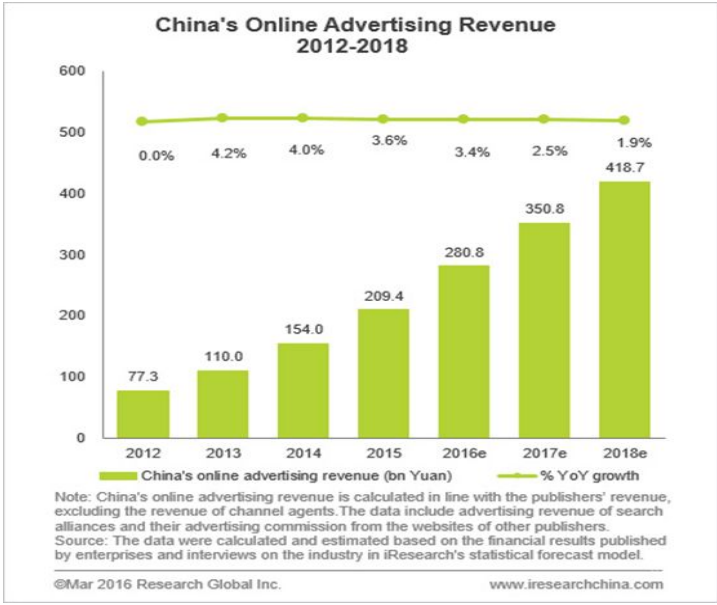
DSO530 Final Report on TalkingData's AdTracking Fraud Detection

Team Members:
Yi-Chun Chi- 9751080834
Mahalakshmi Raghavan- 7928478072
Ushita Palande- 7724267639
Yu Luo- 2919525948
Riddhi Malviya- 8151412391
Contact person:
Yi-Chun Chi chiyichu@usc.edu

Section 1: Business Problem

China has a very large online advertising market. With the steady YoY growth of this industry, they also face several issues like fraudulent clicks.



TalkingData, a big data service platform, which deals with more than 3 billion clicks per day, reported that around 90% of the clicks were potentially fraudulent. By definition, a fraudulent click is when a user (person/script/computer program) clicks a link but never ends up installing your app or buying the product.This results in affecting the probable revenue that could be earned by online advertising for a company.

Our aim for this project was to build a model such that will ensure a quick classification of genuine clicks (users that do end up downloading the app) from the fraudulent clicks.

Section 2: Data Description

The original dataset was from the "TalkingData Challenge" on Kaggle website. It had 1.5 million observations (rows) and columns like IP, App, device, OS etc. Feature engineering was performed on this dataset and the final dataset provided had 1.5 million rows and 74 features (columns). This entire dataset was used in the project to run the models and come up with solutions. 1.5 million rows were divided into training (1 million) and testing (0.5 million) for performing statistical methods.

Section 3: Feature Selection:

From the set of engineered features, we decided to proceed with reduction in the features set. By choosing a subset of features, we can better accuracy whilst requiring less data. Fewer attributes is desirable because it reduces the complexity of the model, and a simpler model is simpler to understand.

We performed Kolmogorov Smirnov (KS) for feature selection. The KS Method provides a cumulative distributions for a variable explaining the goods and the bads. The greater the distance between the two

distributions, implies that it does a good job in explaining the goods from the bads. Hence variables with the highest KS scores are definitely included in the model.

