Executive Summary

Stocking brick and mortar department stores requires a breadth focus approach. A wide range of products must be available to customers to peruse. However, from a business perspective, excessive breadth could lead to unmoved inventory. To address these challenges, we developed a Random Forest Regressor model that predicts anticipated discount rates with an RMSE of 0.2115 and an R^2 score of 0.4753. Further advancements in model design, data collection, and integration of data from third party sources have potential to improve predictive performance and robustness. We aim to apply a future iteration of this model when devising a stocking strategy for Dillard's, opting to not invest in products that require high discounts to sell, in turn saving inventory and logistics costs.

Introduction To The Business Question

With the rise of e-commerce and the developing norm of fulfilling customer orders through distribution centers, the position of the traditional brick and mortar stores in the eyes of consumers has become less and less essential. Combined with the increasing accessibility of online specialty stores that focus on depth over breadth, the role of the department store has become less stable. Despite these market shifts, one undeniable value of department stores is the physical availability of stocked products. However, this poses a unique challenge for retailers such as Dillard's: how would one go about deciding what products to stock, and at what quantities? We propose an approach that focuses on the in store discounting of products. By identifying which products require heavy discounting to sell, the company will be able to anticipate potential demand and incorporate this factor into inventory planning, enabling cost savings and reducing inventory risk.

Getting Data Into The Cloud

Before starting the project, we knew we needed a centralized way to access and run queries to enable our team to operate more efficiently. To do this, we chose to use the SQL server provided by the MSiA program. For this project, we were given five tables (STRINFO, SKSTINFO, SKUINFO, TRNSACT, and DEPTINFO) that could be combined using SQL queries. To upload the data onto the cloud server, we needed to write the names of the columns, their data types, and whether or not they were a primary key. This turned out to be a bigger task than anticipated.

The database diagram that was given to us was wrong for the TRNSACT table. This is the largest file in the database at about eleven gigabytes. This rendered us unable to open the entire file in a text editor and even in pandas. To fix this, we were able to load only a portion of the data into pandas to investigate. After loading the first one million rows into pandas and looking at examples of the data in each column, we were able to deduce what the real names of the columns were. The order given to us on the schema was; SKU, STORE, REGISTER, TRANNUM, SALEDATE, SEQ, INTERID, STYPE, QUANTITY, ORGPRICE, AMT, and MIC. It turned out that the real order of the columns was SKU, STORE, REGISTER, TRANNUM, INTERID, SALEDATE, STYPE, QUANTITY, AMT, AMT2 (Clone of AMT), SEQ, MIC and an unknown binary column. After discovering this, we were able to quickly upload the table to the cloud.

Another challenge we faced occurred while we were trying to upload the SKUINFO table to the PostgreSQL cloud server. The .csv files are separated by commas, and that is how the csv reader functions are able to distinguish between data points within rows and columns. Some of the vendor names in the csv file had commas in them. When the csv reader functions read in the data, they read the rows with those vendors as having more data. To fix this, we selected the rows where there were more than eleven (the correct number of commas). There ended up being about 9000 rows out of 1.2 million. Since this is a small proportion of the total number of rows, we dropped them as we are not losing much data relatively.

Data Cleaning, EDA, & Feature Engineering

Since we are interested in the sales of items at discount, we have created a new variable “discount” using data from the TRNSACT table and the SKSTINFO table. It is calculated as the percent between the original price and the retail price, divided by 100. Afterwards, as we wished to reduce redundancy in the “sku”, we grouped by “sku” and then aggregated the aforementioned "discount" column by mean value, before joining this new average “discount” column with the SKUINFO table. The SKUINFO contains the features that we will use for our predictive model. While we initially wished to use the wealth of data from the TRNSACT table, we discovered that, unfortunately, most features there would cause leakage, as an existing transaction theoretically can only occur under conditions where price and discount amount are already known, making a predictive model unnecessary.

This new data set now contains the following columns: the unique identifier and stock

keeping unit of each product as “sku”, original price as “orgprice”, the calculated average discount proportion as “discount”, the department to which each product belongs as “dept”, the type of product as “classid”, the product barcode as “upc”, the manufacture assigned style information as “style”, the original color specified as “color”, the designated product size as “size”, the number of items per unit as “packsize”, the vendor of the product as “vendor”, the brand of the product as “brand”, and two unknown columns, “unknown” and “unnamed: 0”. We continued to clean our data by dropping irrelevant columns: “unnamed: 0” and “unknown” as they are not recognized.

After that, we focused on transforming the “color” and “brand” to help the data better fit into our model. For the color feature, we wanted to find the most popular colors, but we also noticed that several different names or terms could be used to describe one single color at the same time, so we created a new column with color group and designed a color list containing the 14 most popular color groups: “white”, “black”, “green”, “natural” “purple”, “red”, “blue”, “multi”, “pink”, “brown”, “charcoal”, “yellow”, “orange”, and “undefined” as “others”.

