Amanda Ahn

Final Report: Click-Through-Rate Prediction

Introduction

Background

Online advertising revenue equates to more than hundreds of billions of dollars and is only going to continue to increase as we innovate society further. Accurately predicting click-through rate can help companies increase their revenue from online ads by altering features of the advertisements. Not only can this improve the number of customers the advertisers receive from more people clicking on their ads, but it also increases the amount of money the partnered website receives from the original advertising company. Finding an accurate machine learning model to predict click-through rate helps businesses' revenues all-around.

Problem Statement

An average of 60-80% of company revenues in the US comes from advertisements and ad placements play a major role in increasing click through rate (CTR). Efficiently predicting CTR would help marketing officers and advertisement directors. The goal of this project is to identify the optimal location to place advertisements in a webpage to increase the CTR by 20% by using supervised machine learning techniques.

Audience

Our project and findings will be geared towards addressing marketing, advertising, and design directors. These people are in charge of every aspect in releasing online advertisements whether it be the number released, the placement, the time, the frequency, and so on.

Data

The dataset we chose was the Avazu CTR Prediction dataset retrieved from Kaggle. There are three datasets provided, but for the sake of my research, I only utilized the train csv dataset because it had more than enough data (over 40 million logs). It has 24 column features detailing each online advertisement and whether it was clicked on or not.

https://www.kaggle.com/c/avazu-ctr-prediction

Data Wrangling/Data Cleaning

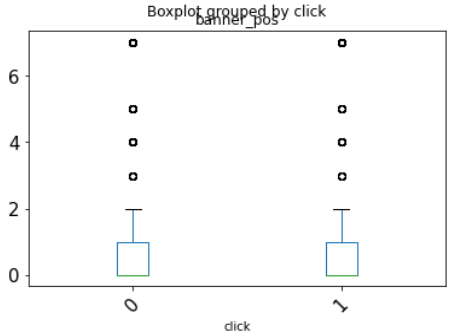
The ctr-data-wrangling notebook portrays information about the shape of the data and the column feature names. First, we analyzed the information of the data by using methods df.shape, df.columns, df.dtypes. We found the data contained 40,428,967 rows and 24 columns. The column names were ['id', 'click', 'hour', 'C1', 'banner_pos', 'site_id', 'site_domain', 'site_category', 'app_id', 'app_domain', 'app_category', 'device_id', 'device_ip', 'device_model', 'device_type', 'device_conn_type', 'C14', 'C15', 'C16', 'C17', 'C18', 'C19', 'C20', 'C21']. The 'id' column is a float type, click, hour, C1, banner_pos, device_type, device_conn_type, and C14-C21 are all integer types, and the rest are object types. After overseeing the general facts

about the data, we moved on to dropping all null and duplicate values. Luckily, our dataset did not have any null or duplicate values, so cleaning was easy. Additionally, all unnecessary columns that would have no numerical importance were dropped, which included the id and site_id columns.