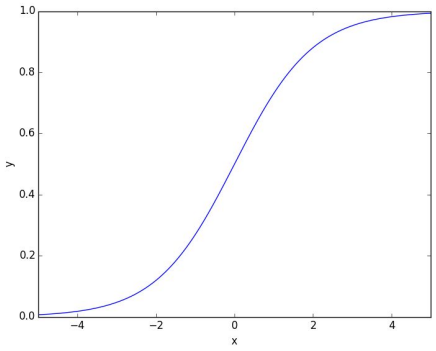
After getting a set of variables, we wanted to further reduce the number, to improve the computation capability. We performed Variable Selection using Random Forest. It is possible to get the importance of each variable using Random Forest. Variables with the highest importance were included in the model, and we finally used a set of 20 variables to run all the models. The variables are as follows:
appAttrib,channelAttrib,ipAttrib,app60s,app30s,channel60s,app10s,appOs10s,channel30s,os60s,app3s,os30s,os10s,ip60s,osIp10s,deviceIp1s,appChannel10s,channel10s,channelDevice10s,appIp10s

Section 4: Statistical Methods

After feature selection, we moved on to trying the models. We started by looking at a scatter plot of the data, in terms of PC1 and PC2 (which explain the highest variance of the data), it was clear that the classes are imbalanced, there were many more 0s (denoted by the black dots) compared to 1's (denoted by blue dots). Our final goal here is prediction. The meaningful metrics to evaluate models will be the area under the roc curve. As the ROC auc score moves closer to 1, our model is closer to predicting no false positives.
Receiver Operating Characteristic area under curve (ROC AUC)
Precision Recall area under curve (PR AUC) – does not consider true negatives, thus unaffected by Class imbalance

4.1: Logistic

Logistic Regression models the conditional probability P(Y = 1|X) using the sigmoid function $f(x) = \frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}}$. Since the sigmoid function f(x) takes values between 0 and 1, it's perfect for modeling probability.



Representation of a Logistic Model

We got an AUC of 94.91% for this model. Since Logistic Regression is often regarded as a baseline model for classification, we would like to try other models and compare the result.

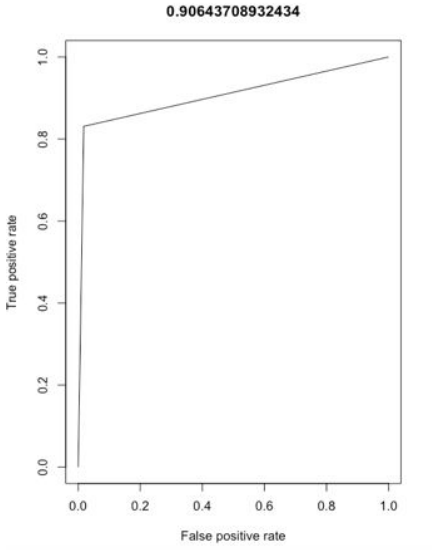
4.2: QDA

Quadratic discriminant analysis (QDA) assumes that an observation from the kth class is of the form X ~ N(μk, Σ k), where Σ k is a covariance matrix for the kth class. QDA classifier would assign x to class k for the largest discriminant function $\delta_k(x)$, which contains a quadratic function.

We performed Leave-one-out cross validation (LOOCV) on the result of PCA to tune number of PCs to put into the model. We got an AUC of 94.69% for this model, which is not as good as the result of the Linear discriminant analysis (LDA) model. We suspected that this is probably because the decision boundary of the dataset is more close to linear.

4.3: KNN

To perform knn and predict the class for an observation, a model usually finds the class labels of k nearest neighbors of the observations, and assigns the majority class label to the observation. The problem with knn was that since the class label 0's are much higher than 1's, based on majority voting, the model will never predict 1. thus, we did some tuning in the 3rd step of knn so that if any 1 of the nearest neighbor is 1, the observation is assigned to 1 because we didn't want to miss the clicks which actually downloaded the app.Only performed on PC's, so that variables are normalized.Otherwise, because of how KNN works, the Euclidean distance of some variables might have higher weightage based on their numerical value.



	Predicted 1	Predicted 0
True 1	TP 1524	FN 310
True 0	FP 9015	TN 489151

This model was not chosen because although the ROC and TPR are good enough at 90 and 83%, precision is very low, which means out of the predicted 1s, only 15% are actually 1s.

