CKME136 - Capstone Project
Project Report and Final Results

TalkingData AdTracking Fraud Detection Challenge
https://www.kaggle.com/c/talkingdata-adtracking-fraud-detection

Phi Huynh - 500777278

https://github.com/PHLHY/Capstone-

Introduction

Click fraud is the act of generating fraudulent clicks on pay-per-click advertisements which results in artificially inflating web traffic and potential loss of revenue to the advertiser.

TalkingData provided a dataset of around 200 million clicks spanning over 4 days for a Kaggle challenge with the aim of building an algorithm for determination of users that will download a mobile application after clicking on a relevant advertisement. This challenge will be the main focus of the research question for this capstone project.

Techniques for classification analysis will be employed on the R platform (RStudio) to tackle this challenge.

Literature Review

A review of research articles and publications on the different areas of click fraud, important features for consideration, and classification algorithms is summarized below to give a comprehensive approach to the research question for this capstone project.

Click Fraud[5]
This article provides a better understanding of what click fraud is and its impact and implications. This is to aid with the terminologies and give background information so that it provides better insight to complete this capstone project. The author explains what click fraud is, the different types of click fraud, ways to detect click fraud, and potential solutions to combat click fraud.

Combating Online Fraud Attacks in Mobile Based Advertisement[2]
This article goes into further details about click fraud in a mobile environment which is relevant to this capstone project. It provided further knowledge on vulnerabilities of click fraud based on advertisement networks and provided suggestions on ways to combat click fraud for the mobile environment.

Classification and Regression Tree[6]
The author explained differences between classification and regression trees along with different algorithms on a car dataset. This would help me select and consider the kinds of algorithms that can help with modeling the data for the capstone project. The author, Loh, found that the GUIDE algorithm has the highest predictive accuracy.

Feature Engineering for Click Fraud Detections[8]
Phua et al. defined feature engineering as “extraction and selection of features” as a key component to click fraud detection to maximize performances of models. While Phua’s team was working on banking data, they found that a relatively large number of clicks, rapid duplicate clicks, or a high percentage of clicks from high-risk countries as important fraud indicators. They

used classification techniques such as Gradient Boosted Regression Models (GBM), Random Forest in R and RIPPER from WEKA as their algorithms for click fraud detection.

Detecting Click Fraud in Online Advertisement: A Data Mining Approach[7]
Oentaryo et al. analyze data mining approaches from different competitors in a competition on real-world fraud data for online advertisements provided by BuzzCity Pte. Ltd. The authors summarized the approaches done from the top three winners, the runner-up and from the organizer. Those include how each team did their pre-processing/feature extractions, their methods, which classification algorithms they used, and the performances of their models.

Learning From Automatically Labeled Data: Case Study on Click Fraud Prevention[1]
The main focus of classification algorithms is to enhance the model's performance, however, Berrar advised that when doing supervised learning that we should be wary of automatic class labels influencing our results. If there is a discrepancy between our models and the testing model, then we need to check the original result and consider how the dataset originally decide whether a click was fraudulent as it most likely came from a fraud detection algorithm. What is interesting to note is that Berrar used exclusively an ensemble approach of Random Forests for the algorithm as the author found better model performance based on the author earlier studies.

An Ensemble Learning Based Approach for Impression Fraud Detection in Mobile Advertisement[4]
Haider et al. explained their approach on European datasets based on mobile advertisements in fraud detection. They were able to make their model's performance at 99.32% accuracy, 96.29% precision, and 84.75% for recall. For algorithms, the authors examined Decision Tree Classifier J48, Random Forest, and REPTree along with ensemble learning techniques (bagging and boosting).

An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization[3]
The usage of the ensemble approach for classification algorithms has been a theme in the previous articles. A further understanding of these methods will be helpful for consideration of use for the capstone project. This article explains the differences between bagging, boosting and randomization on the Decision Tree algorithm C4.5. Depending on if there is classification noise (incorrect class labels), the bagging method is more proper to use compared to boosting which usually provides the best result in low noise.

Dataset

The datasets used for this capstone project are from a Kaggle competition, “TalkingData AdTracking Fraud Detection Challenge” which is found from this link: “*https://www.kaggle.com/c/talkingdata-adtracking-fraud-detection”*.

