ISOM 674: AI and Machine Learning

Predicting Clicks for On-Line Advertisements

Project Description

Team 15

Francis Jingo, Sonali Pandit, Lanston Chen,

Jean-Baptiste Habyarimana, & Alejandro Chumaceiro

Problem Understanding

Given the increasing rates of mobile phone usage and widespread access to the internet, companies are seeking to generate further value from online advertisements. According to a study published by the Statista Research Department, digital advertising spending worldwide for the year 2021 was recorded at $522.5 billion dollars. Hence, given the significant capital investments in digital advertisements, invested companies want to maximize their returns. For small-to-medium sized firms, creating targeted ads with a higher probability of interaction is crucial as they do not have the spending power to probe their markets.

The crux of this project was to employ machine learning methodologies to develop a model that predicts, given a set of features on the user and the unique identifier for the advertisement, the probability of that user clicking said online advertisement. For the context of this project, we have data collected by an *undisclosed* company in excess of 30 million instances of whether or not specific advertisements were clicked. The data utilized to train the predictive model consists of 4.9 GB of user interactions and is located within the file "*ProjectTrainingData.csv*". Furthermore, the predictions of our final model will be evaluated using the negative of log likelihood as the loss function.

Data Understanding

Each observation within the *TrainingData* file represents a specific ad, identified by a unique number, and whether or not a user clicked it. The collection period for advertisement interactions spanned from October 21st 2014, to October 29th of the same year. In the context of the predictive model, the response variable is encompassed by the column *click* which assigns 0 for no click and a 1 for a successful click. On the other hand, all explanatory features within the dataset encoded categorical variables. While ads are uniquely identified by the column *id*, users are identified by other features in the dataset such as: *device id*, *device ip*, and *device model* among others. These features can be grouped into three primary categories: user-related, ad-related, and anonymized features (**Table 1**).

The data used for model evaluation, *ProjectTestData*, follows the same structure as the training data but is limited to 13 million observations. For future reference, we will apply the same transformations across the datasets.

Feature Group	Variables	Description
Ad Information	id	Unique identifier for the ad
	hour	Time information about when the ad was displayed
	banner_pos	The position in the banner
	site_domain side_id site_category	Identifiers for the web site showing the advertisement
	app_domain app_id app_category	Identifiers for the application showing the advertisement
User Information	device_id device_model device_type	Identifiers for the device for which the ad was displayed, including IP, type, etc.
	device_conn_type	The type of the device's connection
Anonymized	C1, C14 - C21	Anonymized categorical variables which are not disclosed by the business

***Table 1**: Feature descriptions present in the training and testing data.*

Data Ingestion and Exploration

Initially, we utilized the template files to read-in the training data in CSV format as a matrix within R. However, we quickly realized that the massive size consumed significant resources when loaded into memory. Therefore, we opted to instead use PySpark with Python to load the data into memory. Since the data exceeded 4 GB of size, we decided to convert the CSV files into Parquet files for easier handling. Furthermore, Parquet's column level compression can significantly reduce file sizes and decrease execution times for data processing.

From our data understanding phase we realized that the data was predominantly categorical. Hence, the explanatory features required specialized summary statistics, focusing on data types, count of unique values (i.e., categories), and null values across all columns. Clarifying the number of unique instances in a categorical variable is a crucial step for implementing them in machine learning models. To illustrate, when categorical features are not inherently ordered or follow a pre-defined hierarchy the technique to handle them completely changes. Therefore, it was crucial for us to implement a summary table to define the current state of our features.

	Feature Name	Data Type	Unique #	Description
0	id	float64	31991090	the identifier of the ad (may or may not be un...
1	click	int64	2	1 means the ad was clicked on while click = 0 ...
2	hour	int64	216	the date and hour when the ad was displayed. F...
3	C1	int64	7	an anonymized categorical variable
4	banner_pos	int64	7	the position in the banner
5	site_id	object	4581	an identifier for the web site
6	site_domain	object	7341	an identifier for the site domain
7	site_category	object	26	a code for the site's category
8	app_id	object	8088	an identifier for the application showing the ad
9	app_domain	object	526	an identifier for the app's domain
10	app_category	object	36	a code for the category of the app
11	device_id	object	2296165	an identifier for the device used
12	device_ip	object	5762925	a code for the ip of the device
13	device_model	object	8058	the model of the device
14	device_type	int64	5	the type of the device
15	device_conn_type	int64	4	the type of the device's connection
16	C14	int64	2465	an anonymized categorical variable
17	C15	int64	8	an anonymized categorical variable
18	C16	int64	9	an anonymized categorical variable
19	C17	int64	407	an anonymized categorical variable
20	C18	int64	4	an anonymized categorical variable
21	C19	int64	66	an anonymized categorical variable
22	C20	int64	171	an anonymized categorical variable
23	C21	int64	55	an anonymized categorical variable

