Predicting E-Commerce Site Visitors' Behavior

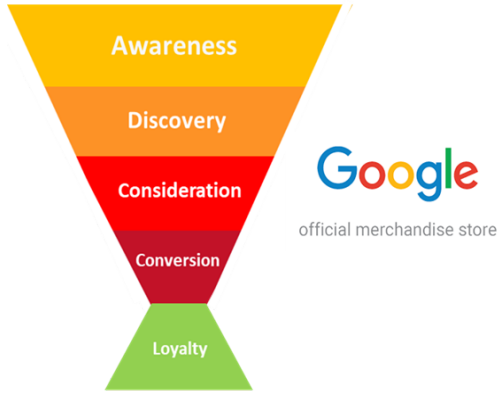
Tia Burton, Fred Gigou, and Idris Hanafi.
May 30th, 2022

1. Project Overview

E-commerce is the primary driver of most companies' growth, representing close to 15% of retail sales in Q1 2022, according to estimates from the Monthly Retail Trade Survey and administrative records. Initially driven by tech giants such as Amazon, brick and mortar companies in the US including Walmart, Costco, or Kroger's are making considerable investments to catch up. Therefore it is critical for us as data scientists to apply our knowledge to bring value to the field. Lastly, both Tia and Idris are former or new Amazonians and Fred is passionate about digital marketing, so there is a strong motivation to learn from each other and bring expertise.

We want to uncover the digital conversion funnel as shown in figure 1, and answer critical business questions with data science solutions to drive conversions. In this context, we have chosen actual data from the Google Merchandise store and want to answer the following questions:

Figure 1. Digital Conversion Funnel



Visitors clustering:
- Can we segment our visitors across the entire digital funnel?
- From this clustering, what insights can we draw to drive conversions in the medium term?
- Are there channels more likely to drive traffic and sales?

Buyers classification (consideration and conversion):
- Can we predict which visitors are the most likely to buy?
- Based on this likelihood, who should we target to drive conversions?

Returning customers (loyalty):
- Which customers are most likely to return?
- Therefore, who should we retarget?

2. Data Source

Data Origin: The Google Analytics 360 dataset is stored in the public datasets section of Google Big Query. Google Big Query is a cloud-based product used for data warehousing, querying, and analyzing. The particular, public dataset includes visits and transactions from the online Google merchandise store; it is partitioned by date, so each Big Query table holds visits, interactions, and transactions for one day. We collectively decided that it was necessary to extract the data and concatenate the files to continue to perform exploratory data analysis and modeling. We use the Big Query API client to query all rows and columns from the table using the following format, `bigquery-public-data.google_analytics_sample.ga_sessions_YYYYMMDD`, where YYYMMDDD represents the year, month, and date of the observed Google Analytics sessions.

Data Retrieval: After retrieving the results, we use the PyArrow API to convert the query result table into a Pandas dataframe. The files are stored as monthly CSV files in our Shared Google Drive where they are available for further processing with Python scripts in Google Colab. The

aforementioned process is considered the first step in our machine learning pipeline; this is our data extraction step (see data_extraction.py). There was a full year of data ending on August 01, 2017, but the team decided the first half of the year would be sufficient for training, validating, and testing models. After 3 hours, where each month takes 30 minutes to query, concatenate, and download, all records from August 01, 2016, to January 31, 2017, are available. In Google Colab, there are other efficient data extraction tools, like Spark through BigQuery without extraction, but these tools are paid features of the technology (https://cloud.google.com/solutions/spark). We continued to use open-source tooling so as not to incur any costs.

Data Storage: The Big Query tables were in columnar format. To make them more accessible for the team, we concatenated the PyArrow results and stored them in separate CSV files. The monthly files are hosted in our shared Drive in Google Drive and are used in the succeeding pipeline components through Google Colab and mounting the data source.

Data Preparation: Both supervised and unsupervised models in this project had similar features for analysis and machine learning. The dataset initially contained 469 columns once expanded and 501,093 rows, which summed to about 12.57GB in size. We reduced the initial dataset with various techniques, of which some are described below, including downsampling. Key columns to this dataset included: date, social engagement type, device, totals, geo network, traffic, and hits. We extracted values from a JSON string, or sometimes lists of JSON strings, from these columns to gain more general information about visitors that was typically available, like country and device category. The dataset used in the models varied in size due to additional transformations. (Appendix B Table B1)

Columns Filtering: We used the google documentation, stats criteria (e.g. just Nan values) or duplication in other features to determine columns to drop. In *appendix A*, we provide a summary of each specific case and rationale.

Downsampling: After our first initial filter, we were still left off with ~10GB+ worth of data which isn't within our computing capability in Colab. Given that our interest group (purchasers) follow the average conversion rate of 1.5%, we aligned on preserving all ~5000 transactions (501,093 * 0.015) within our dataset and randomly sample 10% of the non-transaction. This allowed our dataset to be reduced to 2.46 GB (55,382 rows), which is within our computing capabilities in Colab.

Features Additions: For the purpose of the classification tasks and/or clustering analysis, we added 4 more features: buyers and repurchasers (target classes), frequency of visits over the last 6 months and recency, as the days since the last visits.

Categorical Encoding: Non-null categorical features are numerous in our data set, so they represent a serious challenge for the project, especially given that some of them present high cardinality. Therefore, we used the following strategies depending on the types of features:

- **Binary encoding** for straightforward cases, e.g., isDirectIsTrue.
- **One-hot encoding** for low-cardinality features, for instance, channel source or device category.
- **Binning** with ordinal values. In the case of "hour" of visit, for instance, instead of using 24 different slots, visualization reveals a clear pattern with three bins: low, medium, and peak hours.
- **The weight of evidence algorithm (WOE)** for high cardinality features. For instance, in countries of origin, while some buying patterns may exist, it would not make sense to hot-encode 179 unique values. The WOE [= ln (% of non-events / % of events)] tells the predictive power of an independent variable (in this case, buying) in relation to the independent variable. Advantages include achieving a single numerical column usable for linear or logistic regressions with the handling of Nans and outliers (Bhalia, 2015).
  Lastly, we kept a dataset with the original, non-encoded categorical features to compare models performance whenever possible, such as for decision trees, or the FAMD (Factorial Analysis of Mixed Data).

3. EDA: Initial Data Findings

Figure 1. Pandas Profiling Missing Values Dendrogram

Pandas Profiling: For the exploratory data analysis, we used the Pandas Profiling (PP) package available via PyPi (Brugman, 2019). It is a package that automatically analyzes the input data and builds a summary report either into a notebook or an html file; it was designed to streamline this portion of the data science process. We generated an html report in less than 2 minutes on ~50,000 rows of data. The html report included cardinality, correlations, and insights about columns with zeros, constant, missing, or null values. The highly cardinal columns included *geoNetwork.country* and *trafficSource.source* while most columns derived from the original *totals, trafficSource, and device* columns were highly correlated with one another. This package also generates a missing values dendrogram, or a graphical representation of the hierarchical agglomerative clustering (HAC) of the missing values in the dataset; HAC is " non-parametric—assume little about data, natural and simple in grouping objects, and capable of finding clusters of different shapes by using different similarity measures" (Dash et al., 2003). One drawback is that this dashboard does not explicitly state which distance measure is used in the algorithm, but estimating with the version provided by PP under the assumption that it is using the Euclidean distance, there are two distinct column clusters.

Correlation Finding: Another informative section was the correlation section. We could quickly observe several correlation coefficients, such as Pearson correlation or Spearman's rank (see Appendix B Figure B3). Both revealed that columns retrieved from the same JSON column were highly correlated. This influenced a strategy of varying the number of columns based on various dimensionality reduction techniques. Lastly, PP computes a new correlation coefficient, Phik (φk), and the number of columns with valid correlations increases as this coefficient has no assumptions of linearity or monotonicity (Baak et al., 2019). This method shows correlations between categorical, ordinal, and interval variables and returns non linear dependencies (see Appendix B Figure B2); in other cases, it refers back to Pearson correlation. The chart suggested strong correlations between device and channel grouping as well as device and traffic source. All PP reports for all modeling datasets and the cleaned dataset are found on Google Drive (Appendix B Table B1).

4. Part A: Supervised Learning.

4.1. Likelihood to buy (A1).

Methods & Evaluation: In the a1_b2_dataset file, we collectively decided to transform most variables into numerical values with one-hot encoding or weight of evidence. As a final transformation step for this model, features were removed from the dataset with the following two methods: Phik (φk) and Mutual Information. In the *Phik* (φk) correlation chart, the date column doesn't have a strong relationship with other features, so it was removed (see Appendix B Figure B2). Mutual information scores (see Appendix C), filtered at threshold values of 0.0, 0.001, and 0.05, produced datasets with 31, 27, and 6 columns respectively and were used as filters within multiple iterative cross validation pipeline. Lastly, the *totals_transactions* was transformed from a variable with 8 categories, from 0 to 7, to a binary categorical label. The column was representative of the total purchases a user made in a single session, but the model required a binary label. 0 represents the visitors, and 1 represents the buyers. Lastly, the datasets were named after the sampling method; each has balanced or unbalanced classes. The 'X_train_dsample' represents downsampling to equal classes; the 'X_train_unbalanced' and 'X_train_random' unbalanced samples are of size 50,000 and 5,000 respectively.

All transformed datasets were tested with PySpark ML models and evaluated mostly with scikit-learn. The library has a limited number of metrics implemented for binary classification and is currently running in Google Colab, an environment that has only one CPU core; however, the models have a better opportunity to scale with the Apache Spark based framework than other packages. The models were evaluated with the area under the Receiving Operating Characteristic curve (AUC) from PySpark ML and precision, recall, and f1-score (see Appendix Formulas F2-4) from scikit learn (Buitinck, 2013). While AUC is a measure that generally works well with binary classification, the F1-Score helps compare between different classifiers. Recall and precision metrics were included to supply all components of the F1-Score because the metric is "a weighted harmonic mean of the precision and recall" of a model's inference (Buitinck, 2013).