Training and testing datasets are provided by TalkingData for this competition. In total, there are over 200 million clicks that were captured over a 4 days span. The training set includes 184903890 data points with 8 attributes while the testing set includes 18790469 data points with 7 attributes.

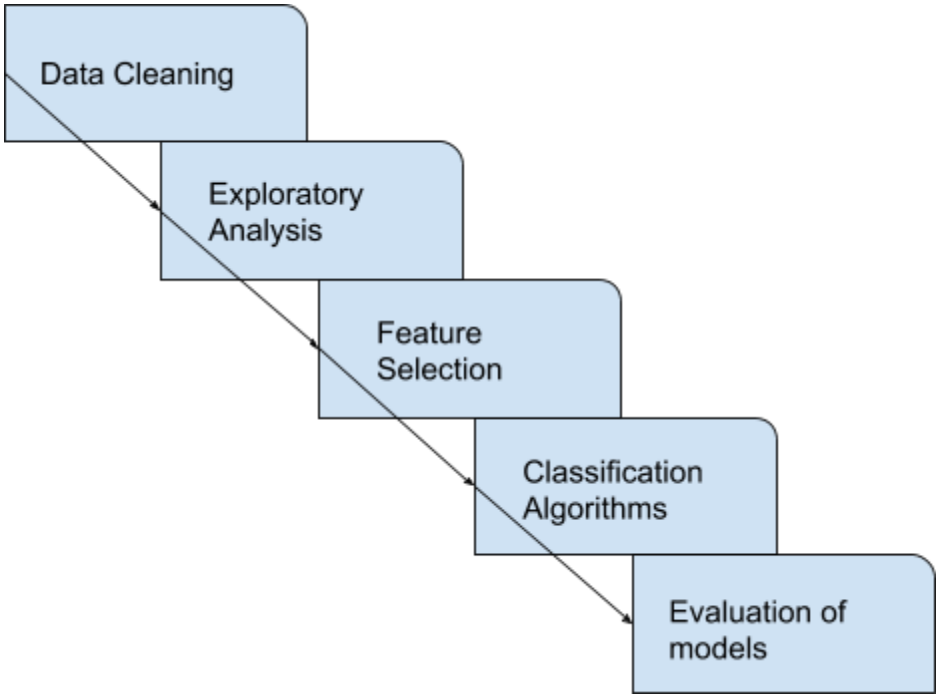
A direct copy of the description of the attributes were provided by TalkingData which is found on this link:
https://www.kaggle.com/c/talkingdata-adtracking-fraud-detection/data

- ip: ip address of click.
- app: app id for marketing.
- device: device type id of user mobile phone (e.g., iphone 6 plus, iphone 7, huawei mate 7, etc.)
- os: os version id of user mobile phone
- channel: channel id of mobile ad publisher
- click_time: timestamp of click (UTC)
- attributed_time: if user download the app for after clicking an ad, this is the time of the app download
- is_attributed: the target that is to be predicted, indicating the app was downloaded

The test data is similar, with the following differences:

- click_id: reference for making predictions
- is_attributed: not included

Approach



Step 1: Data cleaning

Both the train and test datasets that were provided by TalkingData were loaded into the R platform. The test set was not used at this time due to the target variable, “is_attributed” being missing for the training of the models. 184903890 observations were found in the train set with the 8 attributes (“ip”, “app”, “device”, “os”, “channel”, “click_time”, “attributed_time”, and “is_attributed”) present. A quick look into the data using head, tail, and str was also done to provide a better appreciation of what the data looks like.

```
> head(train)
       ip app device os channel         click_time attributed_time is_attributed
1:  83230   3      1 13     379 2017-11-06 14:32:21                           0
2:  17357   3      1 19     379 2017-11-06 14:33:34                           0
3:  35810   3      1 13     379 2017-11-06 14:34:12                           0
4:  45745  14      1 13     478 2017-11-06 14:34:52                           0
5: 161007   3      1 13     379 2017-11-06 14:35:08                           0
6:  18787   3      1 16     379 2017-11-06 14:36:26                           0
> tail(train)
       ip app device os channel         click_time attributed_time is_attributed
1: 100212   1      2 13     125 2017-11-09 16:00:00                           0
2: 121312  12      1 10     340 2017-11-09 16:00:00                           0
3:  46894   3      1 19     211 2017-11-09 16:00:00                           0
4: 320126   1      1 13     274 2017-11-09 16:00:00                           0
5: 189286  12      1 37     259 2017-11-09 16:00:00                           0
6: 106485  11      1 19     137 2017-11-09 16:00:00                           0
> str(train)
Classes ‘data.table’ and 'data.frame':  184903890 obs. of  8 variables:
 $ ip            : int  83230 17357 35810 45745 161007 18787 103022 114221 165970 74544 ...
 $ app           : int  3 3 3 14 3 3 3 3 3 64 ...
 $ device        : int  1 1 1 1 1 1 1 1 1 1 ...
 $ os            : int  13 19 13 13 13 16 23 19 13 22 ...
 $ channel       : int  379 379 379 478 379 379 379 379 379 459 ...
 $ click_time    : chr  "2017-11-06 14:32:21" "2017-11-06 14:33:34" "2017-11-06 14:34:12" "201
7-11-06 14:34:52" ...
 $ attributed_time: chr  "" "" "" "" ...
 $ is_attributed  : int  0 0 0 0 0 0 0 0 0 0 ...
 - attr(*, ".internal.selfref")=<externalptr>
```