***Figure 1**: Descriptive summary of features within the TrainingData set*

To further understand the relation of our features to our response variable *click*, we performed standard exploratory data analysis. Hence, we plotted bar graphs of various variables against our target variable to observe the distribution of clicks across different categories. These visualizations provided insights into which categories or features might have more influence on the likelihood of a click.

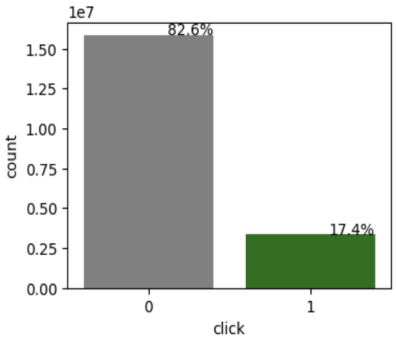

***Figure 2**: Distribution of feature click.*

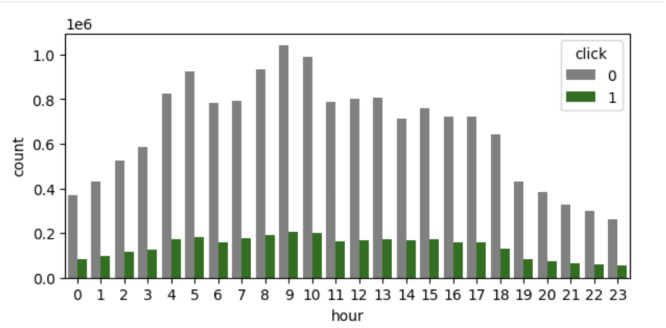

***Figure 3**: Distribution of feature click by hour of day.*

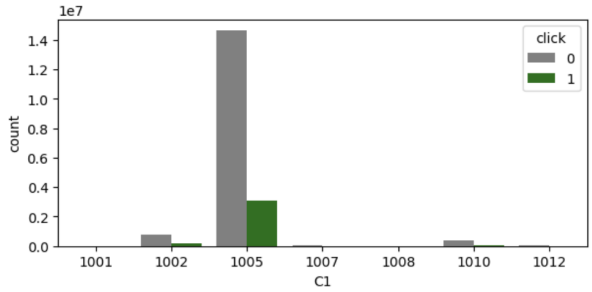

***Figure 4**: Distribution of feature click by anonymized feature C1.*

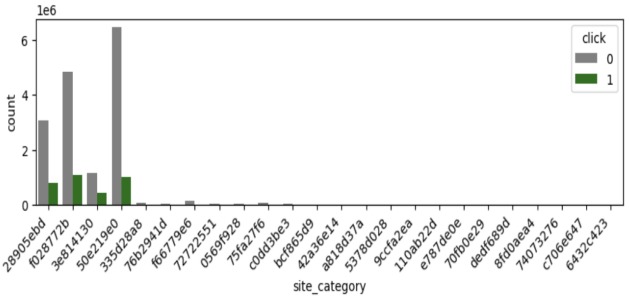

***Figure 5**: Distribution of feature click by site category.*

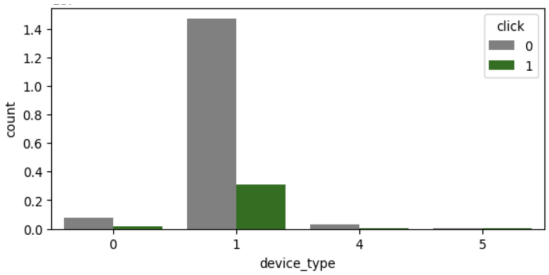

***Figure 6**: Distribution of feature click by device type.*

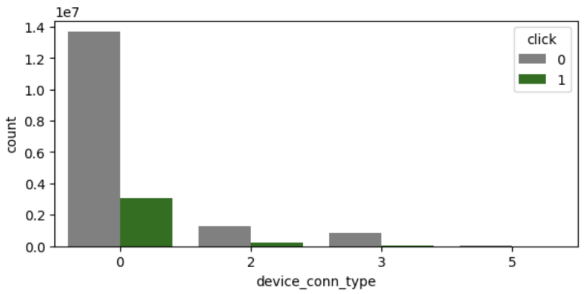

***Figure 7**: Distribution of feature click by device connection type.*

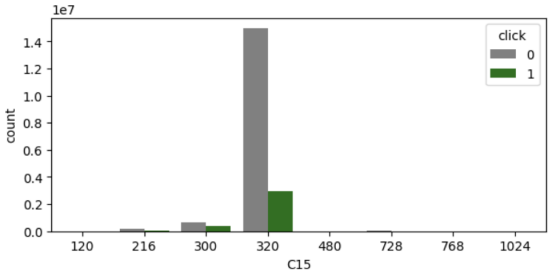

***Figure 8**: Distribution of feature click by anonymized feature C15.*

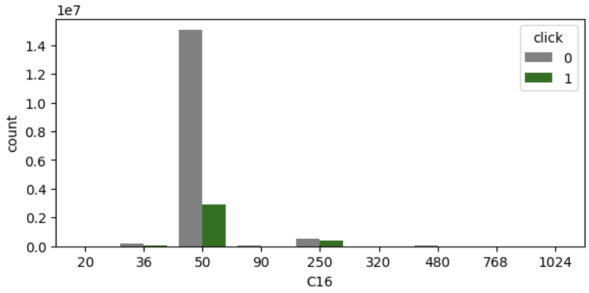

***Figure 9**: Distribution of feature click by anonymized feature C16.*

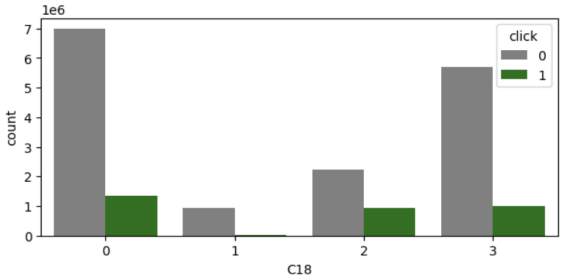

***Figure 10**: Distribution of feature click by anonymized feature C18.*

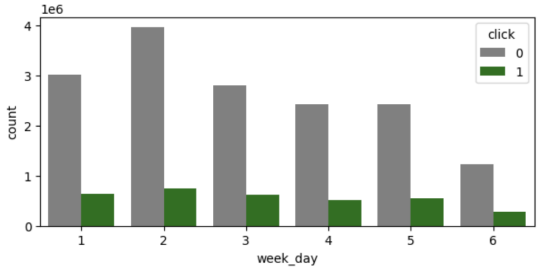

***Figure 11**: Distribution of feature click by week day.*

Utilizing the information derived from our exploratory data analysis phase we concluded the following insights:

- *Site_id*, *app_id*, *device_id* and *device_conn_type* have over 1000 categories encoded.
