Taobao is a Chinese online shopping website and is one of the most visited sites globally. The website is owned by the company Alibaba Group, one of the most valuable companies in the world. In 2016, Alibaba reported a revenue of $22.96 billion - a record for the company. About 86% of that revenue came from their ecommerce platforms such as Taobao. While the company reported record-breaking revenue, the company’s profit is declining.  As such, the marketing executives of Alibaba and Taobao want to know what actions they could take to effectively minimize their expenditure to maximize profits in 2018 fiscal year. The current year is 2017, we must prepare for the upcoming year, so if the revenue generated in 2018 is higher than the revenue from 2016, then the actions taken from it would be considered a success. The executives would like to focus on the platform’s ads; they would like to know what kind of ads are the most effective in getting the user to click on their product shown by the ads. The more clicks generated by the ad would hopefully mean a higher purchasing volume for the products. Of course, more clicks may not necessarily correlate to a higher likelihood of a product being purchased and if products are sold too cheaply then profits may further decline. Thus, we need to analyze the demographics of Taobao’s user base as well to anticipate their needs with future trends. A data set containing million entries collected over an 8-day period from May 05th, 2017 to May 12th, 2017. The dataset is separated into 3 files: the first file contains data about the users of Taobao and the ad they clicked, the second file contains data about the basic information about the ads, and the third file contains information about the users.

 A rough outline of the analysis would be to begin merging the three files with the user id and ad id acting as the primary keys.

A computer screen shot of a number

Description automatically generated

From the merger of the three files, we got a total of 25,029,435 rows by 18 columns worth of data. After the data has been merged together, we continued by explore and cleansing the data, removing any row and column entries with too much missing data. Once the cleansing of the data has been done, exploration of the data was done to view the trends and patterns. First we looked at the days of the Taobao sales promotion and the distribution of the numbers of ads clicked on those days. The following graph can be seen below: A graph with different colored bars

Description automatically generated

Overall, it can be said that the ad clicks are distributed evenly, except for the day May 05th, 2017. Although that specific day could be explained by the sale starting much later in the day, giving customers less time to click on the ads. The same could be said about the last day, May 13th where sales are ending and this generating less internet traffic. We looked at the distribution of the revenue generated on those days.

A graph of different colored bars

Description automatically generated

From the revenue distribution, we can see the first day of the day generated the lowest amount of revenue by far. The low sales of that day backup the assumption the sale started later in the day and consumers didn’t have a large timeframe to view the promotions. However, the sales fluctuated throughout the promotion period, perhaps it due to the different discounts and products promotions offered on those days.

The next pattern we looked at was the shopping levels of the consumers, separated by whether they were a student or not. A hypothesis we made was of college students being more likely to click on ads, as they are more limited in their budgets and thus, would be seeking out discounts for their purchases. As the bar graph shows, college students only make up 5.4% of the total consumer base during the promotion period. They are also more likely to be a high-level shopper. Consumers are divided into 3 levels of shoppers, level 1 being the shallow users, level 2 being the moderate users, and level 3 being the depth users.

A blue circle with orange triangle and black text

Description automatically generated

A blue circle with green and orange circles with text

Description automatically generated

As seen in the pie-chart above, 93% of college students were level 3 users compared to only 84.7% of the non-college student population. We would expect high level users to more likely to click on the ads, so given the high proportion of college students level 3 users they should have a higher percentage of ad clicks compared to their non-college student counterparts. Yet, despite college students being more likely to be high level users, the rates in which college students and non-college students clicked on ads were close.

A blue and orange pie chart

Description automatically generated

Non-college students had a click rate of about 5.1% and college students had a click rate on 5.2%, while the latter does have a percentage as we hypothesized, the difference was not significant. We satisfied our curiosity and move on to the next step of preparing our data for machine learning modelling.

The initial step of the preparation involves stratifying the dataset. Stratifying refers to sub-setting our dataset into a smaller dataset while still retaining the representation ratio of our dataset. The reason why we had to do this was because the dataset has over 25 million rows of data, which would require a vast amount of computation time and resources. Thus, we reduced the amount of data to be used for our machine learning modelling to 250,295 rows of data, which is only 0.01% of the original’s size. We checked the distribution of the dates of the promotions to ensure the distributions of the sales remains intact.