Missing values were also checked in both the train and test datasets. No missing values were noted as seen below.

```
> colSums(is.na(test))
  click_id         ip        app     device         os    channel click_time
         0          0          0          0          0          0          0
> colSums(is.na(train))
            ip             app          device              os         channel
             0               0               0               0               0
    click_time attributed_time   is_attributed
             0               0               0
```

A look into blank values was also done. Only “attributed_time” had blank values with 184447044 observations. This corresponds to the same observations in our 0 - value target variable, “is_attributed” which is not the target observation that we are looking for. A closer look at this attribute will be done later in this report.

```
> colSums(train=="")
            ip             app          device              os         channel      click_time
             0               0               0               0               0               0
attributed_time   is_attributed
     184447044               0
> table(train$is_attributed)

        0         1
184447044    456846
```

Since the train dataset has so many observations, after some attempts of modelling and transforming the data, the dataset was ultimately reduced to 25000 observations for faster computations and to prevent any memory issues. This will be added to the limitations section at the end of this report for future consideration. A seed was also set to make sure randomization can be replicated again in the future for this reduction. Both the original train dataset and the reduced dataset were checked to ensure they have equal distribution of the target variable, “is_attributed” (0.00025% in the original dataset and 0.00024% in the reduced dataset).

```
> table(train$is_attributed)

        0         1
184447044    456846
> table(reduced.set$is_attributed)

    0     1
24940    60
```

Unique values for each attributes were looked at based on the reduced dataset. An attempt was made to change these attributes into categorical data type, however, was ultimately left as integer data type due to computational limitations and also due to loss of valuable information with the reduction in the categorical levels. Only the target variable, “is_attributed” was changed

to a categorical data type prior to the modelling process. Again, this limitation will be added to the limitations section at the end of this report.

```
> sapply(reduced.set, function(x) length(unique(x)))
          ip          app       device           os      channel is_attributed
       15046          115           40           98          150             2
        days         hour
           4           24
```

A closer look at the “click_time” attribute was accounted for and it was determined that this particular attribute can be further split into a date and time format. Since the data was only collected over a 4 days span, the assumption was made that the “year” and “month” can be removed. From the example below, all 25000 observation of the reduced dataset had the same value for “month” (11) and “year” (2017).

```
> table(reduced.set$year)

 2017
25000
> table(reduced.set$month)

   11
25000
```

For further processing, the newly transformed “days” attribute were changed to an integer data type so that each day corresponded to a number (ie. Monday -> 1, Tuesday -> 2, Wednesday -> 3, and Thursday -> 4). Only 4 days were considered as the data collected by TalkingData only spanned over this amount. This transformation is to help with further computations for this capstone project.

