03

Ecommerce Purchase Prediction Model

By Dax, Liam, Don, and Hunter

Executive Summary

Using an ecommerce data set we trained a classifier model to detect if a user would make a purchase.

We analyzed their performances and tried different preprocessing steps to see how it affected each model to gain an understanding of how to best tune a good model.

Exploratory Data Analysis

- The data consists of information about what types of pages the users visited on the site, how long, and whether they left the site or remained. It also contained info the users operating system and browser, customer type and if purchased were made leading up to a “special day” (holidays, sale days, etc.)

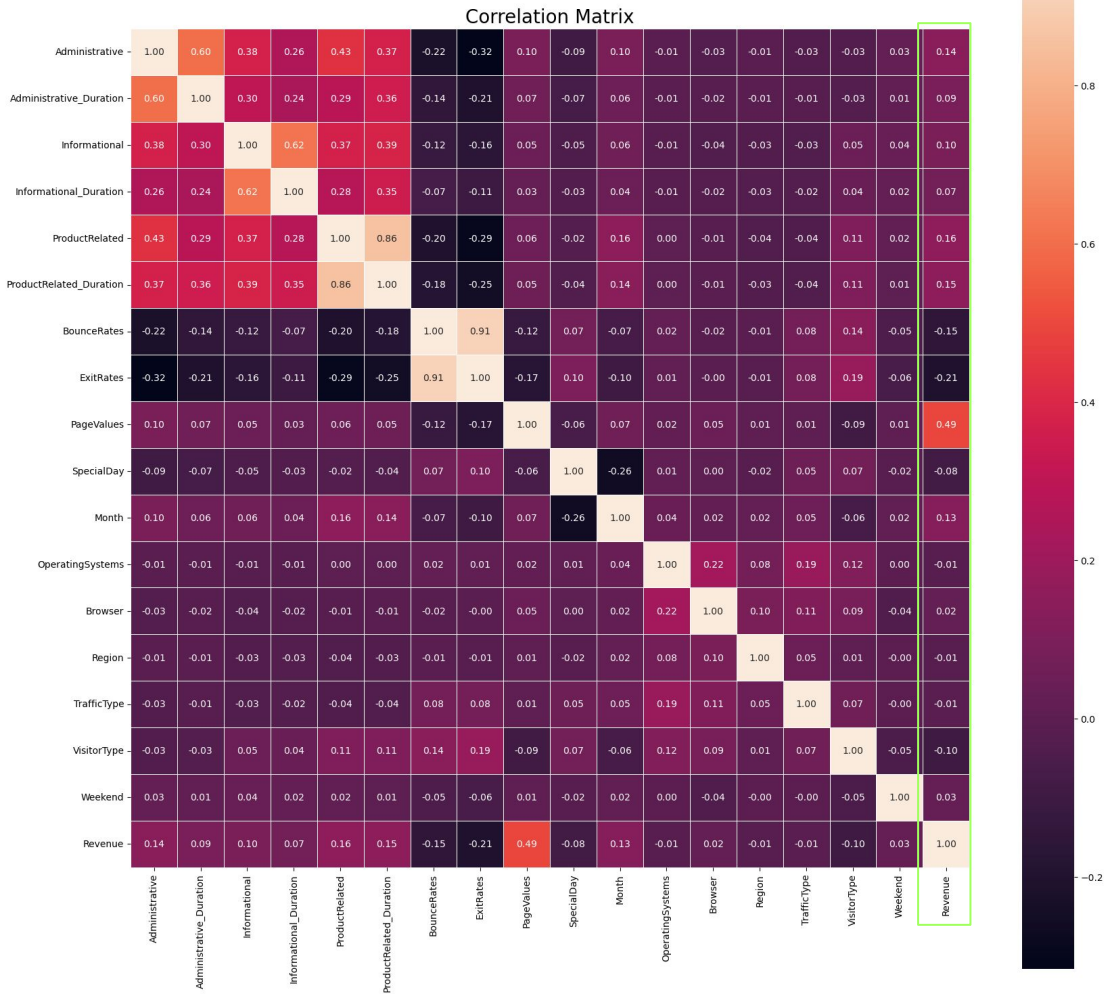
- We identified our Label as a binary indicating whether a purchase was made or not
  - The data was unbalanced
    - Only 15.5% of online shoppers in the dataset made a purchase