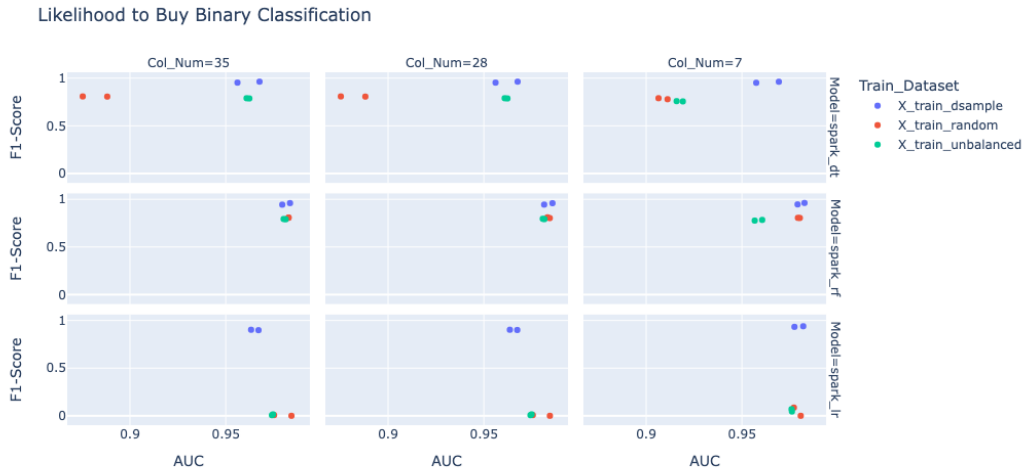


Figure 3. Likelihood to Buy Binary Classification Results from Decision Tree, Random Forest, and Logistic Regression Models; plotted with Plotly Express.

Tradeoffs: For this prediction task, online store visitors were modeled with logistic regression, decision tree classifier, and random forest classifier models. In PySpark, there isn't an implementation of a dummy, or benchmark, classifier. The results of the tree based classifiers are compared to that of the logistic regression model. Nonetheless, the model group increases in bias as they increase in complexity. Decision trees and random forest classifiers are non-parameter, consider individual features with criterion entropy (Appendix Formula F6) or gini (Appendix Formula F5) to split the data, and don't add significance to those numbers with assumptions (Breiman, 1996) while logistic regression models have four assumptions (Stoltzfus, 2011), and problems still may arise after validating the assumptions. In the failure analysis section, we will discuss and analyze the F1-Score of the resulting models in this context.

Performance: Model evaluations varied based on the evaluation metric. With the AUC metric, most mosts performed above 85% across all iterations (see Figure 3), so this metric was not as informative. Instead, we utilize the f1-score. Varying the column size with mutual information did not cause major variance between the models' f1-scores, yet the models were most affected by the balance of the classes in the dataset. Of all our classifiers, the random forest model performed the best based on the mean F1- (.98) and AUC (.85) scores. The logistic regression, though it is acting as our benchmark model, performed well only on one dataset, the X_train_dsample, with mean AUC of .97 and F1-Score of .91; the unbalanced had a mean AUC of .98 and F1-Score of .02 (Appendix D).

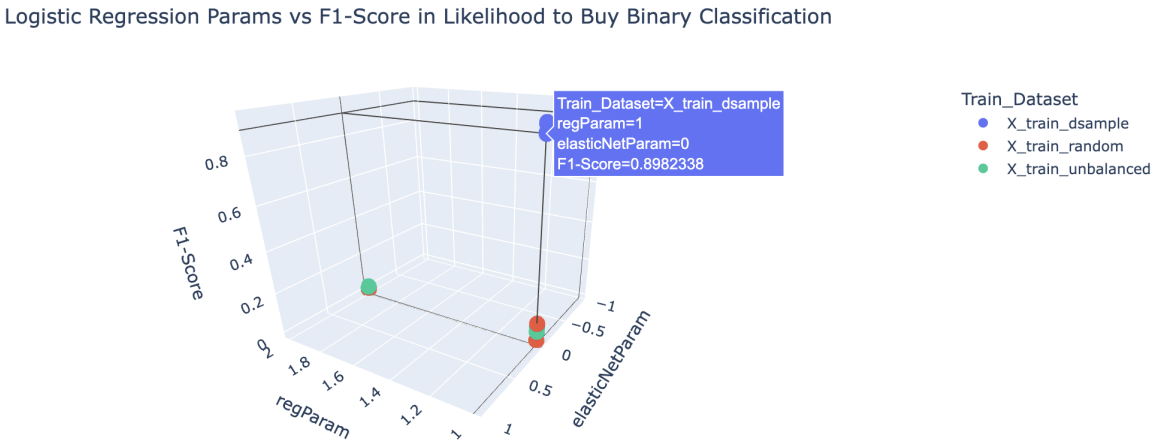


Figure 4. Created with Plotly Express.

Failure Analysis Logistic Regression: Under the F1-Score, the logistic regression model performed worse than the other models. In Figure 4, the F1-Score is mapped against the Elastic Net Parameter and the regularization parameter. Despite the varying both in the cross validation stage, the logistic regression is most affected by the class imbalance and the . This is firstly demonstrated with Figure 4; the F1-Scores chart (Appendix D Figure D1), comprised of recall (Appendix D Figure D3) and precision scores (Appendix D Figure D2), displays the greater than 90% difference between balanced dataset and the remaining unbalanced two.

4.2. Likelihood to repurchase (A2).

Overview: The goal of this analysis is to identify whether a given user page behavior, would the user be categorized as a potential repurchaser (i.e., a returning customer)? This question can only be answered by analyzing users with the following identity:

- **Group A**: All users who purchased once.
- **Group B**: All users who purchased more than once.

Input Dataset and Models: The input dataset also followed the standard feature encoding mentioned in section 2 and was generated during the data pipeline. However, the main target class for this business question is **Group B**. For this analysis, we delved into 5 different default models (Logistic Regression, SVC, KNN, Decision Trees, and Random Forests) and compared it to the Dummy Classifiers of 3 types (Stratified, Uniform, and Frequent).

Feature Correlation: The first question we encountered is, out of the 44 features, which should we use as an input feature? We answered this question by exploring the Pearson correlation across the different features when compared to the target class “repurchaser” (see Appendix E Figure 1). Many of the features were close to 0 (i.e., no correlation), so for our initial analysis, we decided to start off by using all of the features except for visitor unique ID and Date, so now we only have 42 features.

Default Model vs Fined-tuned Models: With our default models, we identified that the Random Forests and Decision Trees tied in first averaging around approximately 81% ROC AUC Score and Logistic Regression going in second (see Appendix E Figure 2). Surprisingly, SVC and KNN achieved accuracies *similar* to the Dummy Classifiers. To verify that these results are consistent across our training and test datasets, we used a 5 fold to confirm that our accuracy isn’t biased from the dataset (see Appendix E Figure 3). Our next approach is to see whether we can improve our accuracy by **hyperparameter tuning via Grid Search**. Using 5 folds and a range of learning rates, solvers, regularizations, weights, etc., we found that the F1 Score and ROC AUC Scores weren’t improving by much; scores were increasing by <1% (see Appendix E Figure 4). This led us to believe that the models didn’t need to be optimized and that maybe it’s the dataset.

Target Class Rebalancing: Our first attempt on the dataset is to tackle the imbalance between targets vs non-targets classes. Our target class originally comprised ~15% of the dataset (see Appendix E Figure 5). One of the main problems with imbalanced classification is that there are too few examples of the minority class for a model to effectively learn the decision boundaries. We applied **(Synthetic Minority Oversampling Technique) SMOTE** to our target classes, which synthesizes new data points for the minority classes by using KNN to extract new points in the given feature space (Chawla et al., 2002). Using our newly rebalanced dataset and 5 folds, we finally achieved accuracy scores (F1 and ROC AUC) greater than 90% and optimized the SVC and KNN (both receiving >80% accuracy scores) to perform better than the dummy classifiers (Figure A2.1).

Figure A2.1. ROC-AUC Scores and F1 Scores of the models after finetuning and rebalancing the dataset.

Model Tradeoffs: The runtime for Decision Trees is many-folds faster to train compared to the other models and yet achieves among the highest accuracies. Part of this is because during the grid search for optimal parameters, it identified the optimal max depth at 10 layers. Even when compared to setting no max depth, a pre-pruned tree of a max depth of 10 layers continuously achieves greater accuracy as it learns enough about the dataset without overfitting.

Failure Analysis SVC: Referring back to (see Appendix E Figure 2), we originally received scores close to the dummy classifiers when looking at SVC and KNN. After rebalancing the dataset, they both were able to achieve accuracies greater than 80%. SVC is based on the Support Vector Machine (SVM) algorithm which is effective for balanced classifications. Generally, the SVM algorithm finds a hyperplane decision boundary that best splits the two classes and by default favors the majority class, hence low accuracy scores when training on the original imbalanced dataset. Another way to tackle this would be to add weights to our SVM (i.e. cost-sensitive SVM) so that it penalizes the majority class.

Failure Analysis KNN: However, for KNN, it's interesting that after rebalancing the dataset, the accuracies are a lot higher as it wasn't as clear as to why that was the case. The intuition that we derived from is the following: Default KNN uses a Minkowski distance with a p-value of 2. This means that the points in the space will be using the euclidean distance algorithm (see Appendix F Formula F1). In our scenario, the inductive bias for KNN is that the input dataset can be partitioned into two disjoint subspaces (repurchasers and non-repurchasers). Since our original training set is imbalanced, therefore, the probability of the euclidean distance picking any of the k nearest neighbors, of any random selected query point, in the subspace will be misclassified as the majority class is higher. Similar to SVC, we can also tackle this with weighting the neighbors by their distances.