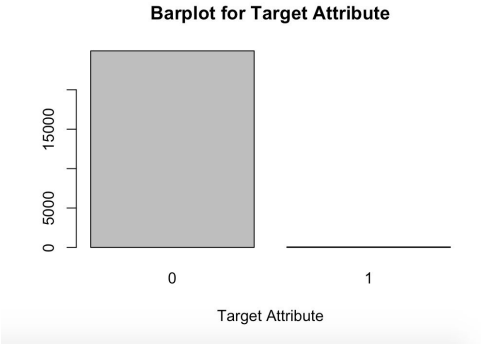
```
> table(reduced.set$days)

   1    2    3    4
1331 7993 8462 7214
```

In addition, for the sake of the challenge which is to determine users who will download an application after clicking on an ad, “attributed_time” was removed as it provided redundant information. Meaning that “attributed_time” only existed if there is a target variable in “is_attributed”. This was seen earlier in this report when looking at blank values where the there are blank values when the target variable, “is_attributed” value was at 0. More information regarding this “attributed_time” for future consideration can be found in the limitation sections at the end of this report.

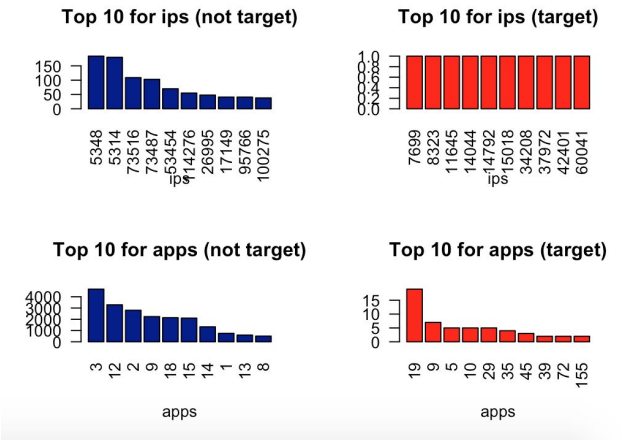
Step 2: Exploratory analysis

Exploring and visualization would provide more information into the data and to reveal important patterns. A barplot was done to look at the distribution of the target attribute, “is_attributed” in the reduced dataset. It is shown that there is a major class imbalance for this attribute as seen in the barplot below. The exact distribution was shown earlier where there are only 60 observations which are the target variable that we are looking for and the rest of the 24940 observations not being the target variable we are looking for. Consideration for balancing the data for modelling was made at this time due to this major class imbalance.

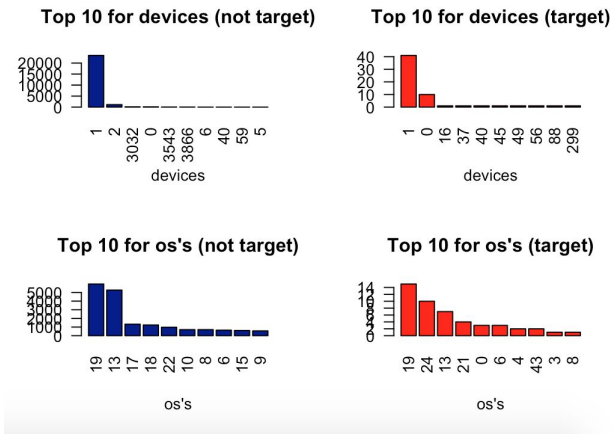


Next, the reduced dataset was split into two different datasets for the purpose of visualization where one of the dataset contains, “is_attributed” being equal to 1 and another where “is_attributed” is equal to 0. Barplots and top frequencies were done with all the attributes to look at how the data differs from between the target and the non-target variable. The blue graphs would represent data where “is_attributed” is equal to 0 and the red graphs would be for data where “is_attributed” is equal to 1, which is our target variable that we are looking for.

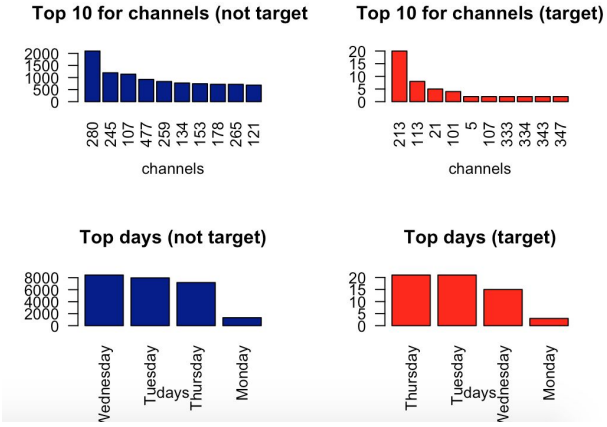
For the “ip” attribute, differences were found between the blue and the red graphs which shows that “ip” should be an important variable for consideration in our analysis. For the “app” attribute, only the value 9 overlaps between the two graphs indicating that “app” could also be an important attribute as well.