Correlations

Highest correlation to Revenue:

- PageValues = Total Revenue / unique page views
  - Google Analytics metric
  - Tells you which pages drive more sales
- Page types & duration on those pages, but not particularly useful
- Exit Rates & Bounce Rates, only indicates most exit rates were bounces

https://infotrust.com/articles/a-guide-to-page-value/



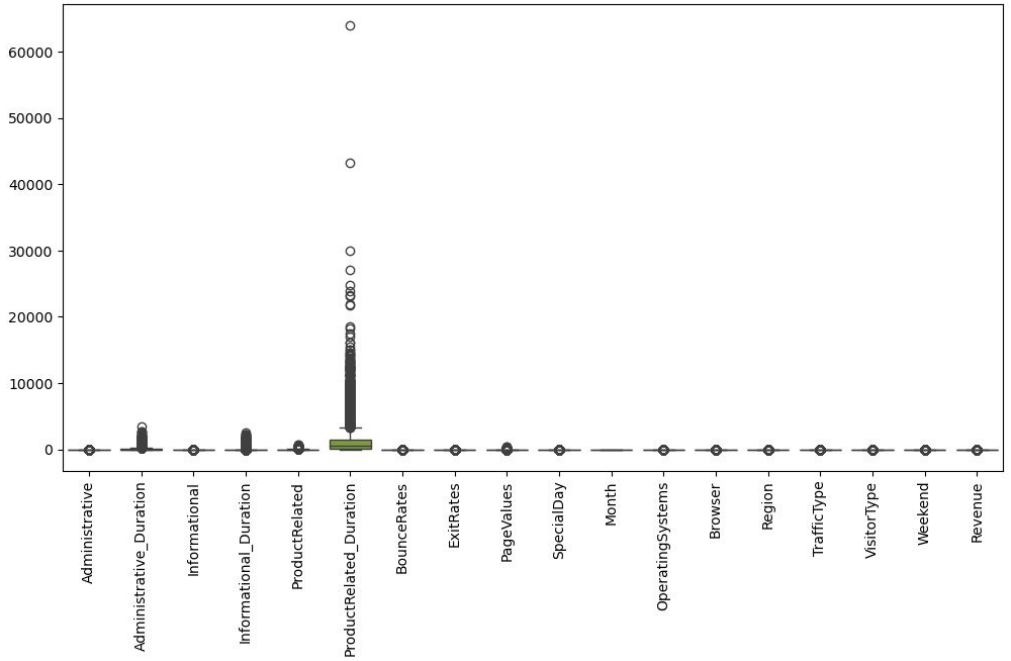
Cleanup and Preprocessing

Cleaning

- Dataset was already clean (no nulls, no redundant columns, etc.)

Preprocessing

- Applied SMOTEENN to handle imbalanced data
- Performed encoding for categorical data
  - **Month** and **Visitor Type**
- Used MinMaxScaler and StandardScaler for **numerical data.**



Approach & Methodology

We tested multiple models:

- Logistic Regression
- Random Forest
- XGBoost
- AdaBoost

Handling Imbalances:

- Used **SMOTEENN** to oversample minority class and undersample majority class.
- Compared model performance **with and without SMOTEENN.**

Approach & Methodology (II)

Evaluated Performance Metrics:

- **Accuracy & Balanced Accuracy** (for imbalanced data).
- **ROC-AUC Scores** to measure classification effectiveness.
- **Confusion Matrices** to analyze misclassifications.