Feature Pruning: We concluded this exploration by analyzing the features and seeing which features are deemed more important to our models. We wanted to further explore Decision Trees as it had the fastest train time as well as the highest accuracy. Recall, we were originally training on 42 features. SKLearn Decision Tree Classifiers come with a handy attribute called “feature_imporances_”, which returns the quantified importance of a feature. In this case, Decision trees use the “Gini Importance” (or “mean decrease impurity”) and is defined as the total decrease in node impurity averaged over all trees of the ensemble, where values closer to 1 are important and 0 least important.

Feature Pruning with Decision Trees Results and Conclusion: Inspecting the important features (see Appendix E Chart 1), there are a lot of features that have close to 0 importance. So we pruned all features with gini importance values of less than 0.01 and tested out to see how well our Decision Tree model can predict without those features. We came to find that pruning these features didn't affect the accuracy (see Appendix E Figure 7), leading us to conclude that those pruned features are irrelevant. This led us to believe that the top driving channels to determine potential repurchasers are users who have the qualities of: high frequency, high purchasers, come from an organic search or direct links, high time on site, and high number of pages. So if a new ad campaign were to consult us for finding potential returning customers, we will be monitoring those metrics for a user to assist in this business campaign. We then visualized the decision tree boundaries with a max depth of 5 as seen in Figure A2.2.

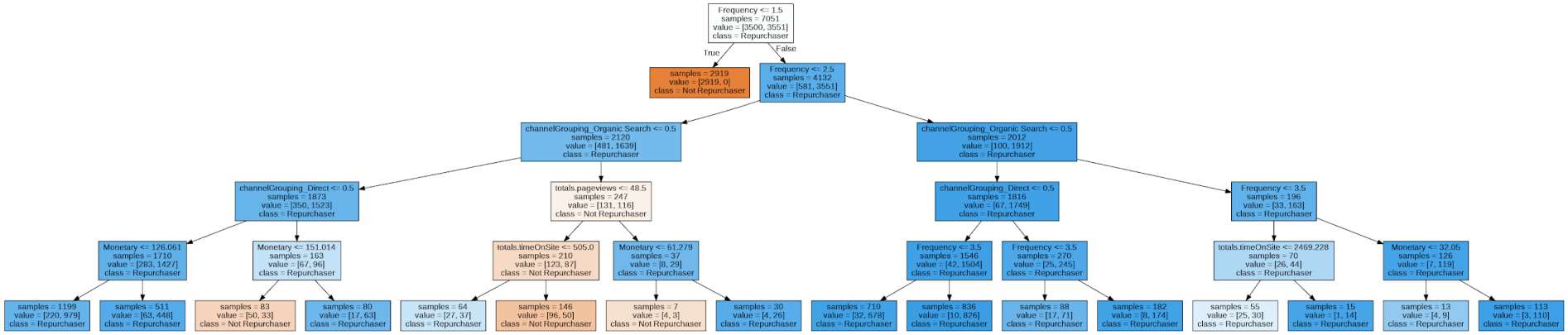


Figure A2.2. Decision Tree Visualized with a max depth of 5.

5. Part B: Unsupervised Learning: Visitors clustering (B1 & B2)

5.1. Workflow Steps

Step 1: We took the cleaned dataset as input from the earlier pipeline step. One is numeric with categorical encoding, and the other includes mixed data (numeric plus the original categorical features).
Step 2: We performed categorical encoding as described in section 2 – data clean up.
Step 3: We rebalanced the dataset using the SMOTE library as described in section 4.2 as buyers only represented a fraction of the data set. As a result, our data set moved from 55'381 rows to 99'040 (an addition of 43'659 synthetic transactions) with 50% buyers and non-buyers.However we run our models with the balanced and imbalanced data sets to understand the impact on our results, actually achieving better scores with the rebalanced data set.
Step 4: We scaled the numeric data and concatenated it with the categorical, encoded data for the numeric data frame (df). Finally, we joined the numeric data with unencoded categories for the mixed df.
Step 5: We performed PCA (2 components) on the numeric df.
Step 6: We performed the K-Means algorithm on the balanced and unbalanced dataset with and without PCA.
Step 7: Besides PCA, we evaluated the Factor Analysis of Mixed Data (FAMD) dimension reduction technique on the mixed data set. This technique provided valuable visualization insights (Figure B2.6), although the documentation states being cautious on the predictive analysis power.
Step 8: We assessed additional clustering methods on the PCA balanced dataset, including K-Medoids, which is better at representing specific points, and DBSCAN, which may help with certain patterns with outliers.
Step 9: We built powerful visualizations to summarize results, insights, and conclude.

5.2. Parameters Tuning

For PCA: we used the components explained variance (98.9% with 2 PCs) with the spree chart, biplot chart, and heatmap for interpretation.
For the K-means: We used the elbow analysis and several clustering metrics to get optimal K (Figures B2.1 & B2.2).
For the K-Medoids and DBscan: We used the same metrics as for Kmeans in addition to visualizations.
For all techniques, we used tables and visualizations to interpret each clustering result.

5.3. Challenges and Solutions

Features encoding: Google data included many categorical data, some with high cardinality (for instance, countries); therefore, we had to research new techniques, besides one-hot encodings, such as the weight of evidence (2) that has the benefit of reflecting class balance while keeping features compact and numeric.
Class imbalance: Buyers only represented 1% of the original data. Therefore, we gathered all the transactions into the dataset and applied the SMOTE technique for categorical balance.
DBScan tuning: Parameters (eps and min sample) are hard to tune, while each iteration is computationally expensive. Therefore, we applied SIADS 543 dbscan fine-tune best practices by using the KNN method to get an estimated eps and then iterated over a closer interval of eps and a minimum arbitrary sample size to get a better solution.

5.4. Model Recommendation and Key Metrics: PCA with K-means k=5.

Key metrics:

Based on a mix of metrics (Figure B2.1), elbow analysis (Figure B2.2), and visualization (Figure B2.3), we determined K=5 to be an excellent solution to our customer clustering problem. The SIADS 543 class indicates that a robust clustering solution relies on combining a high silhouette, lower Bouldin-Davis metrics, and visualizing the elbow or inflection point by plotting the within clusters sum of squares (WCCS) called inertia in the Scikit learn package.

We can also observe that the PCA balanced data set that included balanced classes presented a more convex, curved shape (in Green) than the other dataset options, namely “All feat” which provides for all the features, imbalanced class, and no PCA.

Lastly, we also performed RFM K-Means clustering (Recency, Frequency, Monetary) on buyers as we initially established in our project description but ended up with worse scores, achieving 0.50 in Silhouette scores and 0.82 Bouldin Davies for K=4.

Figure B2.2: elbow analysis by data frame

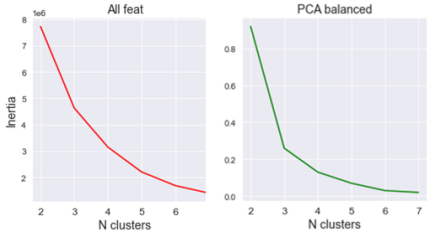


Figure B2.1: scores, balanced class PCA data

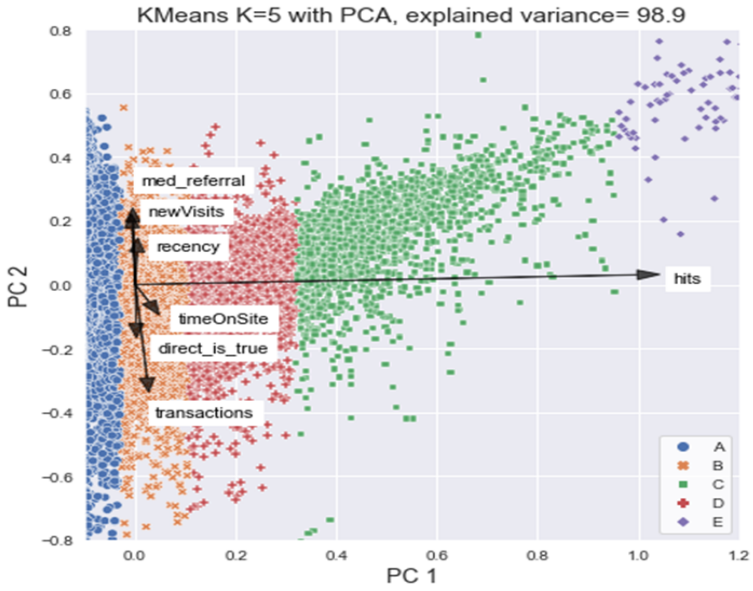
k	WCCS	Silhouette	Bouldin-Davies	Calinski-Harabasz
2	0.92	0.83	0.11	41
3	0.26	0.74	0.3	121
4	0.12	0.71	0.33	180
5	0.06	0.72	0.31	272
6	0.03	0.72	0.3	458
7	0.02	0.75	0.28	622

Clustering visualization:

Figure B2.3 is a very eloquent graph where we visualize the conversion funnel horizontally. The latter means that on the left, the “A- Blue cluster” tends to be fleeting visitors, whereas we move down the different conversion stages being” D- green” high interaction, buying sessions to returning customers “E-purple.”.

Biplot: While we have more than 40 columns, many dimensions are highly correlated. Figure B2.3 plots some of the most discriminating components as the number of hits (or interactions), the #1 driver of principal component 1 (x-axis). On the y axis, PC2, monetary, time on site, and direct true (direct traffic) are correlated while, as opposed to the factors on the same axis, are recency, referral source, and new visits.