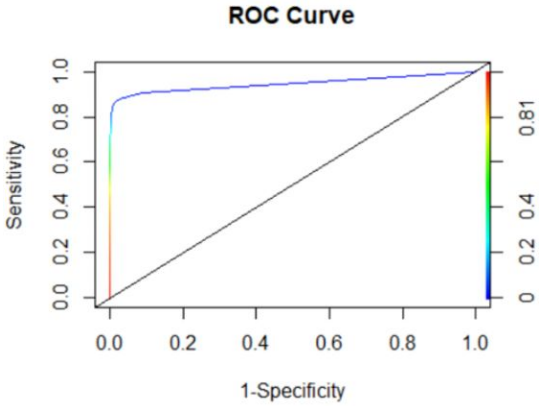
4.4: SVM

Support vector machines are supervised learning models used for classification and regression analysis. Given a set of training examples, each observation labeled as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces. In this analysis, we used our training data to train an SVM with y as labels, cost equal to 100 and gamma as 0.001. We then used our testing and out-of-time datasets to test the efficiency of our model. This model had a very high computational time so we did not use it.

4.5: Random Forest

The Random Forest is an Ensemble of decision trees. It is essentially a bootstrapping algorithm used with a Decision Tree model. This model chooses the best feature among a random subset of features instead of choosing the best feature at every node. The AUC obtained was 92.12%. This model was not chosen because it did not misclassify the bad clicks as well as we would expect it to.

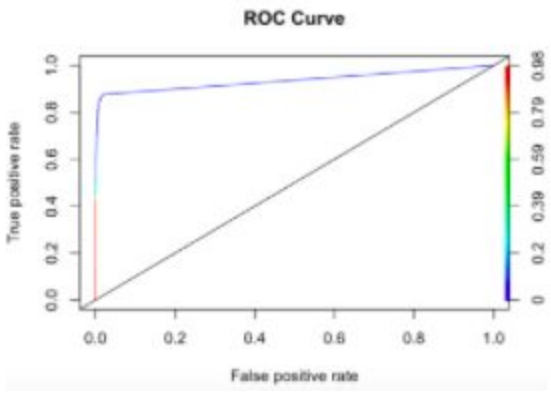
	Predicted 1	Predicted 0
True 1	TP 1088	FN 746
True 0	FP 191	TN 497975



4.6: Boosted tree

In boosting, trees are grown sequentially: each tree is grown using information from previously grown trees. Boosting does not involve bootstrap sampling; instead each tree is fit on a modified version of the original data set. We include all the numeric variables, and the relative influence table shows that the top important features are appAttribe, channelAttrib, ipAttrib, etc. For boosting tree model, we got AUC score of 93.59%, precision of 79.98% and true positive rate of 44.66%,which is not as good as the result of Random Forest. So we did not choose this model.

	Predicted 1	Predicted 0
True 1	TP 819	FN 1015
True 0	FP 205	TN 49796



Section 5: Final Approach

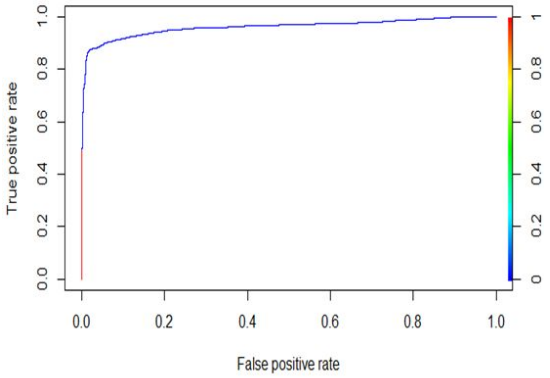
5.1: Summary

We chose **Linear discriminant analysis (LDA)** as our final approach. LDA models the conditional distribution of X given Y by normal distributions. LDA classifier would assign x to class k for the largest discriminant function $\delta_k(x)$. We performed LDA on variables obtained from KS. Since variables with higher KS scores would do a better job in explaining the goods from the bads, those variables with the highest KS scores are definitely included in the model. We used Leave-one-out cross validation (LOOCV) to tune number of variables to put into model. Each time we included one variable with the

highest KS score into our model and calculated the Area under the ROC curve (AUC) of the CV result. After several iterations, we could compare the AUC of different models that have different numbers of variables.