- The target variable *click* is unevenly distributed, meaning only 17.4% of advertisements result in clicks. The uneven distribution of clicks further emphasizes the need for creating targeted ads, which should in turn reduce the proportion of ads not clicked and by extension increase return on investment for online advertising.
- The distribution of clicks throughout the day appear to feature high variance, meaning clicks seem to be evenly distributed across the day. While not by a significant magnitude, it is important to highlight that the highest proportion of clicks occurred during the morning between 8-10 am.
- While we do not dispose of explanatory power for the anonymized C features, we have discovered the following:
    - C1 has the highest number of clicks and impressions for type 1005,
    - C15 has the highest number of impressions for type 320, and
    - C16 has the highest number of impressions for type 250

- Online advertising interactions are significantly concentrated across 4 site categories, which yields crucial information about where to strategically locate ads.
- Advertisements are being primarily displayed through device types 1 and 0, meaning that advertisements are not being shown to other device categories.

Data Processing

To reduce the workload inherent in massive data, we randomly sampled and partitioned the training data into 20 equally sized parts. Therefore, instead of processing 30 million rows for each operation we identified it would be easier to process 1.5 million rows per operation. For each resulting partition, we can further partition using 70-30 percent splits into training and validation data, respectively. For each partition of the training data we can generate 1.2 million rows of training data and ~300,000 rows of validation data. With these extensive partitions we are able to average out results as a means to create generalization performance.