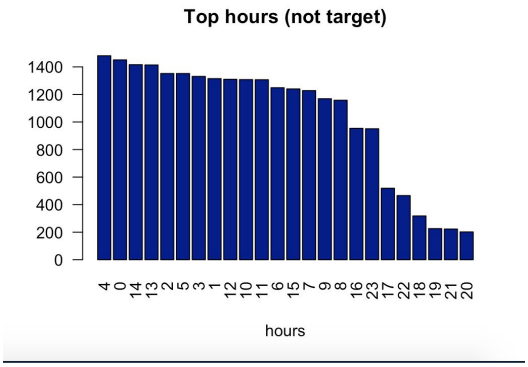
For the “device” attribute, the majority of devices fall under the category valued 1 between the blue and the red graphs indicating that it may not be an important attribute. For “os” attribute, there were some values (19, 17, 13, 8, and 6) that overlaps between the two graphs suggesting that this attribute may or may not be relevant.

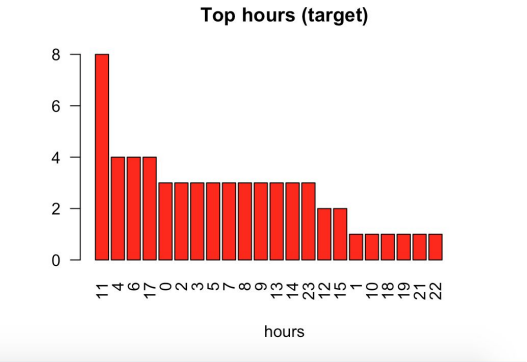


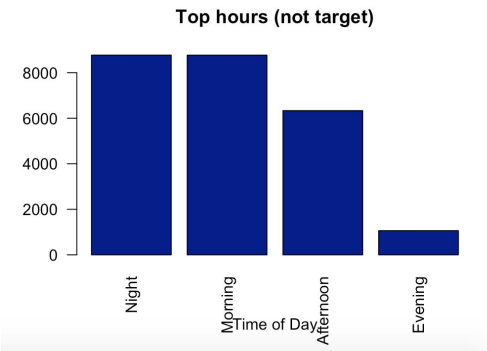
For “channel” attribute, only the value 113 was consistent between the two graphs indicating that this could be another important attribute for our analysis. And for the “days” attribute, the distribution between the blue and the red graphs appears similar suggesting that this may not be as important.

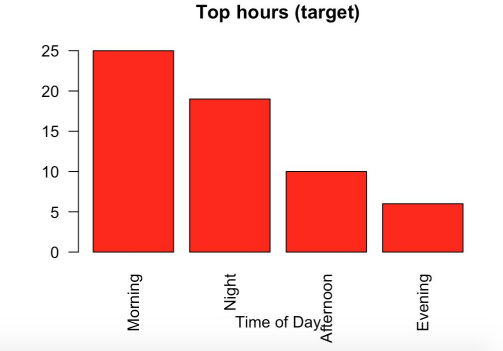


The attribute, “hour” was looked at as well, however, having the data spread out across the 24 hours in the day was hard to examine and interpret. The levels of the hours were then condensed down to morning (5-11), afternoon (12-4), evening (5-7), and night (everything else) for easier interpretation of the data in this attribute. Between the two graphs, it again shows similar distribution between the two groups indicating that this attribute may not be relevant for our analysis.





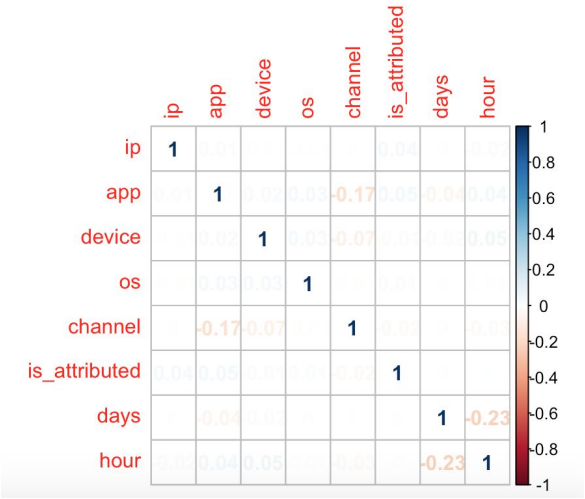




Now that the data is visualized, we can start to answer some of the questions that were set in the abstract.

