalance and is therefore dismissed.

## CLASSIFICATION OVERVIEW

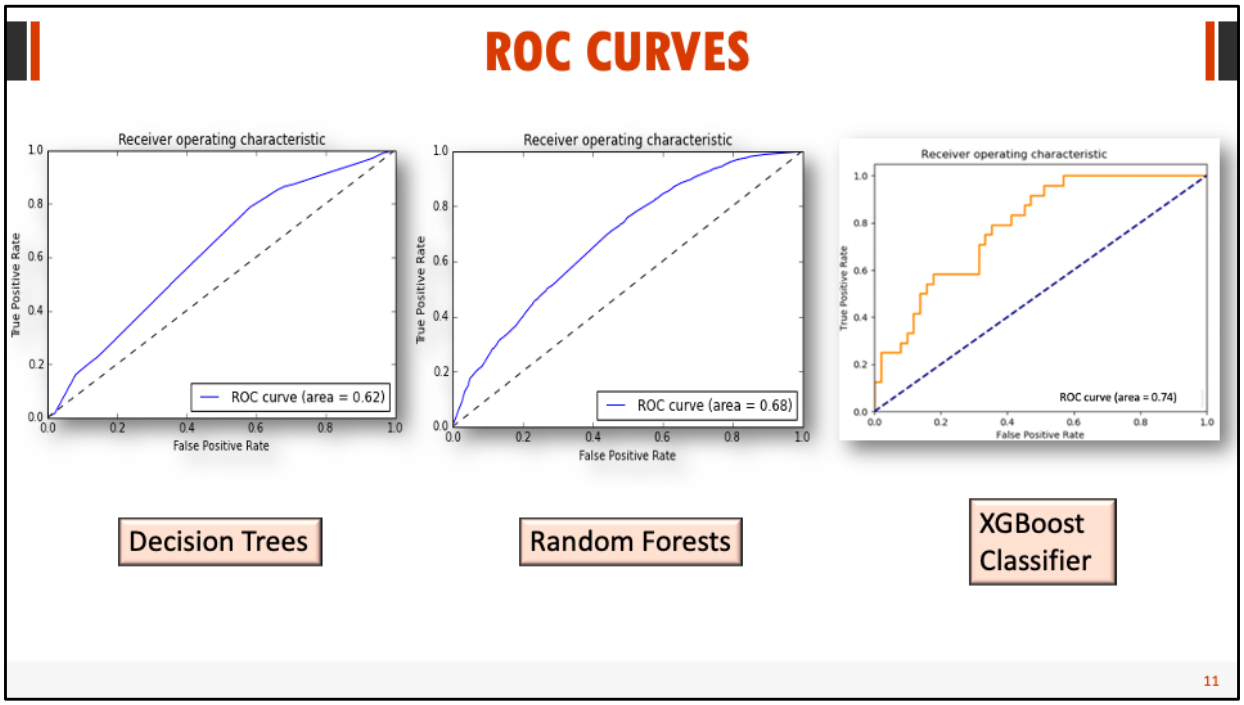
| | LOGISTIC REGRESSION | SVC | NAÏVE BAYES | DECISION TREES | RANDOM FOREST | XGBOOST CLASSIFIER |
|---|---|---|---|---|---|---|
| TRAIN AUC | 0.628 | 0.691 | 0.615 | 0.694 | 0.984 | 0.784 |
| TEST AUC | 0.594 | 0.503 | 0.599 | 0.62 | 0.680 | 0.747 |
| TEST RMSC | 0.061 | - | - | 0.091 | 0.062 | 0.055 |
| TEST WRMSC | 0.160 | - | - | 0.193 | 0.168 | 0.152 |
| PRECISON | 0.57 | 0.17 | - | 0.71 | 0.78 | 0.82 |
| RECALL | 0.69 | 0.81 | - | 0.66 | 0.57 | 0.51 |

10

Another model we eyeball at particularly is Decision trees which promote low latency with their shallow trees and hence prove useful with their low run-time complexity. We realized that with GBDTs, we would have the flexibility with any loss function - so we can opt to use it with performance metrics like pseudo residual log loss or AUC that we're optimizing. Moreover, we made use of several interaction features to account for the polynomial nature of the decision boundary inherently with GBDTs. At each step of building individual tree we find the best split of data. While building a tree we use not the whole dataset, but just the bootstrap sample. We aggregate the individual tree outputs by averaging them.