As previously mentioned, handling categorical variables was a challenging task in the context of this project. Ideally, the method for encoding these variables would simply be one-hot encoding, which splits a categorical feature into K distinct binary columns for each category within the original feature. However, given the dataset's size and the extensive number of categories like in site domain and C14. Simply applying one-hot encoding to these extensive features would create a significant amount of new features which can potentially lead to overfitting and the curse of dimensionality. Hence, we designed a more tailored approach to handling categorical features:

- **High Cardinality Features**: Features with more than 10 categories were treated using count/frequency encoding and *Click Through Rate* (CTR). To illustrate, CTR is defined as the number of clicks / number of advertisements. On the other hand, count encoding essentially replaces the categories with the count of the observations that show that category in the dataset. The goal with this approach to categorical encoding is to make categories ordinal and more interpretable for the model.

- **For Low Cardinality Features**: Columns with fewer than 10 categories were dummy encoded using one hot encoding. While this methodology can lead to higher dimension curse, these new features marginally increase the total number of features hence reducing the impact of this phenomenon.

Model Building (Base Case)

Referencing the methodologies covered in class, we decided to train and test three basic classification models for the purpose of binary classification. Furthermore, the process of training and testing the models was performed in two batches: using only one random partition, and using five random partitions. However, here is the underlying rationale for our choice of models and their respective performance based on log loss.

- **Decision Tree**: Fit as it is capable of capturing non-linearity in the data and yields interpretable results for business stakeholders. Therefore, simplicity and readability were our primary criterion for fitting this model. Without tree pruning, this type of classifier is prone to overfitting.
- **Random Forest Classifier**: The benefits of this ensemble method lie with its ability to create robust predictions based on bootstrap and feature samples. Generally, this methodology yields higher accuracy than deploying a single decision tree.
- **Logistic Regression**: Deployed as it is inherently designed for binary classification and prediction of an outcome. While is part of the family of linear models, it generates straightforward results with the downside of assuming linearity between features and the log odds of the outcome.

Validation Log Loss	Logistic Regression	Decision Trees	Random Forest
One-partition	0.4520	7.1078	0.6715
Five-partitions	0.450283 +/- 0.00159	7.351183 +/- 0.295264	0.67460 +/- 0.013705

***Table 2**: Model performance on validation data of randomly partitioned splits for initial collection.*

While the first batch generates a single performance metric from log loss, the second batch utilizing multiple partitions allows for generalization performance for each model type. From **Table 2** we clearly observe that **Logistic Regression** had the lowest average log loss and lowest standard deviation. Hence, for both batches of model testing, Logistic Regression outperformed the selection of models.