Figure B2.3: visitors clustering visualization



5.5. Clusters Interpretation and Insights

General pattern: Our clustering approach reflects the stages and shape of a conversion funnel drawn horizontally. Therefore, a close look at each dimension in Figure B2.5 and selected original mean values in Table B2.4 gives us great insights for characterizing each cluster. Please also note that this representation refers to the rebalanced dataset that includes synthetic transactions, therefore we need to focus on patterns rather than precise numbers.

Cluster A: "Clickers & bouncers". These users account for most of the traffic but currently represent a small economic value. They come from mobile social networks but interact little (3 page views). Critical implications for management are generating a better inbound strategy with paid media strategy, relevant site sections with right key words and channels (paid search, referral, desktop platform) and lastly, reviewing the mobile site optimization.

Cluster B: "Browsers, I want to know". This cluster is the second largest and presents economic potential. They show interest, are recent, and interact significantly with the site from a Mac but buy little. Our reco is to retarget these users with proper digital advertising stimuli to drive them down the funnel.

Cluster C: "Trialists, I want to buy". This is the third-largest group, and they do have economic potential as they have already bought a small amount recently. Critical strategies for these users can be on-site cross-selling to increase transactions and retargeting to get them to return to the site.

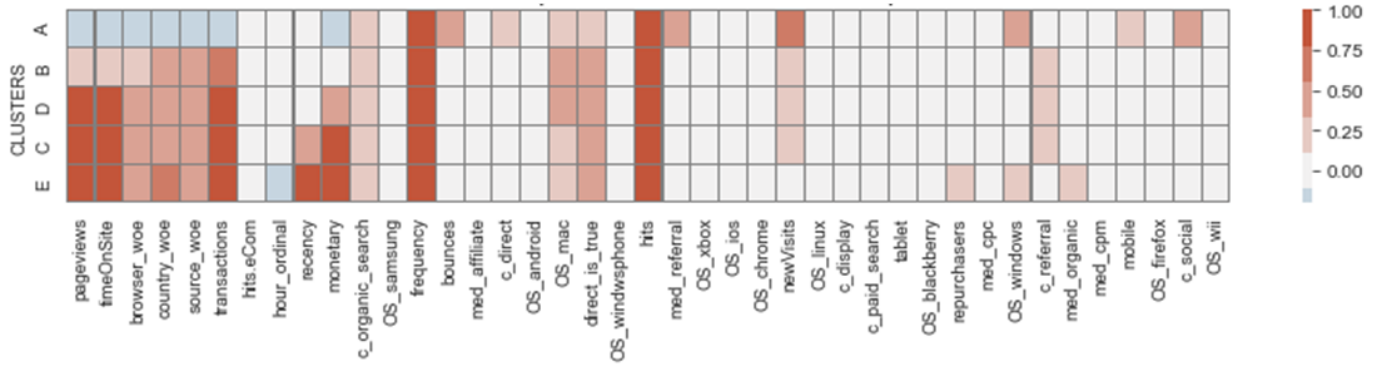
Cluster D: "Lagged, heavy buyers". They are heavy users who have repeated. However, they are lagging (113 days since the last visit). The fundamental strategy is to retarget by e-mail, or other digital advertising means with promos, releases, and promotions to make them come back.

Cluster E: "Loyalists". Although this cluster is the smallest, people interact and buy a lot. However, high average recency (135 days since the last visit) and disproportionate page views relative to number of transactions (2.1) suggest they are lagged users and that our site needs optimization for conversion. Key marketing strategies should include reducing friction in the checkout funnel and creating a loyalty program to cultivate these high-value customers.

Table B2.4: Count of clusters and mean values for selected dimensions

Clusters	Count	hits	page views	time OnSite	Transactions	new Visits	Bounces	Monetary	Frequency	Recency	organic search	Mobile
A- blue	51,550	4	3	98	0.1	0.74	0.48	$9	1.2	91	0.3	0.17
B- orange	30,800	26	22	799	0.9	0.24	0	$95	1.4	91	0.2	0.03
C- red	13,691	57	43	1539	1.0	0.22	0	$180	1.5	97	0.1	0.01
D- green	2,842	115	82	2797	1.2	0.18	0	$351	1.8	113	0.1	0.01
E- purple	157	321	238	6570	2.3	0.06	0	$1,155	2.1	135	0.3	0.03

Figure B2.5: Cluster interpretation heatmap



5.6. Additional Models Assessed

Factorial Analysis of mixed data:
We used FAMD as it is a powerful visual technique for analyzing mixed data. In Figure B2.6, we are sharing a couple of insights:

1) The general plot shape is similar to the one with PCA. We should expect the latter as FAMD belongs to the PCA family. FAMD first hot-encodes categorical variables and then divides them by the square root of the proportion of objects in the column and then centers. Lastly, the PCA algorithm is executed on the resulting matrix to obtain the final output (Keany, 2021).
2) Visual exploration provides a wealth of insights on what dimensions convert, for example, by channels where we can see that social networks or organic search do not convert, while referral tends to do so. However, given the intrinsic one-hot encoding of the technique, you need more than a 100 components to represent a significant explained variance, hence we decided to only use this technique for qualitative / visualization purposes.

KMedoids:
We used KMedoids because this method better represents specific points (real median point) than the means (in K-Means) that outliers may distort. As a result, in Figure B2.7, we can observe more squashed groups to the left with converters agglomerated into one category (orange). However, the method achieved less favorable scores than K-Means at its optimal level k=5, namely silhouette (0.61 vs 0.71 in K-Means) and Bouldin Davies (0.57 vs 0.32).

DBscan:
We also performed the DBSCAN method for comparison and experimentation purposes, although resulting in poor performance in key metrics . DBSCAN is a clustering method used in machine learning that separates high-density clusters from low-density clusters. Unfortunately, the technique struggles typically with groups of similar density (Lutins, 2017), which in our case, results in poor performance visually (Figure B2.8) and metrics (Silhouette score of 0.1 in the best scenario).

Figures B2.6: Factorial Analysis of Mixed Data

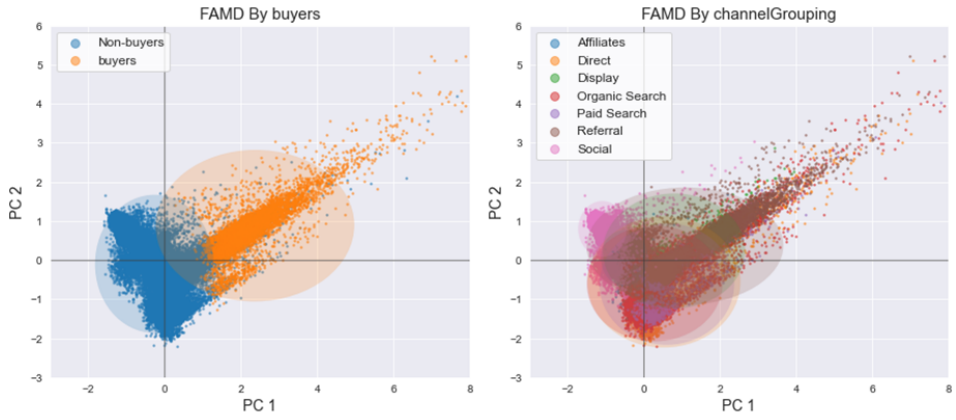
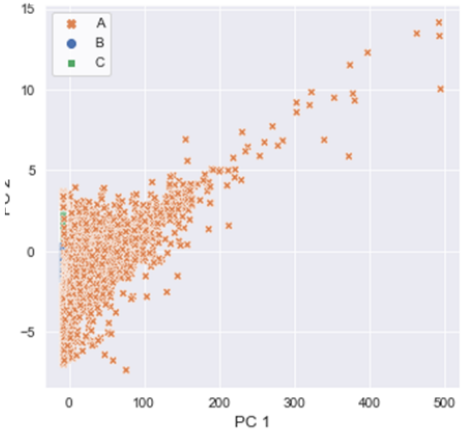


Figure B2.7: K-Medoids clustering

Figure B2.8: DBSCAN clustering



6. Discussion

6.1. Supervised Approach Learnings and Next Steps

In **Part A1**, in terms of the choices made before the model is trained, using SMOTE might provide quicker results that would be useful for real world marketing applications. At this time, all the models take about 15 minutes to run. Undoubtable, here would be one model running for any given API call if the model was productionized . Provided more time, we would apply big data implementations, similar to the scalable version written in scala for neurocomputing (Juez-Gil et al., 2021). We would implement the Approximate-SMOTE technique and leverage our existing pipeline measures, AUC and F1-Score.

In **Part A2**, feature selection is important. A lot of the features that we originally thought would help the model learn turned out to be “useless”. Unfortunately, the GA 360 dataset did not include any campaigning information nor a more granular level of user information. So the findings in this section are only as good as a high level information of the user web page behavior.

Next Steps: Given more time, we would look into other datasets and try to join user demographics and other information to assist in creating an online classification model to help ad agencies with their campaigns. Among these other information we look for are eligibility to convert, for example, some products can prevent groups from certain countries from purchasing as it may be illegal to purchase in said country, or it can simply be a high cost of shipping. Lastly, the conclusions we came up with doesn’t answer the possibility of selection bias. The dataset doesn’t include many pretreatment covariates and confounders that can potentially affect the conversion outcome, (user wealth, innate interests, etc.). Our next step in this aspect would be to approach this by pulling more data from the GA360 dataset and conducting random assignments to the respective interest groups (conversions and repurchasers). By the law of large numbers, the random assignments will converge on an average and eliminate the observed and unobserved differences between the groups.

6.2. Unsupervised Approach Learnings and Next Steps

Features and classes: We learned some of the challenges in managing many categorical features and imbalanced classes and applied various techniques available to cope with these situations.

There’s no one size fits all measure to assess the goodness of clustering: Existing measures have certain limitations in different application scenarios (Liu et al., 2013, p. 1). At this point, we used four metrics to assess clustering (WCCS, Boulding-Davies, Silhouette, Calinsky Harabasz) and explored SDbw for DBSCAN after reading the referenced paper. Still, our learning is that clustering is both science and art that requires metrics and domain expertise to find an acceptable solution.