Exploratory Data Analysis

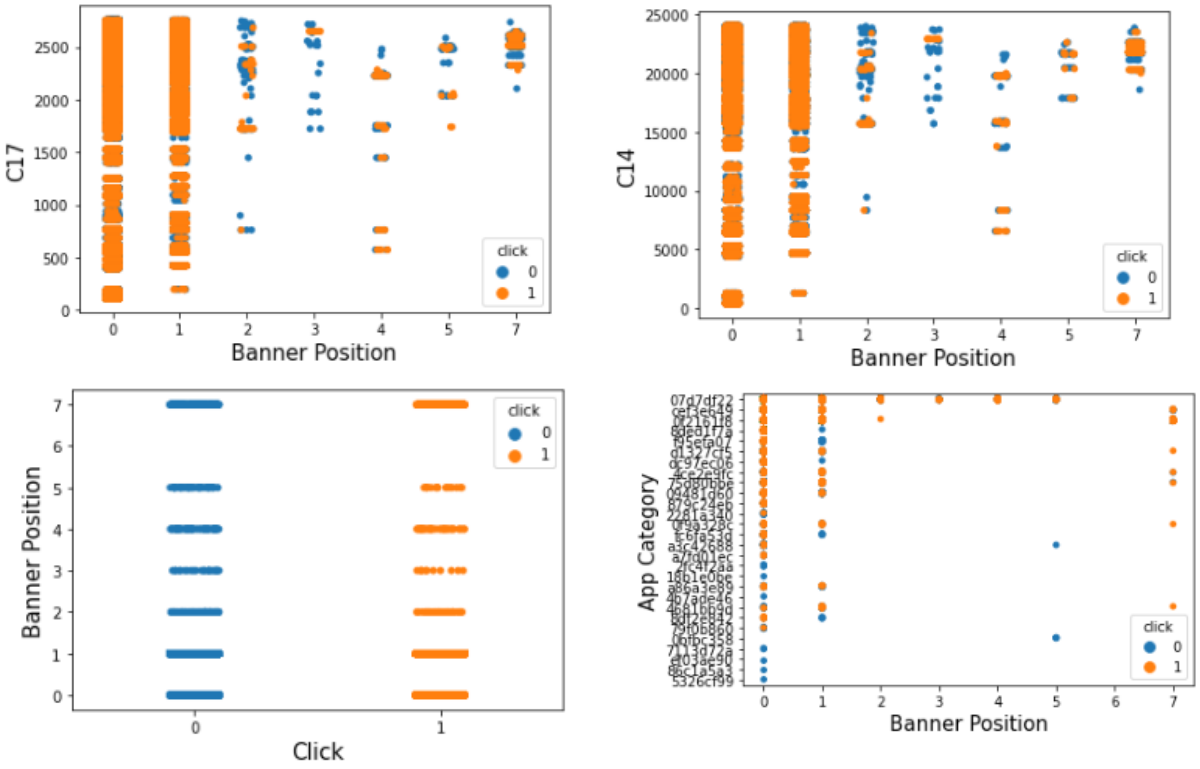
Graphs/Plots

For the sake of faster exploratory data analysis, we sampled our dataset to significantly decrease the amount of data. Then, we plotted a barplot of the click and banner_pos columns because we hypothesized that banner position would be the most important feature in predicting click through rate.



However, our boxplot did not display any significant conclusions, so we went on to use seaborn pairplot to visualize the comparison between each column with each other. From this pairplot, we saw that features C14 and C17 seemed to be the ones that had some sort of positive correlation with each other. These features could hold some importance in future analysis, so we will be investigating these features further.

Following up with this analysis, we used seaborn stripplot to plot specific columns with respect to click to see if we could draw any potential conclusions.



From these plots, we have found that in order to predict click, features C14 and C17 could show some importance in future models. However, we were not able to find any definite importance features from our EDA, so we will be leaning more on the modeling portion to reach conclusions.

Hypotheses

- *Null Hypothesis:* The prediction of a click is due to chance and not due to the (banner) position of the advertisement on the site/app/device.
- *Alternative Hypothesis:* The prediction of a click is not due to chance and has something to do with the (banner) position of the advertisement on the site/app/device.

Going forward, I believe these null and alternative hypotheses would be a good start to our analysis because most of the features are hidden features. Even if we fail to reject the null, then the analysis would be informative because we would know the banner position has no effect on click through rate. It would be a better idea to have a null hypothesis that we can reject so we can find something significant, but it is difficult to make predictions when the majority of the features are unknown.

Preprocessing Training Data Development

Creating Dummy Features/One-hot Encoding

Based on previous analysis of the data, there is no need to create dummy features because I am using the click column as my main feature, and it is already numerical with one two values: 0 for when a click was not present and 1 for when a click was present. Creating dummy variables for any of the other columns would not be helpful and would just make my data a lot larger when it is already very big.

Splitting Testing and Training Subsets

We used train_test_split from sklearn to create our subsets for our training and testing data. X_train and X_test contained all the columns apart from click, and y_train and y_test contained the click column. We used a test size of 0.2 and random state of 100. When we printed the shoes of each of our subsets, we realized they might be too large and unbalanced to efficiently and accurately run models on. Therefore, we sampled the data to make it smaller. Our new training and testing data composed of 335,639 rows for non-clicks and 205,952 rows for clicks.

