Google Store

Automated Revenue Prediction

University of Minnesota
Driven to Discover℠

Danny Moncada
Eduardo Chavez

MSBA 6420
Professor Yichen Song

Business Case

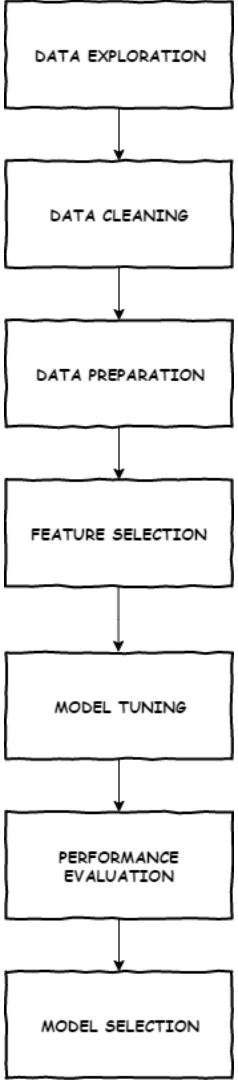
THE **80/20 rule** OF MOST BUSINESSES

APPROPRIATE MARKETING INVESTMENT

PROMOTIONAL STRATEGIES

PREDICTING REVENUE

Solution Overview



Data Exploration

1.7M observations.

JSON attributes

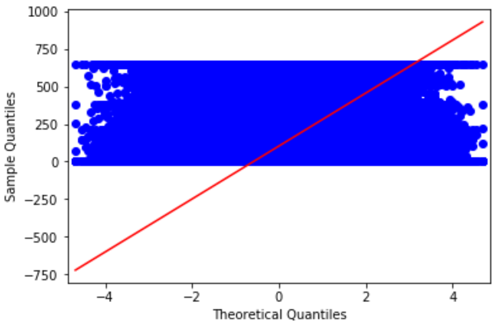
Totals attribute

Flattening data: 903653 observations

1.27% (11515 rows) revenue data

null values and several categorical columns

Testing Normality



Shapiro Test

Statistics=0.642, p=0.000

Sample does not look Gaussian (reject H0)