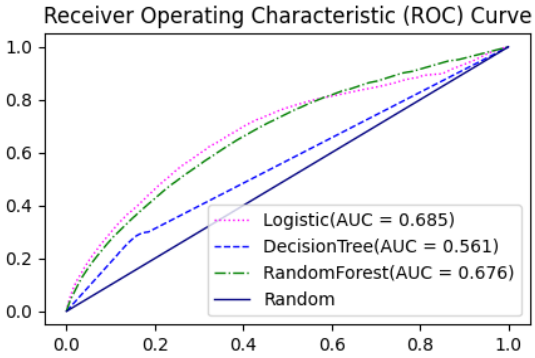

Figure 12: *ROC curve for the 3 models*

Model Optimization

To improve model performance, we utilized forward and backward stepwise regression. In essence, forward stepwise builds the base case model and adds in the next most important variable one at a time. On the other hand, backwards stepwise prunes the complete model (i.e., model that includes all features) by removing features until performance decreases. Since performance is highly dependent on the collection of features, implementing stepwise methods handles collinearity. For each set of optimal features (**Figure 12**) generated from stepwise regression we calculated the validation log loss on 5 of the randomly sampled splits (**Table 3**).

```
.....................................................................
Selected features (forward): ['C1_1002', 'C1_1005', 'C1_1007', 'C1_1008', 'C1_1010', 'C1_1012', 'banner_pos_1', 'banner_pos_2', 'banner_pos_
5', 'banner_pos_7', 'device_type_1', 'device_type_5', 'device_conn_type_2', 'device_conn_type_3', 'device_conn_type_5', 'C15_216', 'C15_300',
'C15_320', 'C15_480', 'C15_728', 'C15_1024', 'C16_50', 'C16_250', 'C16_320', 'C18_1', 'C18_2', 'week_day_1', 'week_day_2', 'time_of_day_ctr',
'site_id_ctr', 'site_domain_ctr', 'site_category_ctr', 'app_id_ctr', 'app_domain_ctr', 'app_category_ctr', 'device_model_ctr', 'C19_ctr', 'C2
0_ctr', 'C21_ctr']
.....................................................................
Selected features (backward): ['C20_ctr', 'C19_ctr', 'C17_ctr', 'app_category_ctr', 'app_id_ctr', 'site_domain_ctr', 'site_id_ctr', 'time_of_
day_ctr', 'month_day_24', 'month_day_23', 'month_day_22', 'week_day_6', 'week_day_5', 'week_day_4', 'C18_3', 'C18_1', 'C16_1024', 'C16_768',
'C16_480', 'C16_320', 'C16_250', 'C16_90', 'C16_50', 'C16_36', 'C15_1024', 'C15_768', 'C15_728', 'C15_480', 'C15_320', 'C15_300', 'C15_216',
'device_conn_type_5', 'device_conn_type_3', 'device_conn_type_2', 'device_type_5', 'device_type_4', 'device_type_1', 'banner_pos_7', 'banner_
pos_5', 'banner_pos_4', 'banner_pos_3', 'banner_pos_2', 'banner_pos_1', 'C1_1012', 'C1_1010', 'C1_1008', 'C1_1007', 'C1_1005', 'C1_1002']
```

Figure 12*: Optimal set of features for forward and backwards stepwise regression.*

Validation Log Loss	Forward Stepwise	Backward Stepwise
Five-partitions	0.38937 +/- 0.00160	0.39300 +/- 0.00123

Table 3*: Generalization performance utilizing optimal set of features.*

Neural Network Fit

With the insights generated from *Model Optimization*, we selected the optimal features from forward stepwise regression and fit them to a neural network model. In terms of parameter tuning, we applied the following: 200 epochs, a batch size of 20000, and a learning rate of 0.001. Then, we initialized 6 dense layers which utilized *relu* as the activation function, with the output layer having *sigmoid* as the activation function to generate a binary classification output. In order to comply with the previous model evaluation methodology, we utilized *binary cross entropy* as the loss function. The neural net was fitted with the same 5 partition approach as done with previous models.

- The neural network model log loss metric was **0.39305 +/-0.00152**

Prediction Generation

Logistic regression remains the best model even when compared with the neural network. We used the logistic regression model with the forward selection features to make predictions on the test data following the approach below.

- Since the test data also was big(~13M records), the test data was split into 20% samples
- For each test split, 10 train splits were used to make predictions
- A majority voting was done based on the predictions and the probability average of the probabilities of the maximum class was output as the prediction

Conclusion

In conclusion, our analysis and prediction report on mobile ad clicks provide actionable insights for advertisers and stakeholders. The comparison of the classifier models including logistic regression and neural networks helps guide decision-making based on the specific requirements of the advertising campaign. Future work to provide continuous improvements to our approach like doing more feature engineering and hyperparameter tuning and adaptation to evolving trends are crucial for maintaining the effectiveness of predictive models.

Appendix

***Project-convert-to-parquets-Team15.py*:** The code file is used to read the csv files containing ProjectTrainingData.csv and ProjectTestData.csv, convert the files into Parquet format and store them in new folders. It contains 4 main functions-one that retrieves the file name and removes the extension, one that extracts and returns the file type , one that converts a single file given its file path to parquet and one that converts multiple files given their folder path. The objective for doing this is to improve execution efficiency. Parquet files are more storage and performance efficient especially in analytics cases.

***Project-Analysis-Team15.ipynb*:** The code file is used for performing exploratory data analysis on the data to give a better understanding of the data which is a crucial step before modeling. It contains two main functions, one that returns the datatype. distinct row count and null count for each column in a table, and one that plots the frequency distribution of clicks across different categorical features.

***Project-Modelling-Team15.ipynb*:** The code is used to create and engineer features, build and compare different models and make predictions. It contains 6 main functions, one that encodes some of the categorical variables using one-hot encoding and the other that calculates train set click through rate for different categorical variables with high cardinality. One that is used to fit different models on the data subsets and one to compute generalization performance across different partitions, a function that performs stepwise regression, a neural network model and the final predictions on the test data.