Random Forests are similar to Decision Tress except that for Random Forest we use Bootstrap Aggregation. It has no model underneath, and the only assumption that it relies is that sampling is representative. It performs reasonably well with an AUC of 0.68.

XGBoost is a good starting point if the classes are not skewed too much, because it internally takes care that the bags it trains on are not imbalanced. But then again, the data is resampled internally. Therefore, we have shown that with simple feature engineering and a good training algorithm it is possible to achieve high precision CTR prediction.




Logistic regression struggled with its restrictive expressiveness as interactions must be added manually. It's interpretation proved more difficult because the interpretation of the weights is multiplicative and not additive.

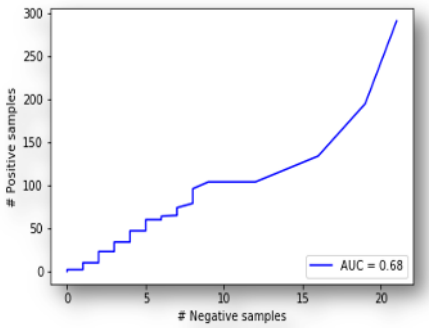
The tree structure is ideal for capturing interactions between features in the data which led to a reasonable AUC of 0.62. However, Trees failed to deal with linear relationships as any linear relationship between an input feature and the target has to be approximated by splits, creating a step function which is very inefficient. We also observed a lack of smoothness in performance. Slight changes in the input feature had a big impact on the predicted outcome which is not desirable. Furthermore, Random forests provided a improved AUC of 0.68 but are also quite unstable as a few changes in the training dataset could create a completely different set of trees. Additionally, decision trees are very interpretable as long as there are only a handful of features, unlike our data where we have over 80 different variables.

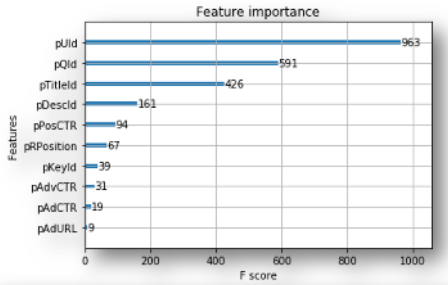
We anticipated that Naive Bayes could be an interpretable model on the modular level because of the independence assumption. It is found to be very clear for each feature how much it contributes towards a Click or no click by interpreting their conditional probability.

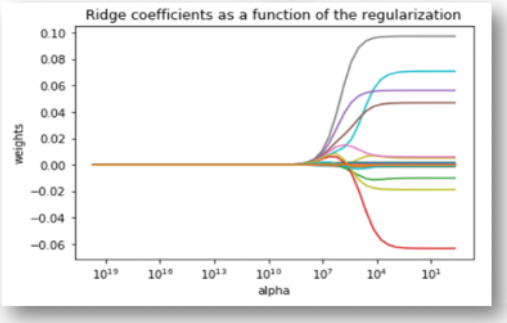
Due to in-built L1(Lasso Regression) and L2 (Ridge Regression) regularization, efficiently handling missing values, and effective tree pruning, XGBoost Classifier outperformed other models with an AUC of 0.74 with 2 depth and 800 estimators.