A screenshot of a computer screen

Description automatically generated A graph with numbers and a number of blue bars

Description automatically generated with medium confidence

As seen on the image on the left the percentage of the distributions seems to be inline with what we saw earlier, and this can be seen with the imagine on the right. Despite the missing bar for the day May 8th due to a visual bug, we can see that the new smaller dataset is well represnted. The same could be said about the percentage of clicks as seen with the calculation below.

 

The figures on the left is the ratio of clicks to non-clicks for the orginal dataset with 25million entires, the right figure displays the ratio of clicks to non-clicks of the new dataset. The differnces for the click ratios are minute so we consider the straification to be successful.

The second part of our continued data prepping is to remove any outliars from the dataset. Plotting the price feature of the of the dataframe displays boxplot as shown below:

A graph with a line and dots

Description automatically generated with medium confidenceA screenshot of a computer screen

Description automatically generated

The boxplot shows a significant about of outliars, as we cannot even see where most of our data point lies. The figure to the right of the boxplot shows the percentiles of the price feature, as seen the mean price of a purchase is 578.73. We don’t actually know what currency the price column is uses, since the data set did not give us any background information regarding that feature, but given Taobao is a Chinese e-commerce website – the Chinese Yuan should be a good assumption for the currency used. Reguardless, lets remove some of the major outliars to prevent major skews in our data. We limited the maximum price to 2,015 yuan since that seemed like a fair upper bound most people would spend on a purchase. Capping the price to about 2,000 yuan reduced the number of data points to just under 242,000 entries. The boxplot for this updated dataframe is as shown:

A graph of a graph

Description automatically generatedA screenshot of a computer screen

Description automatically generated

We can see the boxplot more clearly now but there are still quite a bit of outliars althought it is not as significant as previously seen. From the information displayed on the figure to the right of the boxplot, the mean price has fallen from 578 yuan to 251 yuan, which is a 43.4% drop in the mean. The approach is the keep the current upper bound for the price for the moment, because lower it will reduce our data even farther.

Following the removal of our most significant outliers, the selected features undergo encoding. The most basic aspect of encoding is converting the data entry to binary representation, allowing algorithms to process them more easily. We performed what is called dummy encoding for the columns shown below:



Dummy encoding does, the aforementioned step of convert the data of a column to a binary representation of the data, but it also creates k number of columns where k is the number of categories of a feature. Using dummy encoding is not an issue if a column does not have too many categories to covert since not too many new columns to be generated; however, it does become an issue if we perform said encoding on for example the adgroup\_id column. The function will create hundreds or even thousands of new columns since each id is unique to a group, leading to high cardinality. We address this problem by using target encoding for those columns with high categories. Target encoding works by replacing the value with the mean of the target (the value we’re trying to predict, in this case ‘clk’). We performed this encoding method on the following columns:



After this step we’ve completed all of our encoding and move on to splitting our dataset into their training and testing sets. Splitting requires us to create the ‘X’ and ‘y’ variables, the ‘X’ variable contains everything except for the feature we’re trying to predict while the ‘y’ variable contains only the data for the feature we’re trying to predict (‘clk’ column). Then the splitting is performed, 70% of the data is used for training and 30% used for testing. For the machine learning modelling, we decided on three models to feed our data into: random forest classification, logistic regression, and support vector machine. The difficulty in this part of the project is tuning the hyperparameters – variables that manages the machine learning – to have the best performing algorithms. We addressed this by using randomize search to run through all the hyperparameter settings we’ve chosen and set, then it returns the best performing hyperparameters.

Starting off with the random forest classifier – expected to have the fastest computation time out of the three model we’ve chosen using – it is an ensemble of a decision trees, which predicts the target variable by splitting the data until it has reached a conclusion. The follow hyperparameters were used in the randomized search.