Standardization

- There is no need to standardize my values because most of them are categorical and the rest are unknown variables (C14-C21), so we wouldn't know exactly what standardizing would do to them.
- Our two subsets are still somewhat unbalanced, so we made note that we may need to use an oversampling method to balance the data. However, we did not apply the method yet because we want to use our organic data in our models first and see how they perform. If the performance metrics are lacking, we will use RandomOverSampler from Imblearn to balance the number of 0 and 1 clicks.

Modeling

Based on our data and project, we deduced that logistic regression would be the best model to test first. After seeing the results of our logistic regression model, we will be able to test other predictive models that could potentially obtain better results.

First Model: Logistic Regression

We used LogisticRegression from sklearn and fitted it to our X_train and y-train data subsets. This is the printed classification report and accuracy score.

	precision	recall	f1-score	support
0	0.62	0.97	0.76	67128
1	0.42	0.03	0.06	41191
accuracy			0.61	108319
macro avg	0.52	0.50	0.41	108319
weighted avg	0.55	0.61	0.49	108319

Accuracy Score: 0.6149982920817216

Next, we used cross_val_score from sklearn to calculate the mean cross validation for test and train scores.

```
[0.56904185 0.57170646 0.56874493 0.56117495 0.56779267]
Mean cross validation test score: 0.5676921711582379
Mean cross validation train score: 0.5684506178500486
Standard deviation in cv test scores: 0.0035079554870482425
```

Because we did not get the best results, we will move on to test our next predictive model, Random Forest.

Second Model: Random Forest

We used RandomForestClassifier from sklearn and fitted it to our X_train and y-train data subsets. This is the printed classification report and accuracy score and the mean cross validation for test and train scores.

```
              precision    recall  f1-score   support

           0       0.69      0.75      0.72     67128
           1       0.53      0.45      0.49     41191

    accuracy                           0.64    108319
   macro avg       0.61      0.60      0.60    108319
weighted avg       0.63      0.64      0.63    108319

Accuracy Score:  0.6377828451148921
```

```
[0.63983914 0.63203599 0.64046306 0.63785728 0.64052348]
Mean cross validation test score: 0.6381437886637685
Mean cross validation train score: 0.6528238289912107
Standard deviation in cv test scores: 0.0032032318344256297
```

As expected, our Random Forest model had better results than our Logistic Regression model, especially for the minority click group. Since our Random Forest model performed better, we will be testing a Gradient Boosting model next, rather than a Decision Tree model, and identify which model of the three is the best.

Third Model: Gradient Boosting

We used GradientBoostingClassifier from sklearn and first tested learning rates, 0.05, 0.1, 0.25, 0.5, 0.75, and 1 to find the optimal learning rate. After seeing the optimal learning rate was 1, with a training accuracy score of 0.65 and validation accuracy score of 0.65, we fitted the classifier to our X_train and y-train data subsets. This is the printed classification report and accuracy score and the mean cross validation for test and train scores.

```
[[60186  6942]
 [30531 10660]]
              precision    recall  f1-score   support

           0       0.66      0.90      0.76     67128
           1       0.61      0.26      0.36     41191

    accuracy                           0.65    108319
   macro avg       0.63      0.58      0.56    108319
weighted avg       0.64      0.65      0.61    108319

0.6540496127179903
```

```
[0.67245329 0.66695963 0.67128813 0.67956636 0.67452751]
Mean cross validation test score: 0.6729589833945869
Mean cross validation train score: 0.6681433491516522
Standard deviation in cv test scores: 0.004126920509444569
```

Applying Balanced Data to Models

Because we did not receive amazing performance metric results, we will try RandomOverSampler from Imblearn to create new X and y subsets, in which the data in the click column is more balanced. I will refit the three models to the new subsets to see if we get better scores.

	precision	recall	f1-score	support
0	0.58	0.64	0.61	67128
1	0.60	0.53	0.56	67128
accuracy			0.59	134256
macro avg	0.59	0.59	0.59	134256
weighted avg	0.59	0.59	0.59	134256

Accuracy Score: 0.5881152425217495

Logistic Regression Model Results with new X and y

	precision	recall	f1-score	support
0	0.64	0.57	0.60	67128
1	0.61	0.67	0.64	67128
accuracy			0.62	134256
macro avg	0.62	0.62	0.62	134256
weighted avg	0.62	0.62	0.62	134256

0.6224228339887975

Gradient Boosting Model Results with new X and y

	precision	recall	f1-score	support
0	0.68	0.63	0.65	67128
1	0.66	0.71	0.68	67128
accuracy			0.67	134256
macro avg	0.67	0.67	0.67	134256
weighted avg	0.67	0.67	0.67	134256

Accuracy Score: 0.6680073888690263

Random Forest Model Results with new X and y

After applying the new X and y subsets, our winning model is now our Random Forest Model. With the highest accuracy score of 0.67.