Interpretation needs domain expertise: As a person with a marketing background, I found it particularly rewarding to characterize clusters, an element that requires domain expertise to leverage findings. For example, I was amazed to visualize the digital conversion funnel through the clustering technique.

Additional work and resources: Data and site exploration suggest that the Google merchandise store is not such a commercially active site. There are no paid traffic tactics, limited cross-selling, and no loyalty programs. Therefore, the latter limited the extent of the analysis. I would love to work on a site with more commercial activity to build more advanced algorithms and recommendations for future projects.

6.3. Ethical Considerations

Supervised Learning Ethical Issues: The dataset consists of the purchases and visits to an online store. For any retailer, customer data is restricted and deemed extremely valuable; the data, however, in most cases still belongs to the customer it represents. It would be unethical to include personally identifiable information or demographic data. Google assigns each user a client id and removes most visitor information, like names, ip addresses, physical addresses, and payment information. Lastly, we removed the client id during our model training phase and focused more on the features that were based on interactions with the virtual store, user location, or user device specifications. Removing the features reduces the risk of the data being traced back to a single visitor which uploads their right to privacy.

Unsupervised Learning Ethical Issues: Despite our explanations on part A, our model trained on the selected features still has the potential to be biased. In fact, one of the reflections regards the weight of the evidence encoding that's based on the propensity of a group to be in the target class. Although it is widely used in insurance and credit risks, this technique might introduce a bias based on origin or belonging to certain groups. Using this algorithm indiscriminatly might induce a model to predict behavior, acceptance, or denial of a service based on the social, economic, or protected group a person belongs to, therefore causing potential harm. For instance, an algorithm could use these "encoded" features to predict the likelihood of paying a bank loan or accessing a particular service such as lodging. In conclusion, as data scientists, we need to understand and exert with caution any algorithm we are using, and communicate to management about these risks. We are also compelled to apply preventive techniques such as the what if tools, check lists or external audits in combination with a more corporate approach including a code of conduct.

7. Statement of Work

We've organized our workflow in Google Drive, GitHub, and Google Colab. We orchestrated our initial data and model pipeline in a Colab notebook where we served it as a CLI to run our python files for each of our pipeline steps. After every meeting and status update, we made best intentions to push our updates to GitHub for version control. For our data exploration and modeling, we used Colab notebooks and aligned on the content to be migrated to python files for our pipeline. In the end, we compiled the steps into a Makefile and python files to recreate our analysis.

Overall, everyone in the team participated and worked very closely in the alignment of the dataset, data encodings, data pipeline, model choosing, model pipeline, and project discussions. Last but not least, we want to thank our project coach, Coco, for assisting us and always being ready to respond to any of our questions, and the MADS staff and professors for helping us along the way.

Tia: Data source, EDA, Section 4.1 - supervised machine learning classifier for likelihood to buy, discussion, pipeline contribution.

Idris: Data transformations, feature exploration, pipeline standardization, project management & organization, and Section 4.2 - supervised learning for repurchaser classification.

Fred: Part B unsupervised learning, feature encoding, project definition, and motivation.

References

Baak, M., Koopman, R., Snoek, H., & Klous, S. (2019). A new correlation coefficient between categorical, ordinal and interval variables with Pearson characteristics. In *Phi_K Correlation Analyzer Library*. Retrieved May 26, 2022, from https://arxiv.org/abs/1811.11440

Bhalla, D. (2015). N.p.: Listen Data. Retrieved from https://www.listendata.com/2015/03/weight-of-evidence-woe-and-information.html

Bewick, V., Cheek, L., & Ball, J. (2005). Statistics review 14: Logistic regression. Critical care (London, England), 9(1), 112–118. https://doi.org/10.1186/cc3045

Breiman, L. Technical Note: Some Properties of Splitting Criteria. Machine Learning 24, 41–47 (1996). https://doi.org/10.1023/A:1018094028462

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Müller, A. C., Grisel, O., ... Varoquaux, G. (2013, September 1). API design for machine learning software: experiences from the scikit-learn project. European Conference on Machine Learning and Principles and Practices of Knowledge Discovery in Databases, 15. doi:https://doi.org/10.48550/arXiv.1309.0238

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16, 321-357. Retrieved from https://arxiv.org/abs/1106.1813

Dash, Manoranjan, et al. "Fast hierarchical clustering and its validation." Data & Knowledge Engineering, 44(1), Jan. 2003, pp. 109-38. ScienceDirect, doi:https://doi.org/10.1016/S0169-023X(02)00138-6. Accessed 26 May 2022.

Fleuret, F. (2004, November 4). Fast Binary Feature Selection with Conditional Mutual Information. Journal of Machine Learning Research, 5(9), 1530-1555. Retrieved from https://www.jmlr.org/papers/volume5/fleuret04a/fleuret04a.pdf

Google Merchandise Store (2017). Google Analytics 360 data, 2016-2017 [Public Dataset]. https://support.google.com/analytics/answer/7586738?hl=en#access-the-dataset&zippy=%2Cin-this-article

Hosmer, David W., Lemeshow, Stanley, & Sturdivant, Rodney X. (2013). Applied logistic regression David W. Hosmer, Stanley Lemeshow, Rodney X. Sturdivant. Wiley series in probability and statistics. Hoboken, N.J.: Wiley. Oreilly https://learning-oreilly-com.proxy.lib.umich.edu/library/view/applied-logistic-regression/9781118548356/9781118548356c04.xhtml#c04_level1_3

Hughes, G., Kopetzky, J., & McRoberts, N. (2020). Mutual Information as a Performance Measure for Binary Predictors Characterized by Both ROC Curve and PROC Curve Analysis. Entropy (Basel, Switzerland), 22(9), 938. https://doi.org/10.3390/e22090938

Juez-Gil, M., Arnaiz-González, I., Rodríguez, J. J., López-Nozal, C., & García-Osorio, C. (2021, November 13). Approx-SMOTE: Fast SMOTE for Big Data on Apache Spark. *Neurocomputing*, *464*, pp. 432-437. doi:https://doi.org/10.1016/j.neucom.2021.08.086

Keany, 2021. The ultimate guide for clustering mixed data. Medium. doi: https://medium.com/analytics-vidhya/the-ultimate-guide-for-clustering-mixed-data-1eefa0b4743b

Liu, Y., Li, Z., Xiong, H., Gao, X., Wu, J., & Wu, S. (2013). Understanding and enhancement of internal clustering validation measures. *IEEE transactions on cybernetics*, *43*(3), 982–994. https://doi.org/10.1109/TSMCB.2012.2220543

Lutins, 2017. DBSCAN: What is it? When to Use it? How to Use it? Medium. doi: https://elutins.medium.com/dbscan-what-is-it-when-to-use-it-how-to-use-it-8bd506293818

Patel, S., Sihmar, S., & Jatain, A. (2015, March). A study of hierarchical clustering algorithms. In *2015 2nd international conference on computing for sustainable global development (INDIACom)* (pp. 537-541). IEEE. https://www.sciencedirect.com/science/article/pii/S0169023X02001386

Stoltzfus J. C. (2011). Logistic regression: a brief primer. Academic emergency medicine : official journal of the Society for Academic Emergency Medicine, 18(10), 1099–1104. https://doi.org/10.1111/j.1553-2712.2011.01185.x

APPENDIX A

Categorical encoding	Unique Count	NaN Count	Data_type	End format	Encoding	Rationale
channelGrouping	7	0	categorical	int64	one_hot	low cardinality
trafficSource.medium	6	0	categorical	int64	one_hot	low cardinality
device.deviceCategory	3	0	categorical	int64	one_hot	low cardinality
trafficSource.isTrueDirect	2	36309	categorical	int64	binary	binary
totals.newVisits	2	10351	categorical	int64	binary	relevant imputation
device.browser	30	0	categorical	float64	woe*	high cardinality
device.operatingSystem	14	0	categorical	float64	one_hot	medium cardinality
geoNetwork.country	179	0	categorical	float64	woe	high cardinality
trafficSource.source	96	0	categorical	float64	woe	high cardinality
hits.hour	24	0	categorical	int64	ordinal	3 peak periods

* weight of evidence

Additions	Unique Count	NaN Count	Data_type	End format	Encoding	Rationale
Monetary	na	49521	numeric	float64		RFM analysis
frequency	14	0	numeric	int64		RFM analysis
Recency	184	0	numeric	int64		RFM analysis
buyers	2	0	boolean	int64		Classification task
repurchasers	2	0	boolean	int32		Classification task

Dropped columns	Rationale
totals.screenviews	All NaNs
totals.uniqueScreenviews	All NaNs
totals.timeOnScreen	All NaNs
hits.transaction.transactionRevenue	Duplicate with transaction revenue
hits.item.productName	All NaNs
hits.item.productCategory	All NaNs
hits.item.itemRevenue	All NaNs
hits.transaction	Duplicate with transaction revenue
hits.item	Duplicate with transaction revenue
hits.dataSource	Duplicate with transaction revenue
hits.eCommerceAction	Duplicate with transaction revenue
hits.appInfo.screenDepth	It just contains 0s only, we can use totals.pageviews
geoNetwork	Successfully retrieved unnested columns
totals.visits	Duplicate with page views
trafficSource	Successfully retrieved unnested columns
totals	Successfully retrieved unnested columns
socialEngagementType	Only one value
hits	Successfully retrieved unnested columns
device	Successfully retrieved unnested columns
hits.minutes	captured by bins in hits hours