Data Cleaning

1. Scaling down revenue
2. Missing Revenue
3. Purchase column
4. Missing categorical values

JSON Processing Script

{

"visits":"1",

"hits":"17",

"pageviews":"13",

"timeOnSite":"611",

"transactions":"1",

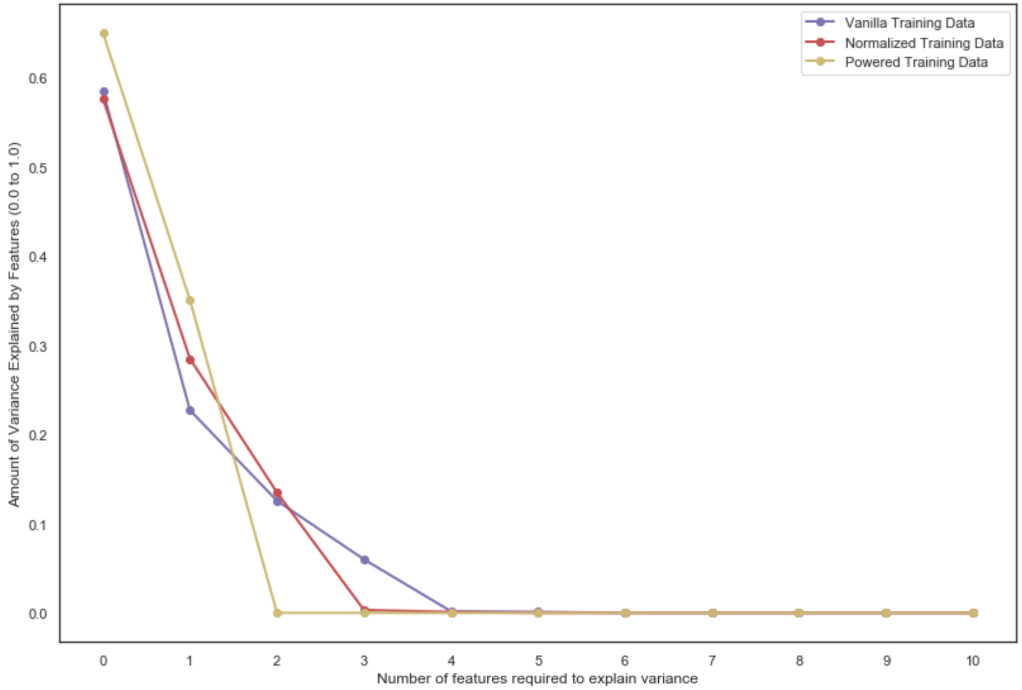
"transactionRevenue":"24980000",

"totalTransactionRevenue":"26980000",

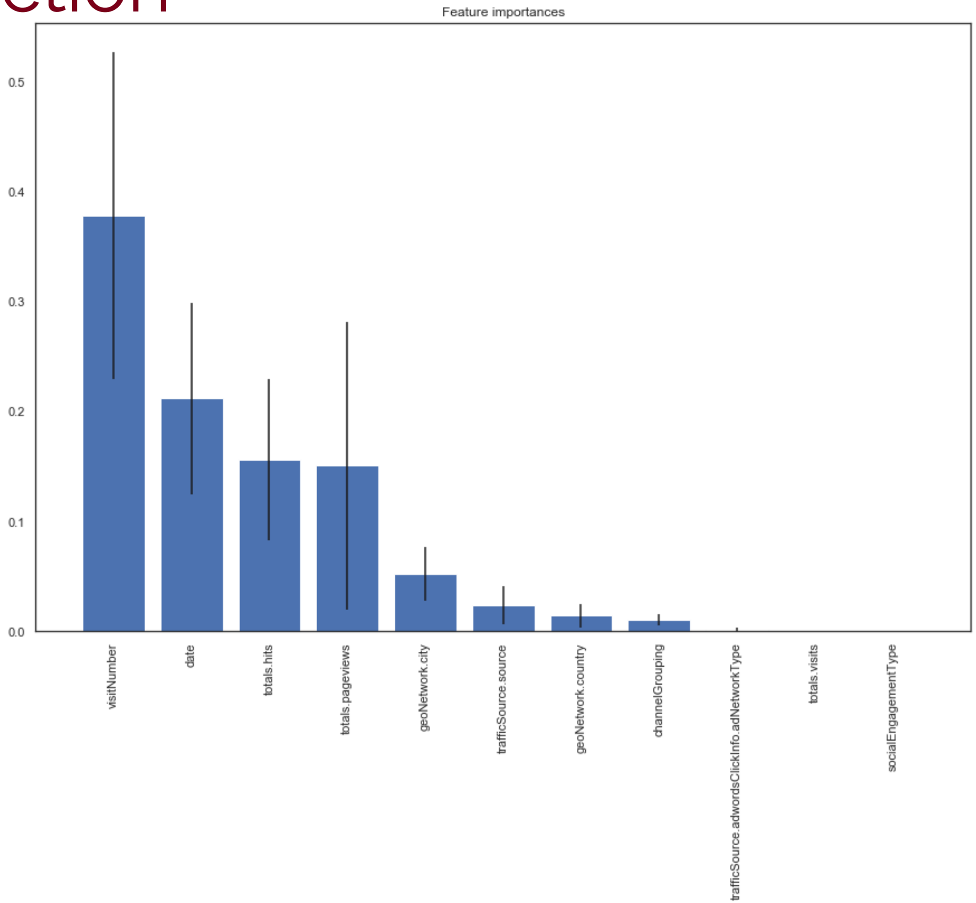
"sessionQualityDim":"20"