# REGRESSION OVERVIEW

| | RIDGE REGRESSION | XGBOOST REGRESSOR |
|---|---|---|
| TRAIN AUC | 0.6102 | 0.7312 |
| TEST AUC | 0.5099 | 0.6806 |



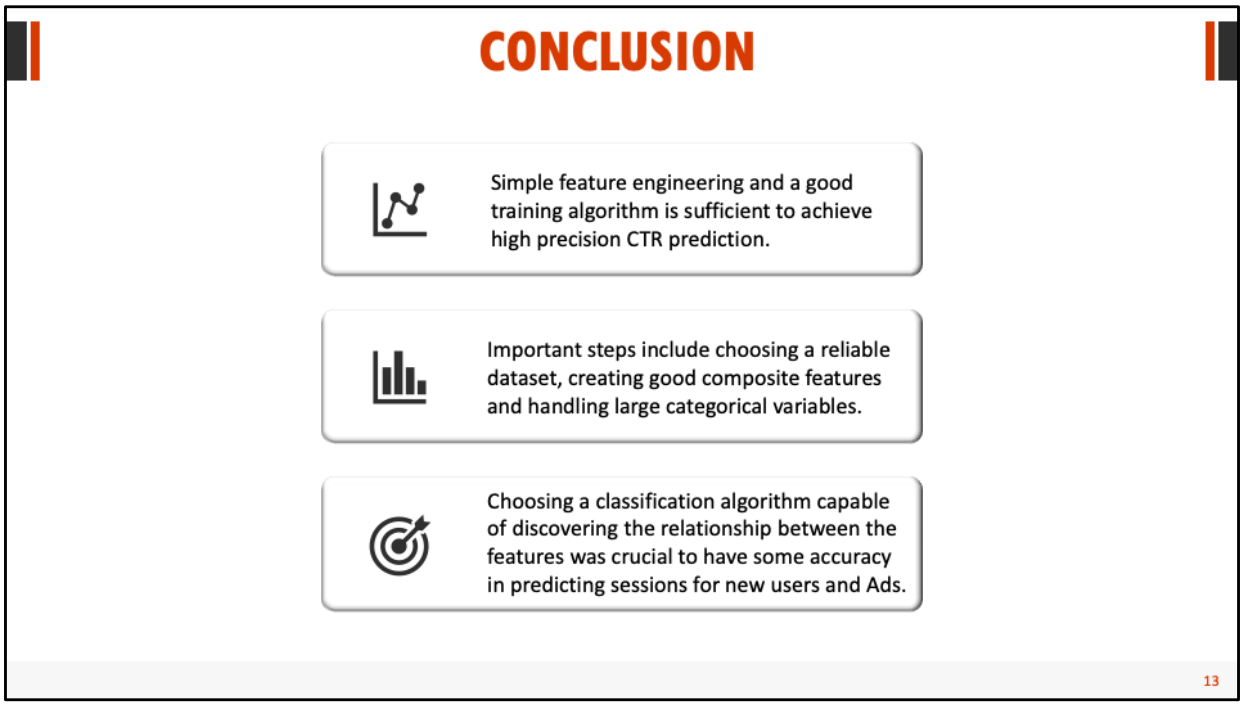




12

Ridge regression can be really useful if you have a huge number of columns and rows. It performs well when there is multicollinearity in the data. A normal linear regression essentially cannot distinguish between different regression functions in this situation. Regularization provides a method for picking an appropriate function in order to improve the AUC score and minimize the RMSE.

The assumption that is made before applying the model is that the data is linear and has a constant variance. The important part of the Ridge regression is the estimation of the hyperparameter alpha which we estimated by using Cross Validation over the training set.
We observed that our regression model has low RMSE on the test data but the R^2 value is negative number which meant that the best fit line was performing worse than the horizontal line. This led us to conclusion that that the data wasn't linear and other non linear methods have to enhance our results. The AUC value for the model was around 0.5 .




We managed to get a good grip over the data and obtained a sufficiently small subset of it for local implementation. Some machine learning techniques including ridge regression, logistic regression, Decision trees, random forests and XGBoost among others were implemented in the smaller data set using R and Python. We have obtained a best AUC score of 0.74 on test set using XGBoost Classifier.

As the data size is quite large, our basic intuition is to try different methods to better model the user click behaviors. We proposed to use four individual models and each of them performs well on the prediction task. We believe that complex algorithms namely RankNet models and Collaborative Filtering perform well on this dataset. We also hypothesize that Blending the diverse models is the key for attaining a momentous performance. Efficient methods for ensembling using Neural Networks and random forest will be looked at further down the line. The complete dataset will be used with Vowpal Wabbit as the tool as it is efficient to handle a large dataset on a single machine.

We model users' behavior on click-through rate by using linear and non-linear models and aggregate all of them. For each single model, we provide various kinds of features combination to capture users' behavior from different perspective with a goal of achieving reasonable performance. We believe the success of our methods is based on capturing various information in the data and utilizing those information effectively.