Table of Contents

FINAL REPORT

MSiA 400 Project - Team 4: Cindy Chen, Ziyan (Cheryl) Liu, Weiyan Zhou, Ruben Nakano

Executive summary

This report presents the process to build a model that identifies products (SKUs) that will need higher discounts in order to be sold, and that therefore will result in lower profit margin for Dillard's. If we can identify these underperforming products then the company can better optimize its inventory and increase profits. The final model has a R2 of 48% and results in a Return on Investment of approximately US$ 6MM annually (rate of ~1500%).

Introduction and Objectives

Dillard's is a department store chain that currently has stores spread out in 29 states. Being a large retailer, it is constantly having sales, deals and discounts so the company can attract new (and repeating) customers and quickly drive sales. Additionally, reducing profit to move excess and outdated inventory is one of the key reasons for the company to have deals. The product characteristics, in particular brand, vendor, color, and size, have great impacts on its popularity. As is known to all, it is hard for a company to hold popular products, so they usually don't get discounted. However, unpopular or overpriced products need discounts in order to be sold. In order to minimize revenue loss and reduce excess inventory cost, Team 4 built a model to determine what products will need discounts based on the product's characteristics and features.

Exploratory Data Analysis

The Dillard's dataset has five different csv files that amount to a total of 12.5 GB of data. We have five tables which are *deptinfo*, *strinfo*, *sksrtnfo*, *skuinfo*, and *trnsact*. There are a total of 60 different departments which have stock items in the *deptinfo* table. From the *strinfo* table, we know 453 different Dillard's stores with their location information like city, state, and zip code. In the *sksrtnfo* table we have information about the total number of stock items for each store along with its stock and retail price.
We also have a data set on each Stock Keeping Unit (SKU) with information that includes the department it belongs to, classification id, universal product code, style, color, size, packsize, vendor and brand. Regarding data on transactions we have a total of 120,916,896 rows with information on the item's stock keeping unit number, unique transaction number, sequence number, internal ID, master item code, type of transaction quantity, original price, and sale price for each transaction.

Although this is not large enough to be considered "big data", the size of this data set can still pose challenges for data analysis in Pandas or similar tools. Additionally, performing exploratory data analysis (EDA) on database systems such as PostgreSQL or MySQL can be cumbersome. To solve this problem we chose to initially import data to PostgreSQL and then to randomly sample rows to perform EDA on a smaller subset of the dataset. We randomly sampled the data and joined the tables *trnsact*, *strinfo* and *sksrtnfo* to get the characteristics of each SKU to allow us to properly answer the business question.

Since the goal of the model is to predict discounts, we spent a majority of the time exploring this variable. The discount amount is defined as the difference between the original price and the price sold (variable ‘amt’ in the data set) divided by the original price (Table 1 in the Appendix shows an example of products with different original and sale prices).

To understand how discounts are distributed we then plotted a histogram, where we observe that most products have less than 20% discount (Figure 1). It is also worth noting that the discount values of 50% and 75% are more popular than random unrounded values. In addition, we also explored how discounts might be related to important characteristics of a product, like its color and size (Figure 2) or brand (Figure 3). Finally, we also plotted the discount percentage against the original price of a product (Figure 4) and observed that there is a trend that products with lower price tend to have bigger discounts.

Since we are interested in products with big discounts we added a column "if_big_discount" to the dataset to flag large discount values (greater or equal to 50%). From Table 3 in the Appendix we see that products with big discounts represent over 35% of items that are discounted. Finally, we computed an average discount and a percentage of “large discounts” for every SKU (Table 4) by grouping by SKU.