APPENDIX A. Categorical feature encoding, drops, and additions

APPENDIX B

Pandas Profiling and Datasets

	socialEngagementType	channelGrouping	date	fullVisitorId	totals.hits	totals.pageviews	totals.timeOnSite	totals.transactions	totals.ne
0	Not Socially Engaged	Organic Search	20170101	8160804435292640144	1	1.0	NaN	NaN	1.0
1	Not Socially Engaged	Organic Search	20170101	2767232911658101391	1	1.0	NaN	NaN	1.0
2	Not Socially Engaged	Organic Search	20170101	5462105261493871939	1	1.0	NaN	NaN	1.0
3	Not Socially Engaged	Organic Search	20170101	95191702071842980	1	1.0	NaN	NaN	1.0
4	Not Socially Engaged	Organic Search	20170101	6692505544910101563	1	1.0	NaN	NaN	1.0
5	Not Socially Engaged	Organic Search	20170101	2907946815222203879	1	1.0	NaN	NaN	1.0
6	Not Socially Engaged	Direct	20170101	5652181055723415564	1	1.0	NaN	NaN	1.0
7	Not Socially Engaged	Direct	20170101	7817519219547424500	1	1.0	NaN	NaN	1.0
8	Not Socially Engaged	Organic Search	20170101	8362452234388944224	1	1.0	NaN	NaN	1.0
9	Not Socially Engaged	Paid Search	20170101	1309141223204608712	1	1.0	NaN	NaN	1.0

Table B1. The first 10 rows and 9 expanded columns of Google Analytics 360 dataset. A 100 row sample with all the columns available included in our analysis can be found in the accompanying repository: https://drive.google.com/file/d/1zqdKCXxCik3PM1CM-fnY1P-d5UOxU2Is/view?usp=sharing. The last 10 rows and other exploratory information can can be found in the dashboard files folder of our Google Drive file repository: https://drive.google.com/drive/folders/1TZEaAD8GEJQbSMb8U6xkpykpYjWgRbVi?usp=sharing. To open a dashboard, select the file, right click to download it, and open the file to a local browser to view the html. At this time, Google does not have an external product for viewing html files natively in Google Drive. The pipeline that orchastrates the entire pipeline can be found here: https://drive.google.com/drive/folders/1kvTrg9TXHbGCfq4WWakTYpJuA1dV6SBc?usp=sharing. The requirement.txt file contains the packages needed to files locally. The Github, located at https://github.com/tiaaburton/Milestone-II-Machine-Learning-Project, provides more insight into how to run the Makefile in the README file.

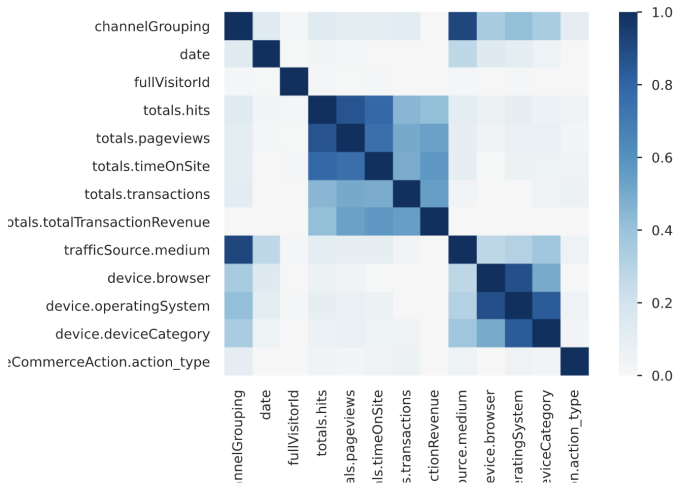


Figure B2. Phik correlation coefficient chart provided by Pandas Profiling.

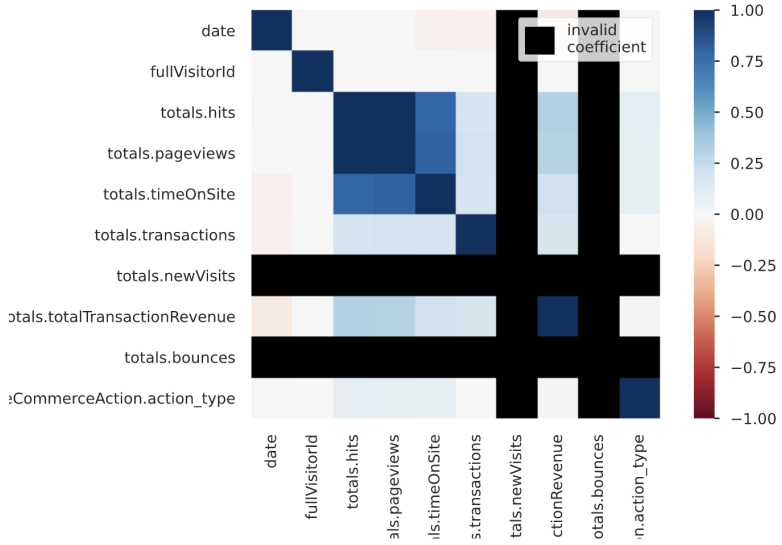


Figure B3. Spearman's rank correlation coefficient chart provided by Pandas Profiling.

APPENDIX C

index	mutual_info_score	column_name
0	0.2224047251232546	totals_pageviews
1	0.21369987102682897	totals_hits
2	0.18434487479249806	totals_timeOnSite
3	0.07947251327279314	geoNetwork_country_woe
4	0.0717881721368232	totals_bounces
5	0.055626753396393	trafficSource_source_woe
6	0.04281617307389696	totals_newVisits
7	0.04080335781813771	trafficSource_medium_referral
8	0.040670776313571855	channelGrouping_Referral
9	0.03833979352667072	channelGrouping_Social
10	0.02259664649428461	trafficSource_isTrueDirect_code
11	0.018251408534723623	device_browser_woe
12	0.01416522914692453	hits_hour_ordinal
13	0.00953319647449491	device_operatingSystem_Macintosh
14	0.008478900187753657	device_operatingSystem_Windows
15	0.005698507284693344	hits_eCommerceAction_action_type
16	0.004892354807879418	device_operatingSystem_Chrome_OS
17	0.004578502132303042	device_deviceCategory_mobile
18	0.004027430832742818	device_operatingSystem_Android
19	0.0036514675365098004	device_operatingSystem_iOS
20	0.0024039320087969607	device_operatingSystem_BlackBerry
21	0.0019666024645521585	socialEngagementType

22	0.0017352577508524814	channelGrouping_Direct
23	0.001603384770593852	device_operatingSystem_Nintendo_Wii
24	0.0013614014616167847	device_deviceCategory_tablet
25	0.0009435229102632281	trafficSource_medium_cpm
26	0.0005940806674398225	device_operatingSystem_Firefox_OS
27	0.000419475746574971	trafficSource_medium_affiliate
28	0.00018128487567548035	trafficSource_medium_cpc
29	0.0	channelGrouping_Display
30	0.0	channelGrouping_Organic Search
31	0.0	channelGrouping_Paid Search
32	0.0	trafficSource_medium_organic
33	0.0	device_operatingSystem_Linux
34	0.0	device_operatingSystem_Samsung
35	0.0	device_operatingSystem_Windows_Phone
36	0.0	device_operatingSystem_Xbox

APPENDIX C. The A1 models in the classification task for converters used columns with the highest mutual information scores used as decided by a varying threshold between 0 and 1. For all datasets, columns with 0.0 mutual information were removed from the dataset.

APPENDIX D - A1 Graphs

Bar Chart Visualizations of the Likelihood to Convert Logistic Regression Model

Likelihood to convert results can be found in the Share Google drive in Google Drive: https://drive.google.com/file/d/1ZkKGeevZLEL4pfliNusVYAUc-WRVDKRS/view?usp=sharing. Means were calculated by filtering the dataset to the correct model and, if needed, column size.

Figure D1. F1-Scores from the Logistic Regression Models trained on balanced and unbalanced datasets. This chart should demonstrate the F1-Score on the training dataset versus the test dataset. Each two bars should represent the training vs test. For example, bar 0 is “training vs training” and bar 1 is “training vs test”, and so on and so forth.

Figure D2. Precision scores from the Logistic Regression Models trained on balanced and unbalanced datasets. This chart should demonstrate the Precision on the training dataset versus the test dataset. Each two bars should represent the training vs test. For example, bar 0 is “training vs training” and bar 1 is “training vs test”, and so on and so forth.

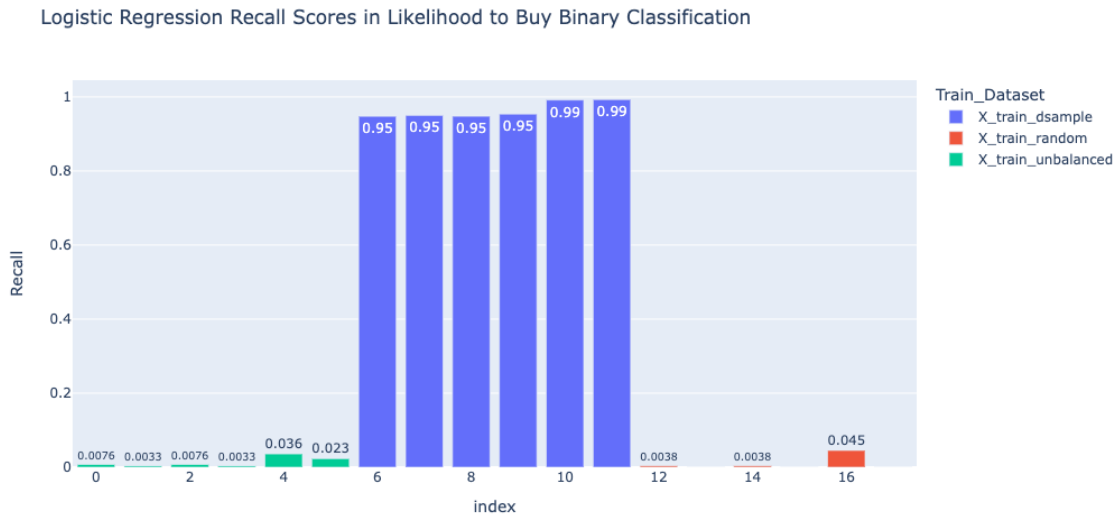


Figure D3. Recall scores from the Logistic Regression Models trained on balanced and unbalanced datasets. This chart should demonstrate the Recall on the training dataset versus the test dataset. Each two bars should represent the training vs test. For example, bar 0 is “training vs training” and bar 1 is “training vs test”, and so on and so forth.

