Grupo Bimbo Inventory Demand Challenge

Using NLP and Logistic Regression Modeling to Determine Which Regions, Stores, and Products will lead to Higher Returned Inventory.

Manav Dalmiya
NYC General Assembly Data Science Course
September 2016

kaggle

GRUPO BIMBO®

Project Background

- Dataset: Grupo Bimbo Inventory Demand Challenge posted on Kaggle.com.
- Objective: For participants to accurately predict returned inventory using historic operations and sales data using time-series data.
- Grupo Bimbo is a Mexican multinational bakery product manufacturing company.
- Mexico's 9th largest company with reported revenues of US$14.1 Billion (2014)
- Operations background

Returned Inventory and Dataset Predictors

Operational	Regional	Clients and Products	Sales Figures
• Week Number • Sales Depot ID • Sales Channel ID • Sales Route ID	• Town • State	• Client Name • Product Name	• Weekly Sales Units (Int) • Weekly Sales (Mex$) • Previous Week Returned Units (Int) • Previous Week Return Sales (Mex$) • Adjusted Demand [Predicted Variable] (Int)

- Adjusted Demand = Weekly Sales Units - Previous Week Returned Units

- Goal of the project is to determine which of these variables is most predictive of Adjusted Demand. A successful model will uncover insights that will allow to produce and distribute the market clearing amount of inventory more consistently.

Data Cleaning and Early Challenges

- Data Constraints:
  - Cut Train data set from 7.4 million to 250,000 because of iPython's data constraints.
  - Ideally, I would've taken a random sample from each of the nine weeks.
  - Transformed data set from time-series data to cross-sectional data only looking at Week 3.
- Translation
  - Data set's columns names, client names, and product names were in Spanish.
  - Working with the data set would require translating the string values of client name and product name in order to meaningfully predict which stores and products would have the highest returns.
  - Fortunately, I found a Kaggle kernel that used TF-IDF identify the most frequent words and client names to filter the terms into client types.

TF-IDF on Client Names

Step 1: First 5 Rows of Client Name Column

	Cliente_ID	NombreCliente
0	0	SIN NOMBRE
1	1	OXXO XINANTECATL
2	2	SIN NOMBRE
3	3	EL MORENO
4	4	SDN SER DE ALIM CUERPO SA CIA DE INT

Step 2: Examine Names with Most Occurrences

```
NO IDENTIFICADO    281670
LUPITA               4863
MARY                 3016
LA PASADITA          2426
LA VENTANITA         2267
LA GUADALUPANA       1299
```

Step 3: Perform TF-IDF Function on Client Name Column

```
(b'no', 0.6888126004393861)
(b'identificado', 0.6849292193081505)
(b'la', 0.14990532034895288)
(b'el', 0.08328478631485127)
(b'abarrotes', 0.0800093233318993)
(b'de', 0.060769929775020375)
(b'maria', 0.046819424108208656)
```

Step 4: Identify Potential Client Groupings

	Cliente_ID	NombreCliente
78	1438	CAFETRIA PREPARATORIA
1095	5045	CAFETERIA DE LA SECUNDARIA 13
1098	5048	CAFETERIA PREPA 2
1233	5416	CAFETERIA
1318	5612	CAFETERIA NORMAL DE PROFESORES

TF-IDF on Client Names

Step 5: Filter existing client names as client types.

```
def filter_specific(vf2):

    # Known Large Company / Special Group Types
    vf2['NombreCliente'] = vf2['NombreCliente'].str.replace('.*REMISION.*','Consignment')
    vf2['NombreCliente'] = vf2['NombreCliente'].replace(['.*WAL MART.*','.*SAMS CLUB.*'],'W
almart', regex=True)
    vf2['NombreCliente'] = vf2['NombreCliente'].str.replace('.*OXXO.*','Oxxo Store')
    vf2['NombreCliente'] = vf2['NombreCliente'].str.replace('.*CONASUPO.*','Govt Store')
    vf2['NombreCliente'] = vf2['NombreCliente'].str.replace('.*BIMBO.*','Bimbo Store')
```

Step 6. Value Count on New Client Names

Individual	100005
Small Franchise	77937
NO IDENTIFICADO	30873
General Market/Mart	18956
Eatery	7447
Supermarket	7351
Post	3699
Hospital/Pharmacy	2471
Walmart	990
School	497
Fresh Market	371
Govt Store	219
Hotel	185
Bimbo Store	107
Oxxo Store	84