- Are there particular times in the days where a user will download an application?
  - The red graph shows that morning and night seems to be the time of the day where users would download an application. However, comparing the two graphs from above also shows that there was little differences in terms of times of the day between the target and non-target variable. What is interesting to note is that in both the target and the non-target variable that was fewer clicks and downloads in the evening.
- Do fraudulent clicks have any relations to the types of mobile devices used?
  - For this question, there is no differences in both the target and the non target-variable as both graphs shows the same distribution. It should be noted that the majority of devices used are in the category valued 1.
- Are there particular types of an application that a user will be more likely to download?
  - There seems to be differences between the apps downloaded as demonstrated in the graphs for the target and non-target variable. Only the application valued at 9 was consistent between the two graphs.

Correlation plot was also done to examine if any attributes were highly correlated to each other. Spearman was used since all the attributes are based on ordinal data and not on interval data. The resultant chart shows there there is a negative weak relationship between the attributes, “channel” and “app” and also between “days” and “hour”.



Step 3: Feature selection

Feature selection was done using StepAIC in the forward direction. As seen below, only “ip”, “app”, “channel” and “os” attributes were determined to be significant and should be used as features for the modelling process.

```
> summary(forward)

Call:
lm(formula = is_attributed ~ ip + app + channel + os, data = reduced.set)

Residuals:
     Min       1Q   Median       3Q      Max
-0.12450 -0.00415 -0.00184  0.00014  0.99995

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -5.851e-04  8.526e-04  -0.686 0.492544
ip           3.910e-08  4.454e-09   8.779  < 2e-16 ***
app          1.702e-04  1.984e-05   8.582  < 2e-16 ***
channel     -8.531e-06  2.377e-06  -3.588 0.000334 ***
os          -1.533e-05  5.339e-06  -2.870 0.004105 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.04877 on 24995 degrees of freedom
Multiple R-squared:  0.006577,  Adjusted R-squared:  0.006418
F-statistic: 41.37 on 4 and 24995 DF,  p-value: < 2.2e-16
```

The removal of attributes as determined by the feature selection was done at this stage as a result of the StepAIC in the forward direction. Thus, the only attributes that would be carrying forward into the modelling process is “ip”, “app”, “os”, “channel”, and “is_attributed”.

A look into the patterns from visualizing the data into graphs in the exploratory analysis section seems to support the features selected from the StepAIC in the forward direction. The attributes, “device”, “days” and “hour” were determined earlier that they may not be relevant for analysis and modelling.

Step 4: Classification algorithms

The reduced dataset (with consideration of the attributes from the feature selection process) is now split into another train and test(validation) datasets for modelling purposes. A seed was set to ensure future replication of this split. The split was done at a 70/30 split for training and testing respectively. A check of the distribution of the target variable still shows that the target variable is at 0.0028% which is close to the original train dataset at 0.0025%. At this point in time, there is still a major class imbalance for the attribute, “is_attributed”. Repeated cross validation was done with 5 folds and 3 repeats to prevent overfitting of the data. Modellings were done with no balancing, SMOTE (Synthetic Minority Oversampling Technique) balancing, and oversample balancing using the RandomForest and XGBoost algorithms which is found from the caret package. A constant seed was also set for each of the training models to keep constant of the folds and resamplings for future comparison of models.

```
> table(train.set$is_attributed)

    No Target
 17451     49
```

Step 5: Evaluation of models

Confusion matrix and Area Under the Curve (AUC) based on the Receiver Operating Characteristic Curve (ROC) curve were used for evaluations of the models. Properties of the confusion matrix such as accuracy, sensitivity, specificity, precision, recall and F1 score are also considered. AUC - ROC curve considers the true positive rate (TPR) against the false positive rate (FPR) through varying thresholds. The greater the AUC - ROC score, it would then indicate better classification performance of the model. The AUC - ROC score would also allow us to compare the different models. In addition, AUC - ROC curve was used over Precision-Recall (PR) curve due to the fact that we will be balancing our datasets using SMOTE and oversampling. Statistical significance was also considered for the models by determining any differences from the resampling processes which was held constant across the different models by using the same seed. The paired t-test is assumed as the resampling distribution is the same between models because of the same seed, thus, allowing us to see if there are any significant differences between the models. The function also assumes the bonferroni correction for p-value adjustment for multiple comparisons. Null hypothesis was already set in that there are

no differences between the models while the alternative hypothesis is to reject the null hypothesis in which there are differences between the models.