APPENDIX E

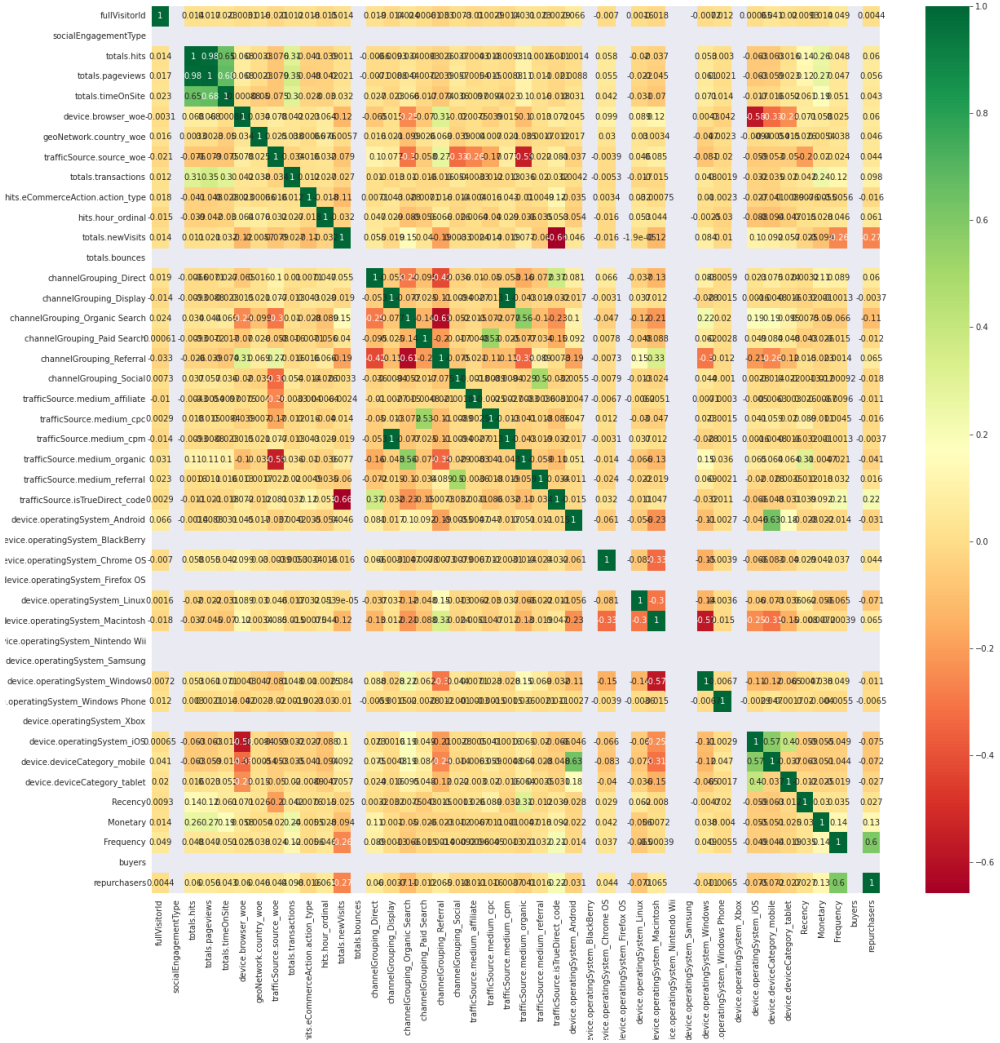


Figure 1. A2 Feature Correlation.

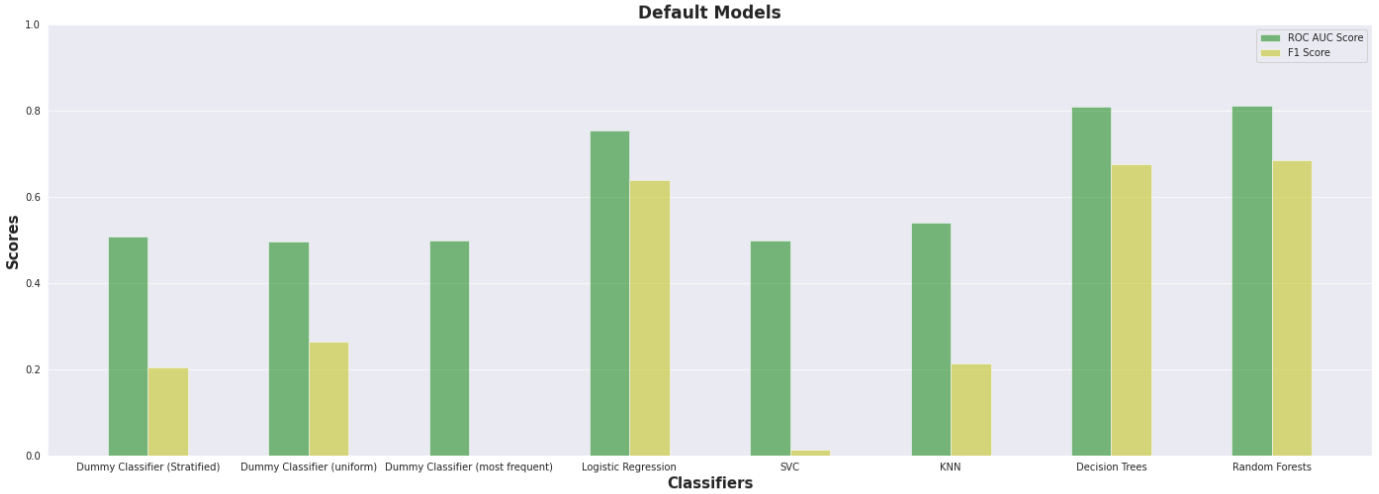


Figure 2. Default Model trained on original dataset with no param tuning.

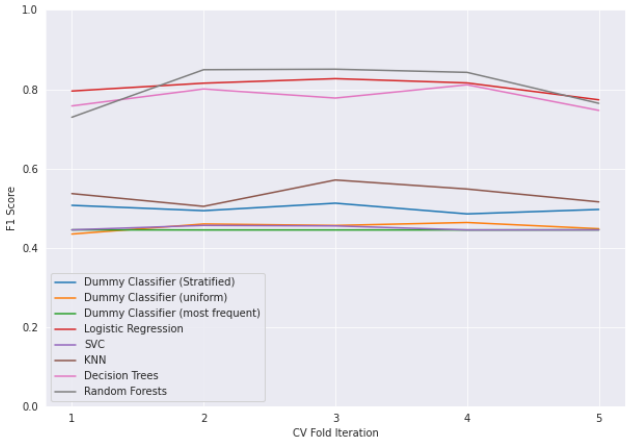


Figure 3. 5 Fold Cross Validation Result to verify whether our dataset is biased.

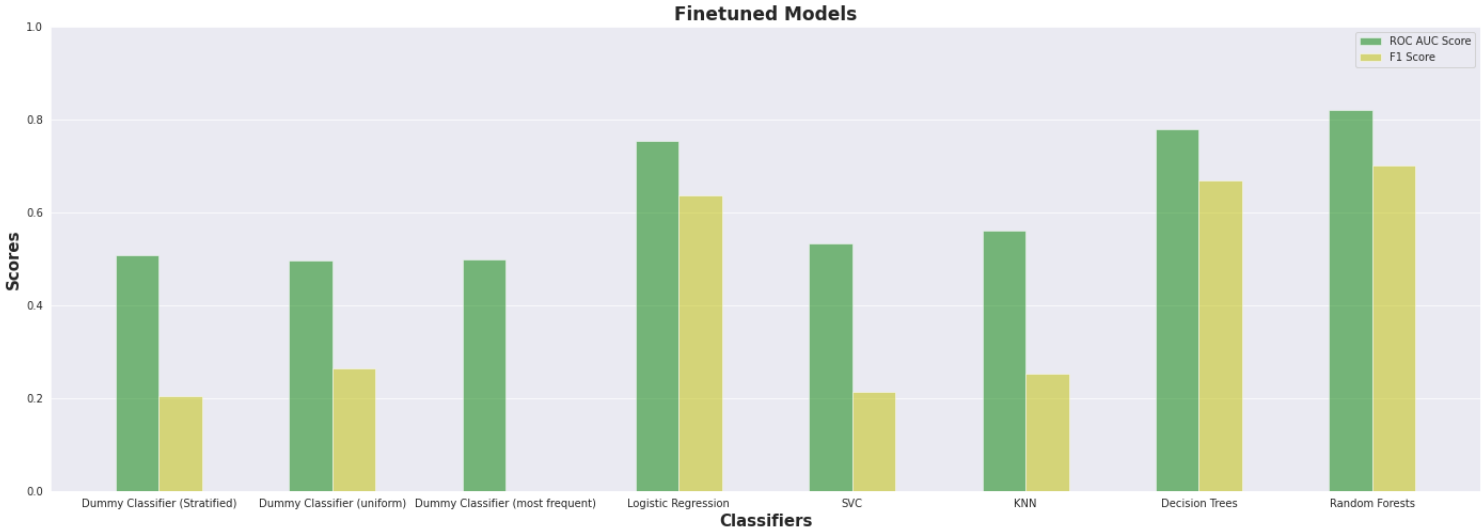


Figure 4. A2 Classifiers trained using optimal parameters via Grid Search.