It turned out that the model with 22 variables that have the highest KS score gave us **the highest AUC**, so this model was chosen to be our final model.

5.2: Results



	Predicted 1	Predicted 0
True 1	TP 907	FN 927
True 0	FP 547	TN 497961

The **Area Under ROC Curve (AUC)** of our final model is **96.07%**, which is the highest among all the methods we tried. The Bayes classifier $(P(Y = 1|X = x) \geq 1/2)$ gave us the minimum **classification error** of **0.71%**. The confusion matrix of the Bayes classifier shows that the **True Positive Rate** (proportion of positives that are correctly identified) is **49.45%** $(\frac{907}{907+927})$, **True Negative Rate** (proportion of negatives that are correctly identified) is **99.89%** $(\frac{497961}{547+497961})$, **Precision** (a.k.a. Positive Predictive Value**,** proportions of positive results that are true positive results) is **62.38%** $(\frac{907}{907+547})$, and **Negative Predictive Value** (proportions of negative results that are true negative results) is **99.81%** $(\frac{497961}{927+497961})$.

	Predicted 1	Predicted 0
True 1	TP 907	FN 927
True 0	FP 547	TN 497961

	Predicted 1	Predicted 0
True 1	TP 1474	FN 360
True 0	FP 4984	TN 493182

The left table is the confusion matrix of the Bayes classifier that gave us the highest classification accuracy. In the real business world, for those Predicted 0's, we think they are those who produce lots of clicks, but never end up installing apps. Therefore, we might want to put them into our IP blacklist and device blacklist. However, if we do so, we would also block more than half of the people that will download an app after clicking a mobile app ad. Next time, when they want to download an app by clicking a mobile app ad, they won't be able to do that. This is bad for the advertisers because those people are exact their target audiences. Moreover, this is bad for TalkingData's reputation.

To solve this problem, we can lower the threshold in our model. The right table is the confusion matrix of the classifier with lower threshold. With this threshold, we got a classification error of 1.07%, which is of course slightly higher than the result of the Bayes classifier. However, our True Positive Rate is increased to 80.37%, which means we only misclassify less than 20% good guys!

By lowering our threshold, the number of observations that we classified as positive would increase. As a result, the number of False Positives would also increase.

NAME	AUC	PRECISION	TPR	Features
Logistic Regression	94.91%	9.97%	87.79%	KS, 10-fold CV
QDA	94.69%	19.96%	84.95%	PCA, LOOCV
KNN	90%	14.45%	83.1%	PCA
Random Forest	95%	85.06%	59.32%	KS, RF
Boosted Tree	93.59%	79.98%	44.66%	All numeric
LDA	96.07%	22.82%	80.37%	KS, LOOCV

5.3: Business Interpretation

- Developing better marketing strategies: After figuring out the customers that actually end up downloading after the click, their characteristics can be clustered. Segments of customers with similar characteristics can be targeted with specific marketing strategies.
- Trade-off: When the hosting company identifies such fraudulent clicks blocks their IP, they in turn lose money that they would otherwise gain from their client. However, when they detect such fraud, they gain the goodwill of their clients which is a good tradeoff.
- Change of Business model: From having the cost-per-click (CPC) and cost-per-impression (CPM) business model, they should change it to cost-per-action(CPA). That means that the client company does not pay the hosting company for every click but for every time some action was taken after the click (downloading the app). A lot of companies have already implemented this strategy.
- Algorithm of the bot:Many companies install automatic bots to increase their click rate. However, once such bots are detected, their algorithm can be understood from the pattern trace they leave behind.