}

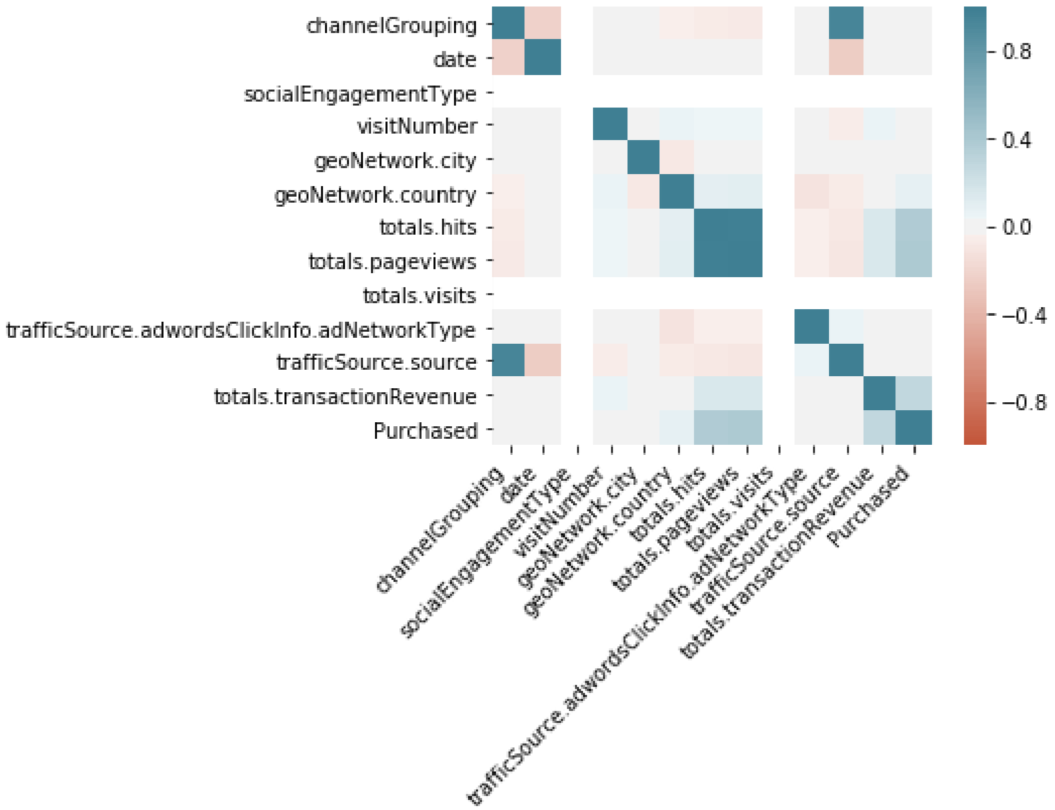
Feature Selection



Feature Selection



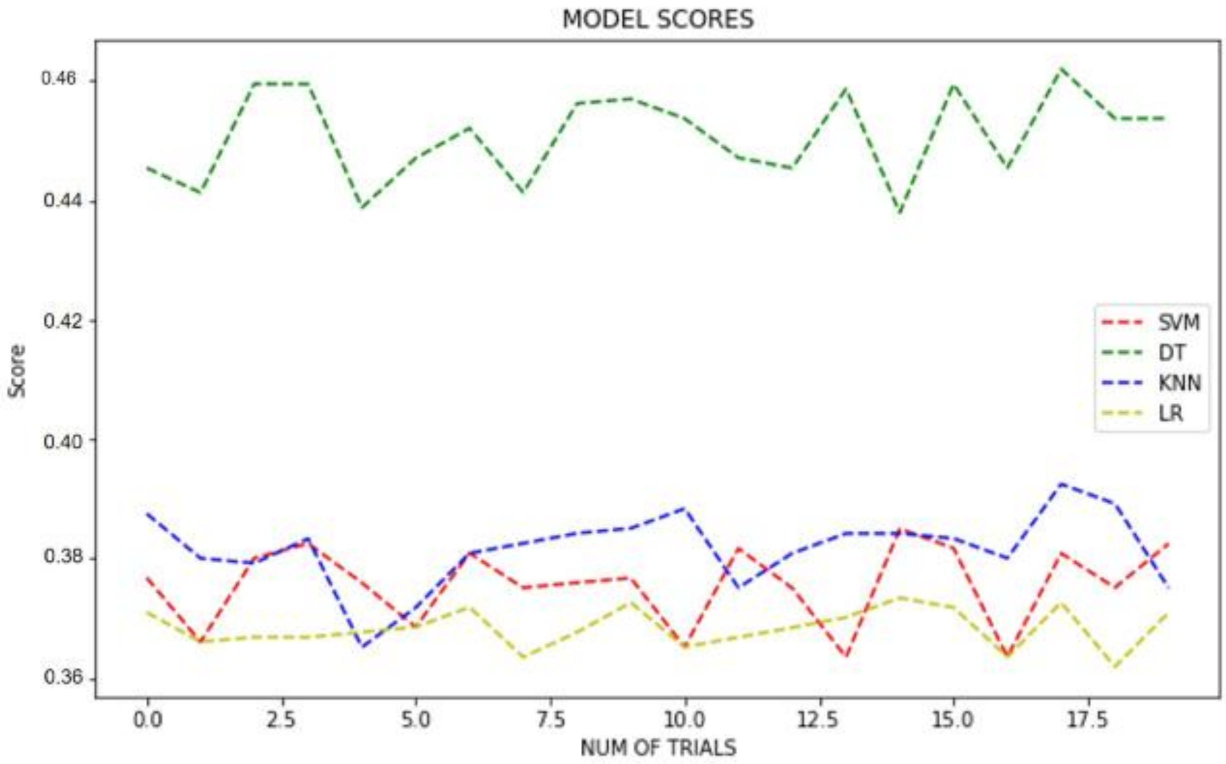