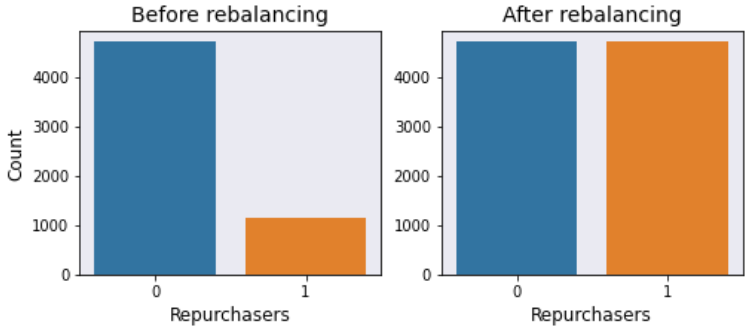


Figure 5. Before and After rebalancing (via SMOTE) the Repurchasers Dataset.

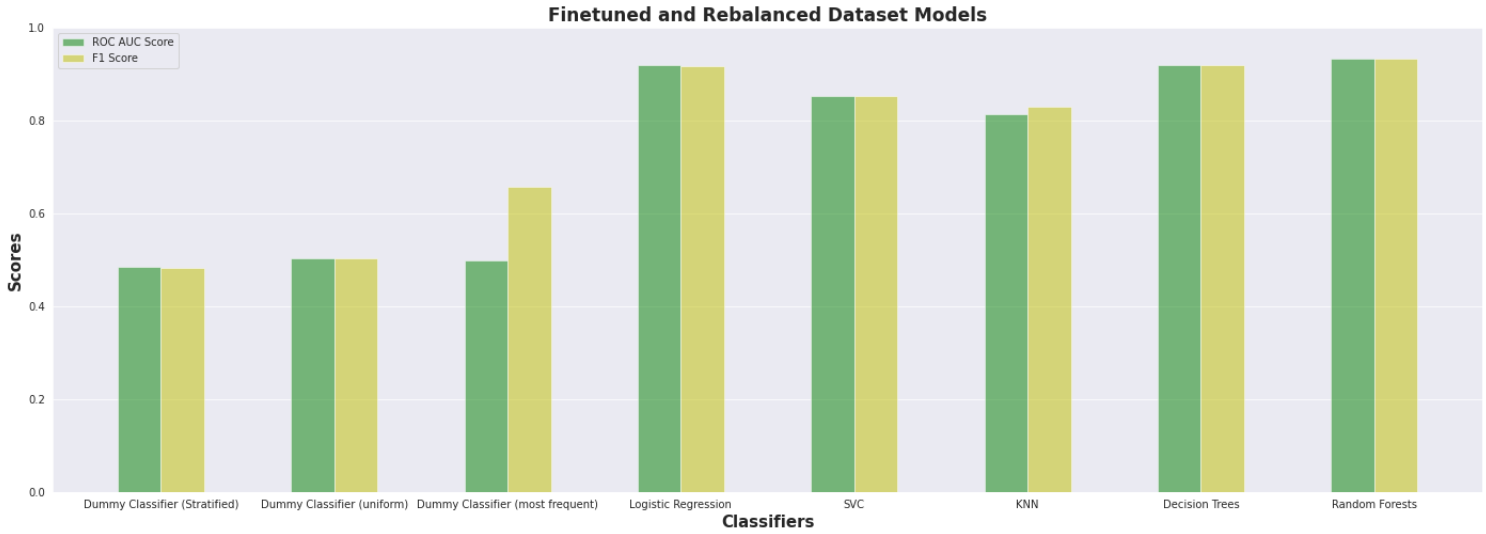


Figure 6. A2 Model Accuracies after training the model with optimal parameters found with Grid Search and training from the rebalanced dataset.

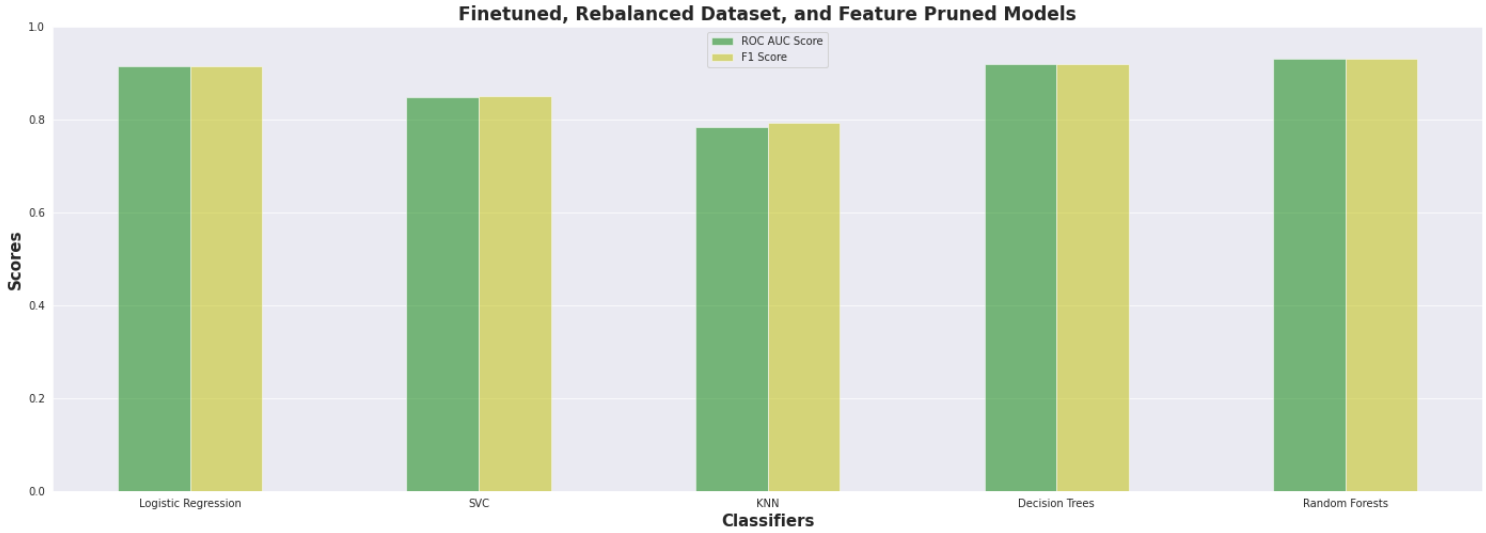


Figure 7. Finetuned Model trained on the rebalanced dataset with pruned features (down from 42 to 6 features–with Gini Scores above 0.01) accuracies.

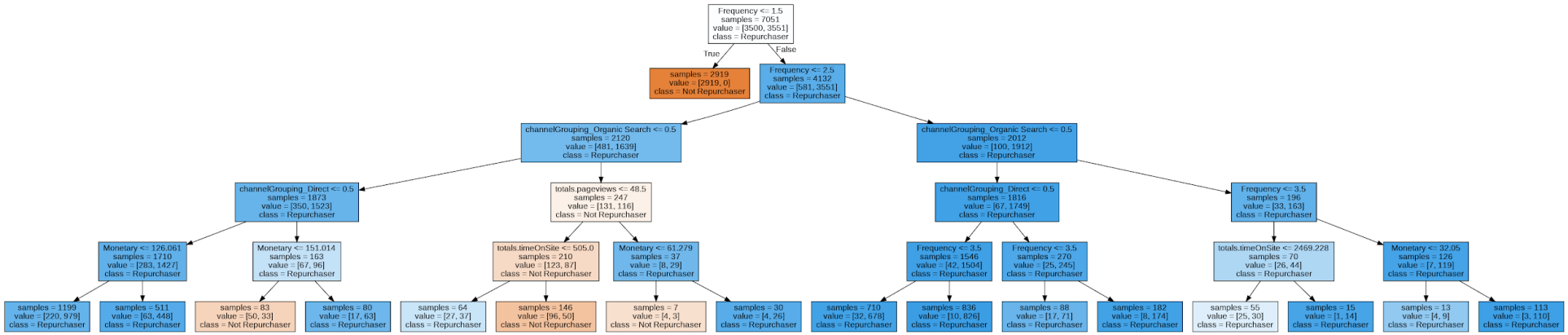


Figure 8. Graph of Decision Tree with Max Depth of 5.

feature	importance
Frequency	0.8626647389
channelGrouping_Organic Search	0.01817720719
Monetary	0.01670718011
channelGrouping_Direct	0.01229520975
totals.timeOnSite	0.01111920728
totals.pageviews	0.01001248176

Chart 1. Top 6 Feature Importance as determined by the Decision Trees, “Gini Importance”. Link for full list of feature importance values: https://drive.google.com/file/d/1-VZm-UFbYAdfcM1zZG48pVkGSSOjkqee/view?usp=sharing

APPENDIX F

$$\left(\sum_{i=1}^{n} |X_i - Y_i|^p\right)^{1/p}$$

$$\left(\sum_{i=1}^{n} |X_i - Y_i|^2\right)^{1/2}$$

Formula F1. Minkowski Distance to Euclidean Distance Proof

$$F_\beta = (1 + \beta^2) \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{(\beta^2 \cdot \mathrm{precision}) + \mathrm{recall}}$$

Formula H2. F1-Score

$$\mathrm{Precision} = \frac{tp}{tp + fp}$$

Formula F3. Recall

$$\text{Recall} = \frac{tp}{tp + fn}$$

Formula F4. Precision

$$\phi(\boldsymbol{p}) = \sum_j p_j(1 - p_j)$$

Formula F5. Gini

$$\phi(\boldsymbol{p}) = -\sum_j p_j \log p_j$$

Formula F6. Entropy