Hyperparameter Tuning

First, we tried several different hyperparameter tuning methods to acquire the best hyperparameters for our Random Forest model, but they wouldn't complete running. Therefore, we simplified our GridSearchCV method to just test the best number of estimators, which was 600. Then we tuned our random forest model with our new set of optimized parameters. Here are our old random forest parameters vs. our new random forest parameters.

```
{'bootstrap': True,
 'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': 'auto',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 100,
 'n_jobs': None,
 'oob_score': False,
 'random_state': None,
 'verbose': 0,
 'warm_start': False}
```

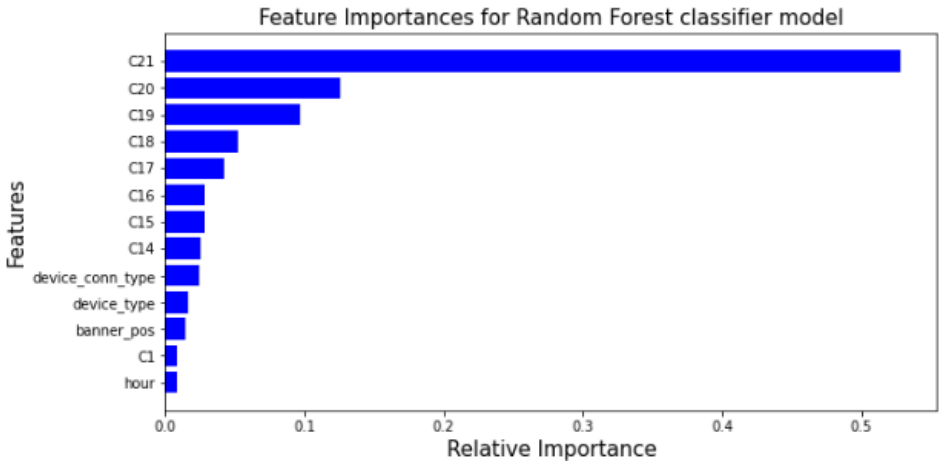
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
       max_depth=None, max_features='auto', max_leaf_nodes=None,
       min_impurity_decrease=0.0,
       min_samples_leaf=1, min_samples_split=2,
       min_weight_fraction_leaf=0.0, n_estimators=600, n_jobs=-1,
       verbose=1, random_state=None, warm_start=False, oob_score=False)
```

However, our new accuracy score stayed the same at 0.67. This will be our final score.

Feature Importances

Next, we calculated our feature importances for each feature and plotted them.

	Features	Importance scores
0	hour	0.008086
1	C1	0.008771
2	banner_pos	0.014281
3	device_type	0.016927
4	device_conn_type	0.024372
5	C14	0.025311
6	C15	0.028052
7	C16	0.028483
8	C17	0.042065
9	C18	0.052380
10	C19	0.097249
11	C20	0.125926
12	C21	0.528097



Here we see a significant score for feature C21, which will be our most significant feature in predicting click through rate. Therefore, we fail to reject our null hypothesis and conclude that banner position has no significant effect on click through rate.

Conclusion

Overall, our final results were not the best, but they were decent enough for us to reach a conclusion. We chose the best model and parameters that we could find through our analysis. The best model was the Random Forest Model, while using GridSearchCV for hyperparameter tuning. We can conclude that (unknown) feature C21 is the most predictive in predicting click through rate for an online advertisement with our model accuracy score being 0.67 and ROC-AUC score of 0.72. Although we determined that feature C21 is the most significant in predicting click through rate, there is not much we can conclude in the real world because we do not know what that feature is. Therefore, there are no follow up steps we can offer to stakeholders to make our findings significant other than the fact that feature C21 should be paid more attention to if companies want to increase the click rates on online advertisements they post.