Further Data Preparation

- Binarizing Variables

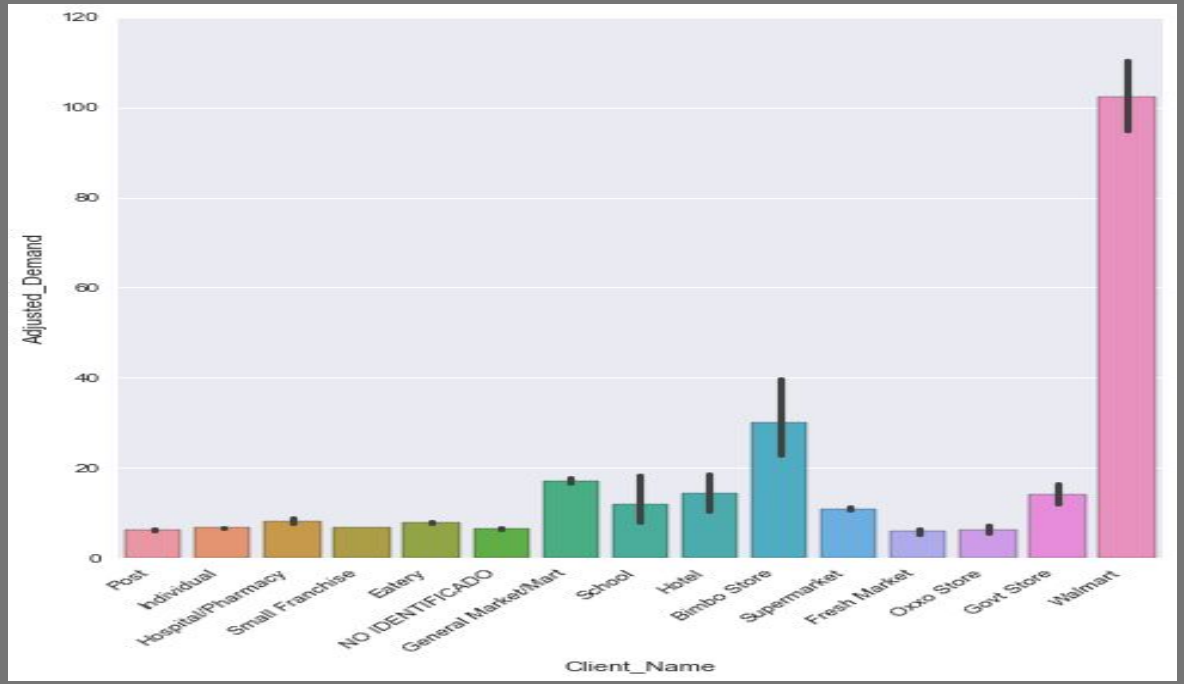
Categorical Variable	Binarized Variables
Sales Depot ID	• Depot_1110, … , Depot_1116
State	• State_Mexico, State_Mexico_City
Client Name	• Bimbo Store, … , Walmart

- Dropped columns: Sales Channel ID, Route ID, Product Name, Town

```
table5["Product_Name"].value_counts()
```

```
Pan Blanco 640g BIM 2233                    34496
Nito 1p 62g BIM 1278                        34471
Donas Azucar 4p 105g BIM 1250               33934
Mantecadas Vainilla 4p 125g BIM 1240        31643
Madalenas 3p 93g BIM 35651                  29534
```

Visualizations



Factor Plot on Adjusted Demand by Client Name

Modeling Approach

- Logistic Regression Model:
  - Categorical nature of the data set.
  - Logistic Regression also returned stronger evaluation metrics than Linear Regression and Random Forest.
- Binarizing Adjusted Demand:
  - Logistic Regression requires that Adjusted Demand be binarized.
  - Since Adjusted Demand typically ranged between 0 and 8 units and binarizing on returns or no returns would have be too heavily skewed towards returned product, I determined all values greater than or equal to 4 would be 1 and all values less than 4 would be 0.
  - This led to a 55/45 split between high returns [1] and low returns [0].
- Top-Down Approach:
  - Fit a model with all predictor columns and then eliminated predictors that weren't statistically significant.