Feature selection and cleaning

Before starting modeling we focused on cleaning the data and choosing the appropriate features for the model. To do so, we initially got rid of columns with unnecessary information such as ***‘upc’***, ***‘packsize’*** and ***‘style’***. Then we grouped the dataset by the remaining variables: ***’brand’***, ***‘vendor’***, ***‘dept’***, ***‘classid’***, ***‘color’*** and ***‘size’*** - this resulted in 186410 groups out of the 266225 rows. The number of unique values for each feature is as follows:

- ***‘color’*** has 23,888 types
- ***‘size’*** has 3,365 types
- ***‘classid’*** has 60 types
- ***‘dept’*** has 807 types
- ***'vendor’*** has 1,374 types
- ***‘brand’*** has 1,134 types

We then proceeded to clean the features ***‘size’*** and ***‘color’*** since they have the largest amount of unique values (a reasonable amount of these values were not actually unique - e.g. “NAVY” and “CL NAVY”).

1. Size cleaning
    a. We only select out non-numerical size first, and then wrote list comprehension to extract only 'ALL', 'L','M','ONE', 'S', 'XL', and 'XS' for size
    b. since the numerical value of size is inconsistent for different types of items.
2. Color cleaning
    a. We first defined a color system which includes the majority of popular colors from our items. There are 13 colors and they are BLACK, BLUE, WHILTE, PINK, RED, MULTI, SILVER, GREEN, NOCOLOR, GOLD, BROWN, YELLOW and OTHER.
    b. We then use regular expressions to map our current color of each item to one of the levels from our color system. Here are some examples of color grouping
        i. BLACK, BLAK, xx -> black
        ii. NAVY, DARKBLUE->blue
        iii. CHOCO -> other
    c. For the color of items which cannot be classified to our color system, we marked them as “other” for the cleaned color.
    d. After re-grouping, there are 13 levels of the color column and they are balanced and clean.

```
other      62416
BLACK      14013
BLUE        8494
WHITE       7025
PINK        4887
RED         4345
MULTI       2810
SILVER      2042
GREEN       1949
NOCOLOR     1389
GOLD        1378
BROWN       1212
YELLOW      1069
```

<u>One-hot encoding</u>
For “vendor”, “classid”, and “dept”, we applied basic one-hot encoding techniques.

Modeling

The table below contains the summary of the models that we tested.

Table 2 Modeling results and hyperparameters

Model / Metrics	Decision Tree	Random Forest	XGBoost (base model) XGBRegressor with objective='reg:squarederror'	**XGBoost (final model)**** XGBRegressor with objective='reg:squarederror'
RMSE (train)	0.197684	0.205070	0.238745	**0.219994****
RMSE (test)	0.258360	0.248859	0.243273	**0.235858****
R^2 (train)	0.681214	0.650584	0.467211	0.579880
R^2 (test)	0.284971	0.384112	0.430557	0.484026
Hyperparameter choice	No tuning with default hyperparameter	n_estimators **=** 5, random_state **=** 42	random_state = 2, tree_method = 'hist'	n_estimators = 2500, random_state = 22, learning_rate = 0.18, colsample_bylevel = 0.8, max_depth = 5, tree_method = 'hist'

<u>Decision Tree</u>
The first model we tried was Decision Tree (Regression Tree). The basic logic of building a decision tree is to construct multiple if-else statements as branches that can be used to predict

a numerical result based on available data. The advantage of this model includes great explainability and automatic feature engineering. However, since we only constructed one single tree for our prediction, it is prone to overfit and unstable for new unseen data. Our model performance metrics also indicate that there is a problem of overfitting: the R^2 on the train set is 68% but only 28% on the test set. RMSE shows a similar trend. We did not do any hyperparameter tuning and we treat it as our baseline model.

Random Forest

We then tried random forest as our second model to avoid overfitting. This model consists of a large number of individual decision trees that operate as an ensemble. By taking the majority votes from multiple decision trees as final predicted value, random forest has some ability of reducing overfitting issues. Also, it is robust to outliers and more stable. The R^2 on the train set is 65% but 38% on the test set. It indicates that we still need to improve the overfitting issue.