Feature Selection



Modeling Prep

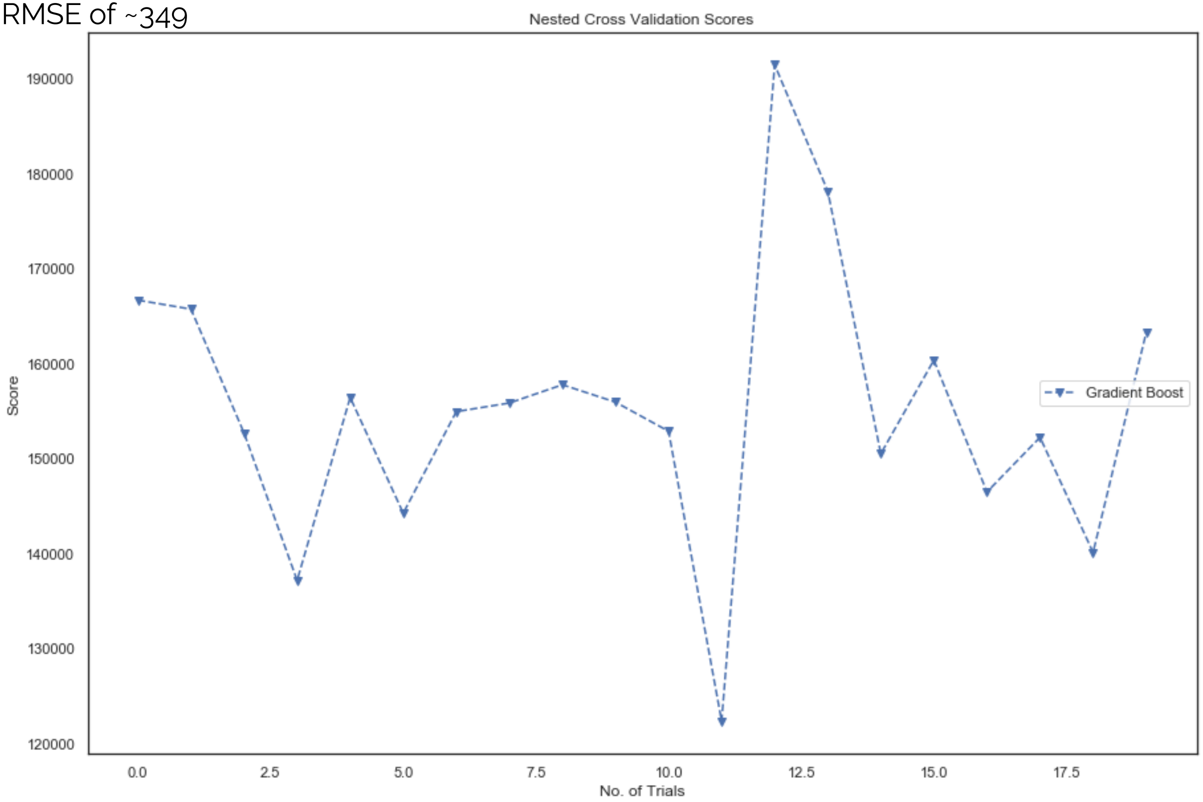
1. Normalization: RobustScaler
2. Stratified K-Fold
3. Target Log()