Coding for each of the steps can be found in each of the respective areas in the FinalClickFraud.R file from the link: *https://github.com/PHLHY/Capstone-*

Results

Confusion matrix statistics is the first evaluative metric for consideration on the unbalanced dataset, balancing with SMOTE, and finally with oversampling using the RandomForest and XGBoost algorithms.

Starting with the no balancing dataset, both algorithms shows a high percentage of accuracy, however, both RandomForest and XGboost does have poor sensitivity affecting precision, recall, and F1 scores. Sensitivity was 0 in both cases as the models did not accurately identified any true target variables. So far, both of these no balancing models are not effective based on the confusion matrix statistics.

Confusion Matrix Statistics for No Balancing

	Random Forests	XGBoost
Accuracy	0.9983	0.9985
Sensitivity	0	0
Specificity	0.9997329	1
Precision	0	NA
Recall	0	0
F1	NA	NA

The next of modelling was done by balancing the training dataset using SMOTE while using the same two algorithms, RandomForest and XGBoost. Just like above, accuracy is in the 90s for both the RandomForest and XGBoost algorithms. However, while sensitivity and specificity are high compared to the no balancing models, it is noticeable that the precision for both algorithms suffered. The reason for this is because a high proportion that was identified as the target variable in the algorithms were actually not relevant. The confusion matrix of both the RandomForest and XGBoost using SMOTE shows a high amount of false positives (type 1 error) which affected the precision and the F1 scores.

Confusion Matrix Statistics for SMOTE Balancing

	Random Forests	XGBoost
Accuracy	0.9447	0.9131
Sensitivity	0.909091	0.818182
Specificity	0.944719	0.913206
Precision	0.023585	0.013657
Recall	0.909091	0.818182
F1	0.045977	0.026866

The last approach was done using the oversampling balancing technique, and again both algorithms have high accuracies in the 90s as well. Sensitivity in both of these algorithms are lower than the models that were balanced using SMOTE. However, it does have a higher precision score and resulting F1 scores for both the RandomForest and XGBoost algorithms compared to the other models.

Confusion Matrix Statistics for Oversampling

	Random Forests	XGBoost
Accuracy	0.9973	0.9953
Sensitivity	0.1818182	0.4545455
Specificity	0.9985312	0.9961277
Precision	0.1538462	0.1470588
Recall	0.1818182	0.4545455
F1	0.1666667	0.2222222

So far, it is safe to assume that accuracy would not be a good indicator for evaluation as all the models are in the 90s. Based on research articles from the literature review section, precision is the key metric that we are aiming for here as we are trying to determine users downloading an application when clicking on a relevant advertisement. Based on this, RandomForest and XBoost with oversampling balancing had the highest precision scores. With consideration of recall, the F1 score would also be an appropriate evaluative measure as it balances out both precision and recall. However, since the overall precision and the F1 scores are low, another evaluative measure should be considered at this time.

To further analyze the results, another evaluative measure was done based on the Area of the Curve (AUC) of the Receiver Operating Characteristic (ROC) curve.

Based on the AUC - ROC curve, we can see that all algorithms and balancing methods are in the 90s which is usually a good score. As mentioned prior, a high AUC - ROC score is desirable, however, all of our models appears to scored similarly to each other.

AUC ROC Results

	None	SMOTE	Oversampling
Random Forests	0.9429	0.97	0.9314
XGBoost	0.9687	0.9351	0.9726

Statistical significance for the models should be looked at as the AUC - ROC scores appears similar to each other and it should be determined whether the differences in scores are significant. The p-value (which is shown below the diagonal in the chart below) is used to distinguish whether a model is significantly different to other models. A p-value of less than 0.05 is assumed to reject the null hypothesis.