XGBoost Model

In order to prevent overfitting, we select XGBoost as our model, which is a decision-tree-based ensemble Machine Learning algorithm that uses a Gradient Boosting framework. Since the target variable (*discount percentage for each sku*) is continuous and the tree is used to predict its value, The XGBoost model for regression is called XGBRegressor. So, we will build an XGBoost model for this regression problem and evaluate its performance on test data (unseen data/new instances) using the Root Mean Squared Error (RMSE) and the R-squared (R^2-coefficient of determination). All the prediction results and hyperparameter choices are in Table 2.

The performance evaluations are shown in Figure 5 - 8 in Appendix. Prediction error plot: We can not see a clear pattern that most of the points are on a straight line. Residuals plot: We cannot see any pattern between predictions and residuals. So, we can verify that the residuals are uncorrelated or independent. This is a good sign for our model. Distribution of residuals plot in Figure 8: By looking at this plot, we can verify that the residuals (actual values-predicted values) are approximately normally distributed for the XGBoost (final model).

Return On Investment analysis

The following assumptions were made to calculate Return on this project:

- Dillard's has very limited stocking and inventory space, therefore they need to choose which products to stock and only keep the best performing products (i.e., the ones that yield higher profit margins)
- Currently, Dillard's have a baseline model that predicts which SKUs will need higher discounts in order to be sold (we assume it's the Decision Tree model we first fitted)
- Each time they receive new products they run this baseline model to predict which ones they don't need to stock and also decrease the production rate of these underperforming products by 50%
- They determine which products are underperformers by selecting the top SKUs (5000) ranked by their predicted discount amount in descending order
- If the model predicted correctly, these products would have amounted to a loss for the company because they would need heavy discounts in order to be sold. By decreasing their production rate the company would minimize this loss.
- These products might even be sold for a profit in the end, but the company doesn't have

enough inventory space to keep all products, so it needs to choose which ones to keep

- With our proposed model, Dillard's will be able to better identify underperforming products and thus reduce their losses even more (or reduce the profit they "left on the table" if the model predicted incorrectly)

Investment assumptions:

- We assume 4 data scientists, 1 data engineer, 1 business analyst and a manager (part time) will have to work in the duration of the project (3 months)
- We also assume 1 data scientist, 1 data engineer and a business analyst (part time) will have to keep working on model maintenance to make sure the predictions make sense, to update the infrastructure, etc.
- Cloud computing costs for the whole period the model will be active and no licensing costs since all tools used are open-source

With these assumptions our final annual Return is US$ 7,281,159, our annual Investment is US$ 419,025 and thus, ROI is: US$ 6,862,134 (rate of ~1500%). Details of the calculations are in the spreadsheet.

Risks

Model Risk: overall, the training set prediction has higher R^2 and RMSE than the test set prediction. There's overfitting in the model we've built. The company should work on improving the model in the future and minimize the performance difference between train set and test set if possible.

ROI Risk: the ROI analysis was based on the difference in profit between the baseline and proposed model on test data and then extrapolated to company-level profit. It would be advisable to make further tests to confirm the result. Additionally, the maintenance cost might be higher if changes need to be made on the model (add new features, tune hyperparameters, etc).

Conclusion

The project objective to make a model to help Dillard's determine which products will need discounts in order to be sold was achieved. The final model has an R2 of 48%, which we believe is reasonable for this type of application. Compared to the baseline model, the proposed model has an ROI of approximately US$ 6 MM based on our assumptions.

As a class assignment, the project was a good exposure to "real world" messy data and the challenges of having to come up with a model to solve an open business problem and then quantify its value.

References

1. Salary of Dillard's Employee:
https://www.comparably.com/companies/dillards/salaries/data-scientist

2. Annual business report of Dillard:
https://investor.dillards.com/financial-information/annual-report-and-proxy/default.aspx

Appendix

Table 1. Different price between sale and original

	Original Price	Price
0	44.0	29.99
1	16.5	16.50
2	44.0	44.00
3	30.0	30.00
4	45.0	7.87
...	...	...