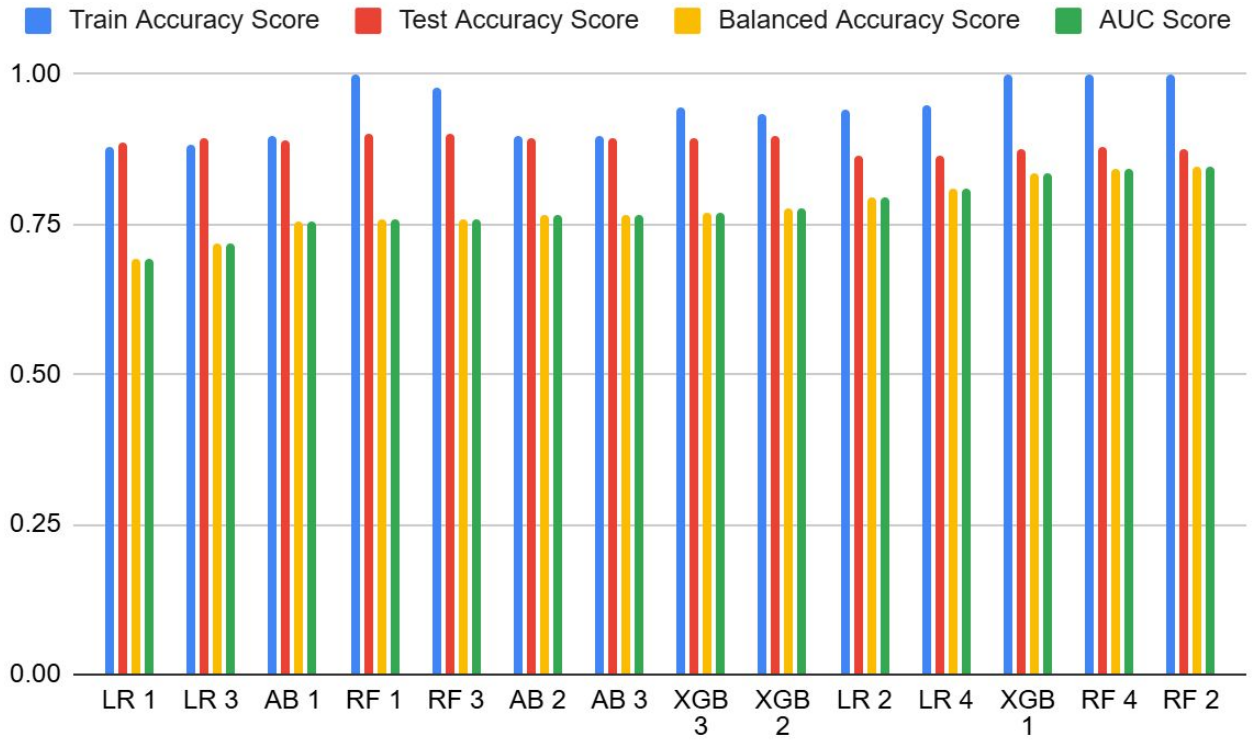
Optimizations:

- Hyperparameter tuning using **GridSearchCV** and **RandomizedSearchCV**.
- Identified overfitting in some models

Why test different models?

- Logistic Regression
  a. Was supposed to be less prone to overfitting
  b. Works well with small data sets, 10–50 features, between 100s to a few thousand rows.
  c. Assumes a linear relationships between features and the log–odds
- Random Forest
  a. Reduces Overfitting
  b. Handles noise and imbalanced data well
  c. Oversampling can make it less performant
- XGBoost
  a. One of the best models for structured data
  b. Effective for imbalanced data
  c. Can overfit easily
- AdaBoost
  a. Good for imbalanced data
  b. Sensitive to noise

Model Performance Results



Results & Conclusions

- Best Performing Model: XGBoost with Hyperparameter Tuning (Attempt #2)
  - Test Accuracy: 89.9%
  - Balanced Accuracy: 77.6%
  - AUC Score: 0.7759
  - Improved Recall for Minority Class (Purchasers).

- Effect of SMOTEENN:
  - Increased recall for the minority class but **slightly reduced overall accuracy.**
  - Balanced accuracy improved, showing better handling of class imbalance.

Results & Conclusions (II)

Overfitting Detection:

- Some models (e.g., XGBoost Attempt #1) had overfitting issues (train−test accuracy gap > 8%).
- Hyperparameter tuning reduced overfitting while maintaining performance.

Conclusions & Next Steps:

- XGBoost with Hyperparameter Tuning is the best tradeoff between accuracy and balance.
- Further tuning of class balancing techniques (e.g., adjusting SMOTE parameters) may improve results.
- Potential future work:
  - Try ensemble learning (e.g., stacking models).
  - Explore feature engineering to improve prediction robustness.

Future Research & Questions

Are there alternative feature engineering techniques that might enhance predictive accuracy?

Can we combine classifiers or make a voting system that combines different classifiers for better performance?

Can we investigate nonlinear relationships between columns?