As for the brand, we transformed brands into 5 different levels based on the average price of each level of the group. Firstly, we grouped the data by “brand”, and aggregated them by the mean of their original price. Then, we sorted the different brands by this mean price and grouped them into 5 buckets of price levels, as “level”. In this way, we made brands a feature that can be associated with the customers’ consumption level.

We checked the collinearity of our data by making a table of the correlations of each variable. We found that the correlation between original price and “level” of brand price is 0.47, which is explained by the fact that we transformed the price value of different brands to form the new variable “level”.

To fit the models for predicting the discount, we discussed potential data issues and selected several features from the SKUINFO table that are the most reasonable in this situation. We first excluded variables like “sku” and “upc” because they work as ids for specific products. Including them would cause data leakage and overfitting. Next, we decided to exclude the variables “style” and “size”. At first, the “style” and “size” features seemed to intuitively carry the information about the product, that is, the information is helpful for fitting the model. However, after the exploratory data analysis, we found style and size are too messy to be cleaned or transformed like variables color and brand, so in this project, we did not pick these two

variables. Thus, the features we eventually decide to choose are: “dept”, as it provides a reasonable initial classification of the product department, “classid”, as it could inform the model what type of product is being considered, “color group” (cleaned color), as it offers an approachably sized set of colors to consider, “vendor”, as it allows us to consider the value of products from different vendors, and “level” (cleaned brand), as it is a reasonable clustering of brand prestige and level of luxury. In consideration of the business application of our model for Dillard’s, these features should all be available to us when presented with a new product to consider stocking.

After selecting these features, we then performed feature transformation. Note that the features we chose are all categorical. That is, even though some of them are float or integer (like “dept”, “classid”, etc.), we still need to perform one hot encoding (and ordinal encoding in the case of “level”, as there is a reasonable ranking to the level of luxury of brands). to transform them for use in our models.

Modeling & Model Evaluation

After feature engineering all the variables, we are ready to fit the model. In this project, we fitted three different types of regression models (linear regression, random forest regressors, KNeighborsRegressor) to compare the performance of prediction. We first declared the dependent variable y as discount, and declared the transformed features as independent variable X. Then we performed the train_test_split method to split the data into X_train, X_test, y_train, y_test sets.

Our base model is linear regression. We first created an instance of a linear regression model using the sklearn package. We fitted the model using X_train and y_train, and made predictions of ypred using X_test. After making the prediction, we used RMSE and R^2 score to evaluate the model. It showed that linear regression has 0.2904 RMSE and 0.0114 R^2 score.

The second model we used was Random Forest Regressor. Following the same steps, the Random Forest Regressor has 0.2115 RMSE and 0.4753 R^2 score.

The last model was KNeighborsRegressor. The result showed that KNeighborsRegressor has 0.23 RMSE and 0.3795 R^2 score.

Compared to the base model linear regression, we found that random forest regressor made better predictions on discount since it had the lowest RMSE and highest R^2 score.

ROI Analysis

We observe that not only are roughly 53% of products sold at discount, but that the average discount percentage is over 48%. This translates to an average profit margin of only about 3.6% for products sold at discount. In contrast, when products are sold at full price, profit margins are at an average of about 50.5%. With this in mind, there is a significant benefit to reducing the share of products that are sold at discount. If our model is able to correctly identify 0.01% of the products offered by Dillard's as requiring a discount to sell, and we replace these with alternative products that sell at full price, we would see a 0.02% lift in profit, which amounts to over $550,000. Consider that the cost of investment in this project is under $360,000, we can expect an ROI of over 55%. If we increase our confidence to say we can correctly replace 0.03% of the products sold at discount, projected lift becomes 0.06%, amounting to over $1,540,000, which is an ROI of over 330%. We deduce that, with any marginal reduction in the stocking of products that require discounting to sell, we will gain positive returns on our investment.

Risks

The risks associated with undertaking this project are relatively minimal. If the model were to repeatedly incorrectly identify products as having a low discount requirements, then stocking could be affected negatively, leading to reduced profitability (due to lower profit margins on discounted products). However, this risk could be reduced by adjusting decision cutoffs (e.g. predicted discount > 0.3 versus predicted discount > 0.5), and by augmenting predictions with human judgment.

Conclusion

Ideally, this model will be used to supplement the judgment of merchandising managers. When a new product is being considered for purchase from a vendor, the user could input existing information about the product into our model, and see a predicted average discount percentage for future sales. If the discount percentage is considerably higher than anticipated, it could be an indication that stocking the product would be unwise. So long as these predictions improve stocking decisions by 0.01%, there will be positive returns on investment. For future iterations of this model, we could look to incorporate third party data (such as consumer interest, demand, and trends) to increase features and enhance the predictive power.