Figure 1. Histogram of discount

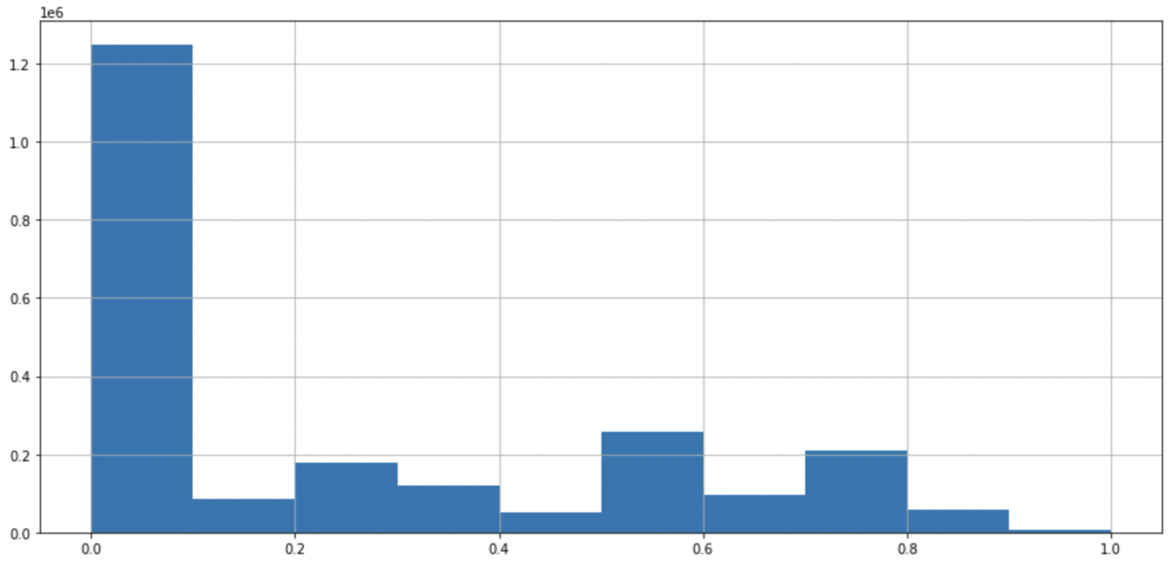


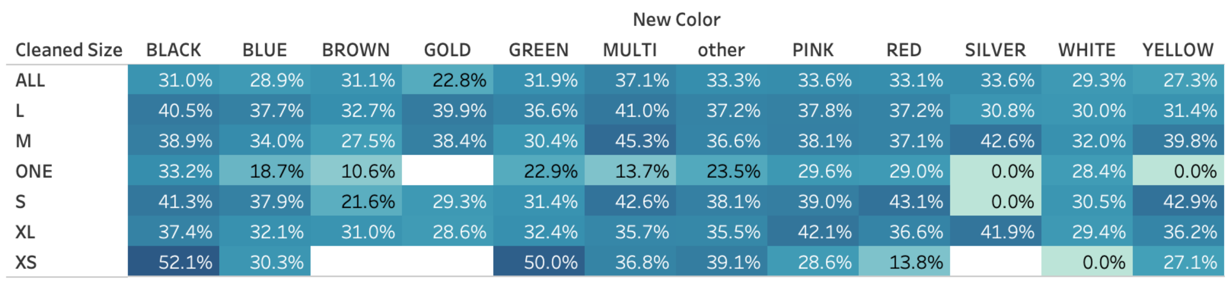
Table 3. Number of "big discount (over 50% off)" products

Not big discount (< 50%)	Big discount (> 50%)
1717662	626338

Table 4. Different price between sale and original

sku	if_big_discount	discount
3	0.50	-inf
78	0.00	0.000000
156	0.00	0.000000
180	0.00	0.000000
268	0.00	0.000000
...	...	...
9999926	0.00	0.000000
9999933	0.25	0.174450
9999950	0.00	0.000131
9999956	0.00	0.000000
9999974	0.75	0.375000