Modeling Classification



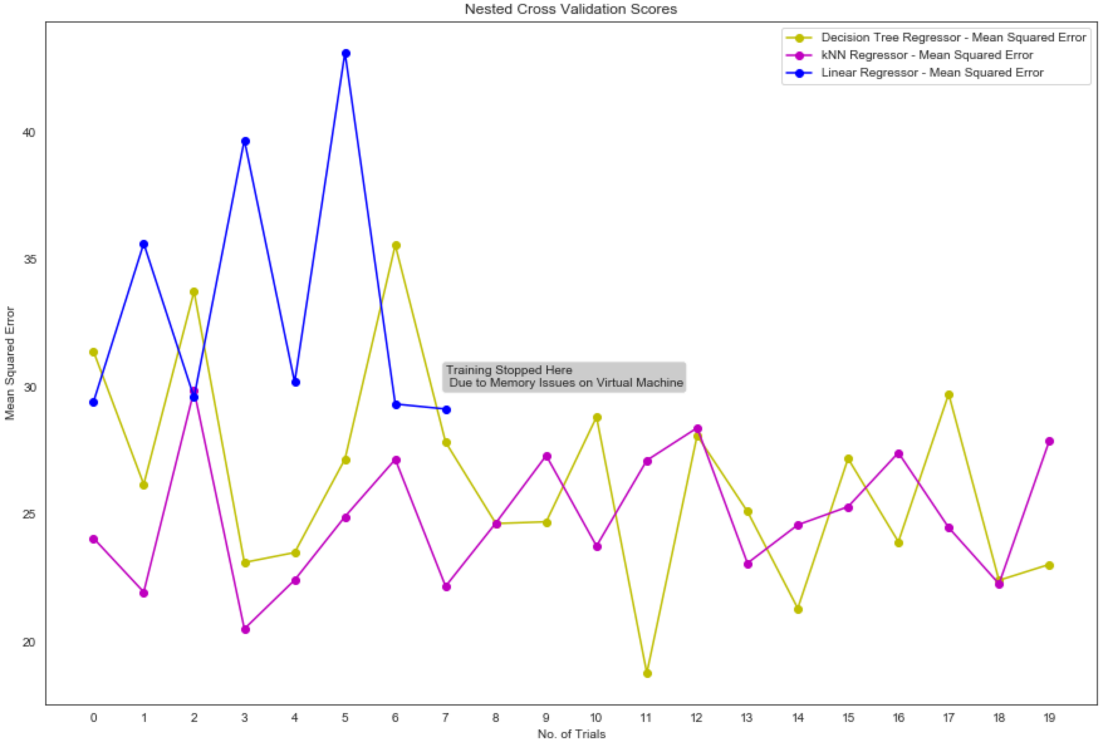
Regression: Gradient Boost

MSE of 122344, with an RMSE of ~349



Regression: Decision Tree, KNN, Linear

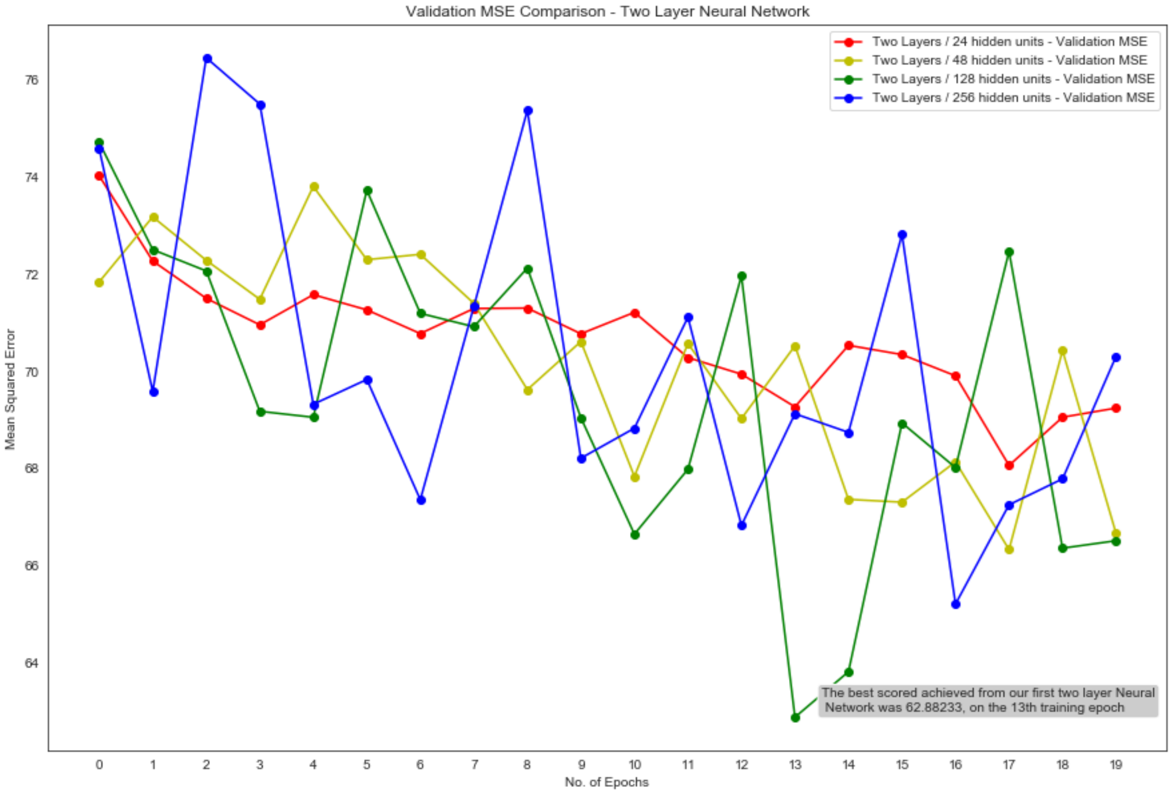
MSE score was ~18 (which is an RMSE of 4.24) by our DecisionTreeRegressor



Regression: Deep Neural Networks

1. Build a two-layer neural network (using a function).
2. Initialize a list of hidden units to test with the two layer model.
3. Run some training epochs to see if the neural network could perform better than our earlier attempts.

Regression: Deep Neural Networks



Regression: LIGHT GBM!

1. Faster training speed and higher efficiency
2. Lower memory usage
3. Better accuracy
4. Parallel and GPU learning
5. Handling large-scale data

Regression: LIGHT GBM!