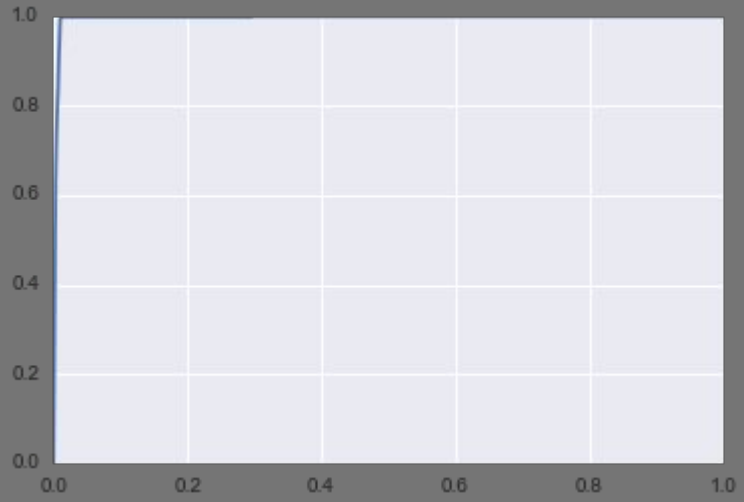
Logistic Regression Results

Variable	Coefficient	Odds Ratio
Depot_1110	-6.5651	0.001409
Depot_1111	-7.7400	0.000435
Depot_1112	-8.8934	0.000137
Depot_1113	-6.5452	0.001437
Depot_1114	-6.8199	0.001092
Depot_1116	-10.7049	0.000022
Weekly_Sales_Units	2.4798	11.938657
State_Mexico	-2.3605	0.094370
Walmart	-2.2708	0.103233

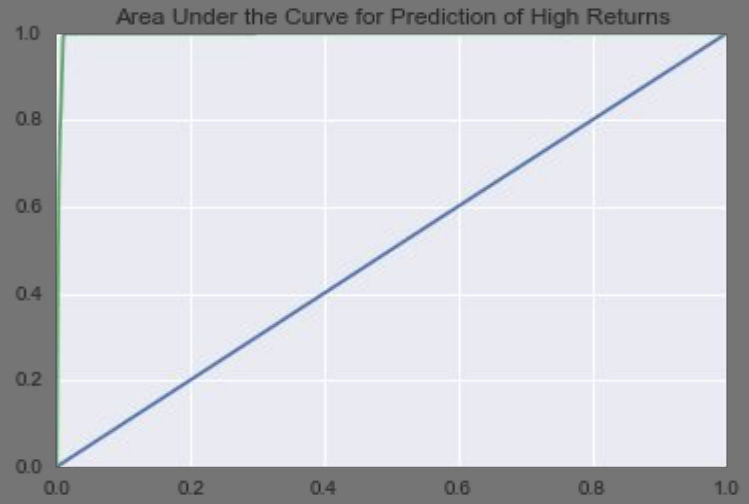
- Walmart is the only client grouping that is statistically significant.
- Though the binarized Depot variables are statistically significant, they marginally increase the odds of a high return.
- Walmart and State_Mexico only increase the odds of a high return by ~10%.
- The odds ratio of the most significant predictor, Weekly_Sales_Units, indicates that for each additional unit sold, the odds of a high return increase by ~1193.9%.

ROC-AUC Score and Accuracy

ROC-AUC Curve (Score = 0.99453)



Accuracy



Analyzing Results and Potential Next Steps

- Based on the Odds Ratio values, ROC-AUC score, and Accuracy graph, I believe the model results are too good.
- Reasons why:
  - By only using the first 250,000 rows from the train dataset, the dataset was transformed from a time-series to a cross-section.
  - Though Weekly_Sales_Units proved to be the predictor with the greatest impact on the likelihood of high returns, it's impact is skewed. Since it is likely that the greatest predictor of higher returns is high returns in the past, the model less predictive than the strong evaluation metrics suggest.
- Potential Next Steps:
  - Shrink column bit sizes to shrink data
  - Get PostGresSQL working
  - Do further TF-IDF inquiry on the Product_Name category
  - Heat mapping Adjusted Demand on map of Mexico
- Words of wisdom: Be careful with large data sets