Figure 2. Average discount by color and size



	New Color											
Cleaned Size	BLACK	BLUE	BROWN	GOLD	GREEN	MULTI	other	PINK	RED	SILVER	WHITE	YELLOW
ALL	31.0%	28.9%	31.1%	22.8%	31.9%	37.1%	33.3%	33.6%	33.1%	33.6%	29.3%	27.3%
L	40.5%	37.7%	32.7%	39.9%	36.6%	41.0%	37.2%	37.8%	37.2%	30.8%	30.0%	31.4%
M	38.9%	34.0%	27.5%	38.4%	30.4%	45.3%	36.6%	38.1%	37.1%	42.6%	32.0%	39.8%
ONE	33.2%	18.7%	10.6%		22.9%	13.7%	23.5%	29.6%	29.0%	0.0%	28.4%	0.0%
S	41.3%	37.9%	21.6%	29.3%	31.4%	42.6%	38.1%	39.0%	43.1%	0.0%	30.5%	42.9%
XL	37.4%	32.1%	31.0%	28.6%	32.4%	35.7%	35.5%	42.1%	36.6%	41.9%	29.4%	36.2%
XS	52.1%	30.3%			50.0%	36.8%	39.1%	28.6%	13.8%		0.0%	27.1%

Figure 3. Average discounts by brand

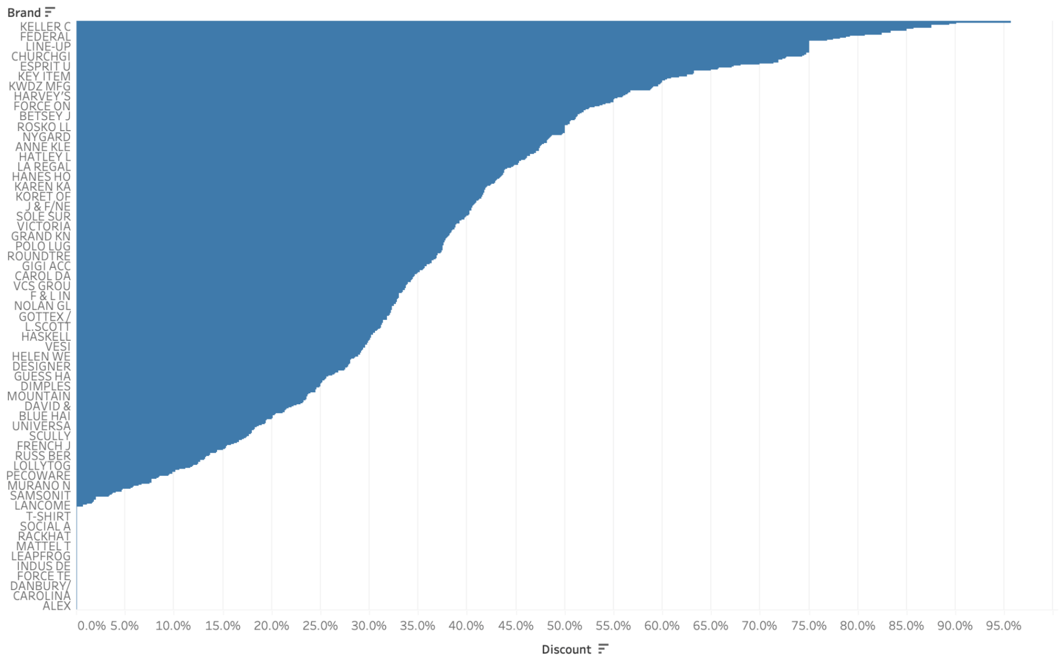


Figure 4. Average discount x Original price (log scale)