A computer code with red and green text

Description automatically generated

Bootstrap the process of generating data using random samples of data with replacement, so it will most likely always yield a better result; thus, we always want it to be enabled. Max features are the number of features to consider when looking for the best split, so we’ll be looking for the square root and the log2 of the number of features. Since, we don’t know which criterion of measure is the best, we’ll best testing all three: gini, entropy, and log loss. Finally, we have the min\_sample\_leaf which is the minimum number of samples required for a leaf to be considered a node. The randomized search returns the following output:

A screenshot of a computer code

Description automatically generated

We can see a min\_samples\_leaf of 5, max\_features using sqrt, and using the criterion log\_loss offers us the best performing model. When outputting the confusion matrix of the best performing hyperparameters it shows the following image:

A yellow and purple squares with numbers

Description automatically generated

We see the matrix has a good rate for true positives and true negatives, but we should check the F-score which is the measurement of the accuracy. To calculate the F-score we would need to calculate the precision and recall first, the formulas are provided below.

A black text on a white background

Description automatically generated A black text with black letters

Description automatically generated with medium confidence

Once the precison and recall have been calculated, we can use those results to calculate the F-score which is shown below.

A black text on a white background

Description automatically generated A screenshot of a computer

Description automatically generated

The highest possible F-score is 1, and the results achieved by the random forest classification model is 0.973 which is close to the maximum; thus, we are off to a good start.

The second model is logistic regression, this algorithm essentially predicts whether something happens or not, and it is one of the most popular algorithms for binary classification. Since we're determining 1's and 0's as our target label in this scenario, we believe it would be a very well-suited model to predict whether users would click on an ad. By default, logistic regression uses the 'lbfgs' solver but the sklearn documentation stated that the 'sag' and 'saga' solvers are better choices for large datasets. The documents also mentioned 'newton-cholesky' as a good option for datasets where n\_samples > n\_features, especially features that are one-hot encoded and it is limited to binary classification. Thus, we decided to choose those three as the solver hyperparameter. C parameter controls the strength of the penalty, so we want to try various metrics to get the best results. As for the max\_iter parameter, since the dataset is so large, we run into the issue of the solvers not converging. By default, the 'max\_iter' hyperparameter is set to 100 but we had to set it 15,000 to ensure it converges. The following hyperparameters to be ran through randomized search is shown below:

A computer code with numbers and letters

Description automatically generated

The best parameters for logistic regression scored the following for its f-score:

A screenshot of a computer

Description automatically generated

The f-score for logistic regression is lower than the score of the random forest classifier, which is surprisingly considering the algorithm’s affinity for binary classification problems. Perhaps there are more hyperparameters to fine-tune, something to consider later.

The final algorithm to run is the support vector classifier, like with the logistic regression algorithm we want to test out various C metrics to see how the penalty affects the performance. Gamma determines the amount of influence a data sample exerts, so it affects the curve of the decision boundary. The kernel, by default uses ‘rbf’ but since ‘rbf’ has a high computation time we decided to use ‘linear’ instead which is often used in datasets with large features.

A screen shot of a computer code

Description automatically generated

From the randomized search, the results showed the following as the best parameters and the f-score:

 A screenshot of a computer

Description automatically generated

With an f1\_score of 0.938 it performed better than the logistic regression model, but worse than the random forest classification model.

During this research problem, we ran into numerous issues with figuring out the meaning behind the data. Features such as the cms\_segid, cms\_group\_id lacked any information on the meaning behind them and age\_level lacked any context behind their categorization. In the end, almost half of the features were not useful because we had no context behind them to make conclusions on. A significant issue is also the computation time and resources needed to run these models, to run the three models chosen at the current 250,000 rows of data we have it took 6 hours to finish. Ideally, we would want to use all 25 million rows of data but that is not feasible currently. As for further research, we would like to provide the random forest classifier with more data – maybe all 25 million entries – since it ran the fastest and see how the results would be. For logistic regression, perhaps the scaling performed affected the performance of the algorithm so maybe changing the standardization to before the test split might yield a better performance. An article about SVM stated gamma is not required if one does not use the default ‘rbf’ kernel, since we used the ‘linear’ kernel, we would like to test out whether removing the hyperparameter from our randomized search would yield better results. Overall, the results we got are good, but allowing the use of a higher portion of the total data would be even better.

The recommendations for the Taobao executives would be to use the random forest classification and take all of the information they have on their users to predict whether they would click on the ad. The project only considers a small sample of dataset that only recorded the promotional period of the ecommerce platform.