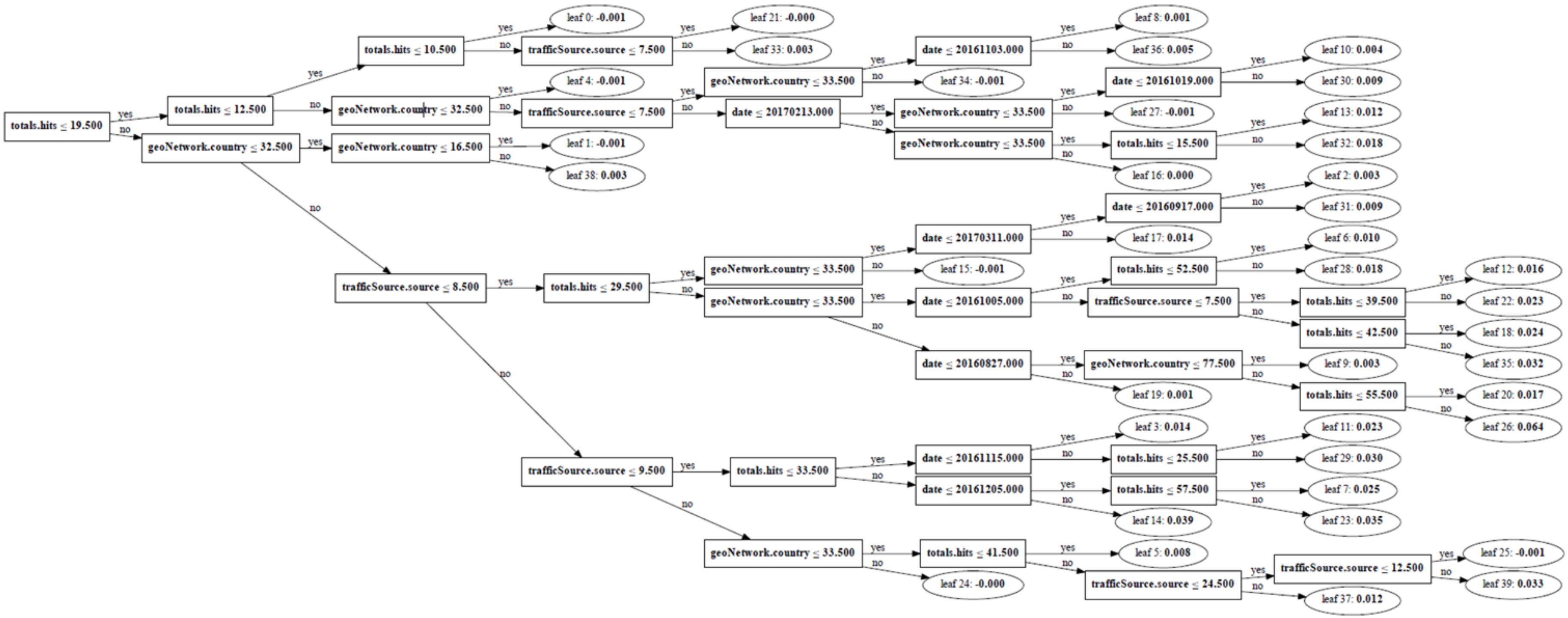
1. Build an LGB Model
2. Revisit our training data and create a new instance of training data
3. Create a new target variable using the appropriate function - using a np.log1p function
4. Use a pandas function factorize to transform our categorical features into integers
5. Convert all numeric fields to integer
6. Run the model

Regression: LIGHT GBM!

Results:

Training until validation scores don't improve for 200 rounds
[500] training's rmse: 1.65597 valid_1's rmse: 1.66146
[1000] training's rmse: 1.61644 valid_1's rmse: 1.63829
[1500] training's rmse: 1.59667 valid_1's rmse: 1.63148
[2000] training's rmse: 1.58229 valid_1's rmse: 1.62939
[2500] training's rmse: 1.57079 valid_1's rmse: 1.62905
[3000] training's rmse: 1.56038 valid_1's rmse: 1.62844
Early stopping, best iteration is:
[3276] training's rmse: 1.5551 valid_1's rmse: 1.62817
LGBM: RMSE val: 1.62817 - RMSE train: 1.5551
Wall time: 1min 21s

Regression: LIGHT GBM!



Conclusions

- Technical limitations: Sampling and Partitioning
- Aggregating to the right level
- Separate pipelines
- LightGBM and XGBoost
  - histogram based split vs sklearn GBM
  - Reduced cost of calculating the gain for each split
  - Pre-sort-based algorithms have time complexity
- CatBoost
- Preparing Marketing teams to predict revenue and find user behavior that leads to the final sale.