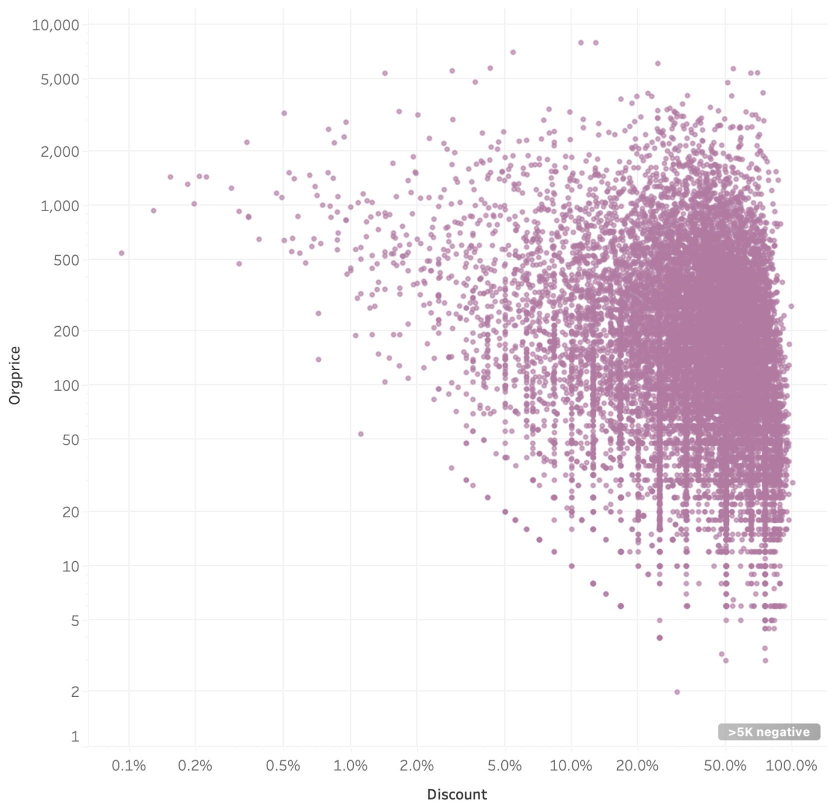


Figure 5. Prediction Error Plot for XGBoost (base model)

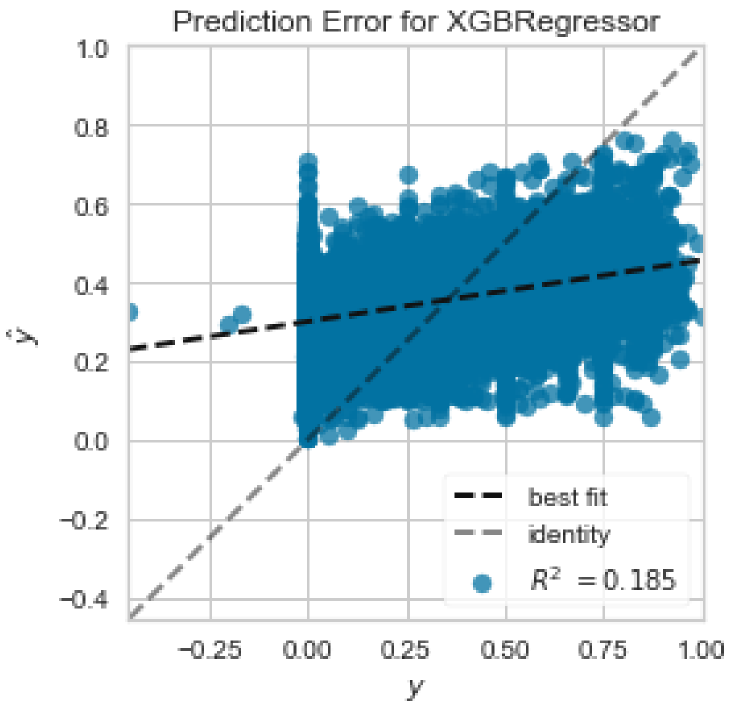


Figure 6. Residuals Plot for XGBoost (base model)