Since the AUC is based on the ROC curve, the ROC was used for comparison between the models to see if the differences are significant.

```
p-value adjustment: bonferroni
Upper diagonal: estimates of the difference
Lower diagonal: p-value for H0: difference = 0

ROC
           RF_NB     XGB_NB    RF_SMOTE  XBG_SMOTE RF_UP     XGB_UP
RF_NB                -0.094942 -0.036986 -0.070935 -0.045130 -0.080962
XGB_NB     0.0010032            0.057956  0.024007  0.049812  0.013980
RF_SMOTE   0.1038082 0.0003148           -0.033949 -0.008144 -0.043976
XBG_SMOTE  0.0190770 0.0348973 0.0351131            0.025805 -0.010027
RF_UP      0.5750569 0.0040562 1.0000000 1.0000000           -0.035832
XGB_UP     0.0212108 1.0000000 0.0665815 1.0000000 0.0421181
```

In this case, XGBoost which is balanced by oversampling will be the main model for comparison due to it having the highest AUC - ROC score. The following comparisons are made below with the cell being filled by yellow indicating that there is a significant differences between oversampling XGBoost and to no balancing RandomForest and to oversampling RandomForest. From the AUC - ROC scores seen above, we can see this made sense as they had lower AUC - ROC scores and that the higher AUC - ROC score is in favor of the oversampling XGBoost model. For the rest of the models, we can not reject the null hypothesis.

Statistical Differences Between Models (P-Values) - ROC Metric

Main Model for Comparison = Oversampling XGBoost

	Oversampling
NB - RF	2.12E-02
NB - XBG	1
SMOTE - RF	6.66E-02
SMOTE - XBG	1
OVER - RF	0.0421181

Conclusion

The Kaggle challenge was to build an algorithm for determination of users that will download a mobile application after clicking on a relevant advertisement. A test dataset was provided by TalkingData for this competition to apply the algorithm of choice for submission. Based from the results using the evaluative measures above, the AUC - ROC scores should be the main evaluative metric used as it has scores that are high compared to the statistics that are found in the confusion matrix. Since **oversampling XGBoost model** has the highest AUC - ROC score and is significant different compared to two of the other models in regards to the ROC, **this model will be the choice selection for this Kaggle challenge**. In addition, while the overall precision and F1 scores are low from the confusion matrix, oversampling XGBoost again also has the second highest precision score and the highest F1 score further supporting that it should be the choice model for this competition.

Some limitations of this project are identified below:
Computational limitations

- Dataset was reduced significantly to only 25000 points for training and testing. This could impact on important patterns not seen and potentially affect the training of models
- Some of the attributes were not converted to categorical data type due to the many different levels resulting in memory errors while running models. Attempts were made at reducing the amount of levels, however, a significant amount of information was lost as a result (ie. 15046 unique ips in the reduced set)

Further exploratory analysis and feature engineering

- “Attributed_time” was not examine closely as it would have provided similar information to “is_attributed”. However, differences in time between “click_time” and “attributed_time” could have resulted in a new attribute which could have been potentially important.
- Potentially all attributes could have been included for modelling as a benchmark for comparison

Modelling

- Preprocessing was not done (ie. standardizing such as scaling and centering) due to all algorithms mostly being tree-based and not on calculating distances. Perhaps preprocessing could have been considered to see how it affected the evaluation of the models.
- Could have considered different types of balancing (ie. ROSE, undersampling etc.)

References

1. Berrar, Daniel. "Learning from automatically labeled data: case study on click fraud prediction." *Knowledge and Information Systems* 46.2 (2016): 477-490.
2. Cho, Geumhwan, et al. "Combating online fraud attacks in mobile-based advertising." *EURASIP Journal on Information Security* 2016.1 (2016): 2.

3. Dietterich, Thomas G. "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization." *Machine learning* 40.2 (2000): 139-157.
4. Haider, Ch Md Rakin, et al. "An ensemble learning based approach for impression fraud detection in mobile advertising." *Journal of Network and Computer Applications* 112 (2018): 126-141.
5. Jansen, Bernard J. "Click fraud." *Computer* 40.7 (2007): 85-86.
6. Loh, Wei-Yin. "Classification and regression trees." *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 1.1 (2011): 14-23.
7. Oentaryo, Richard, et al. "Detecting click fraud in online advertising: a data mining approach." *The Journal of Machine Learning Research* 15.1 (2014): 99-140.
8. Phua, Clifton, et al. "Feature engineering for click fraud detection." *ACML Workshop on Fraud Detection in Mobile Advertising*. 2012