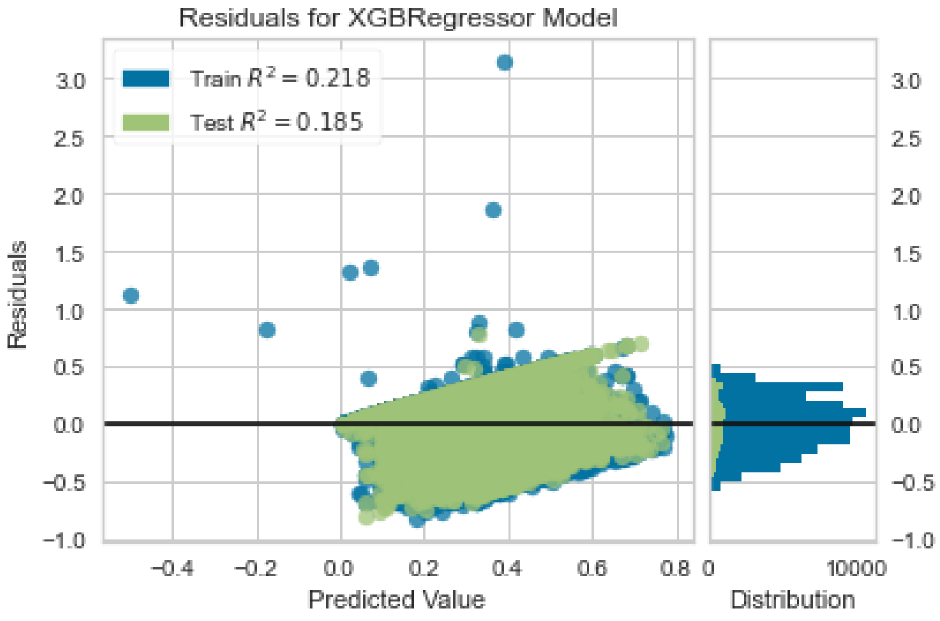


Figure 7. Prediction Error Plot for XGBoost (final model)

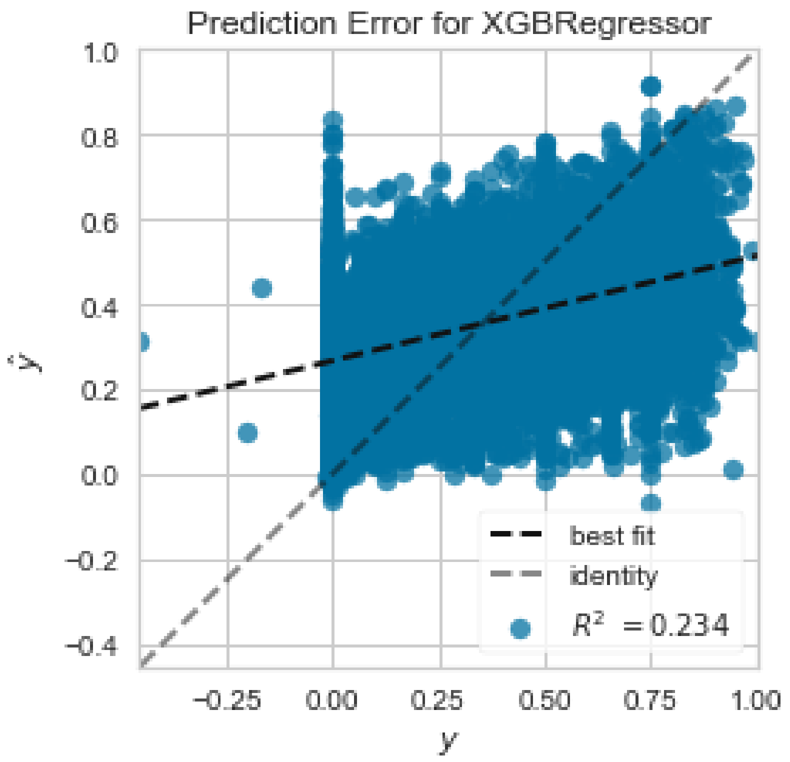
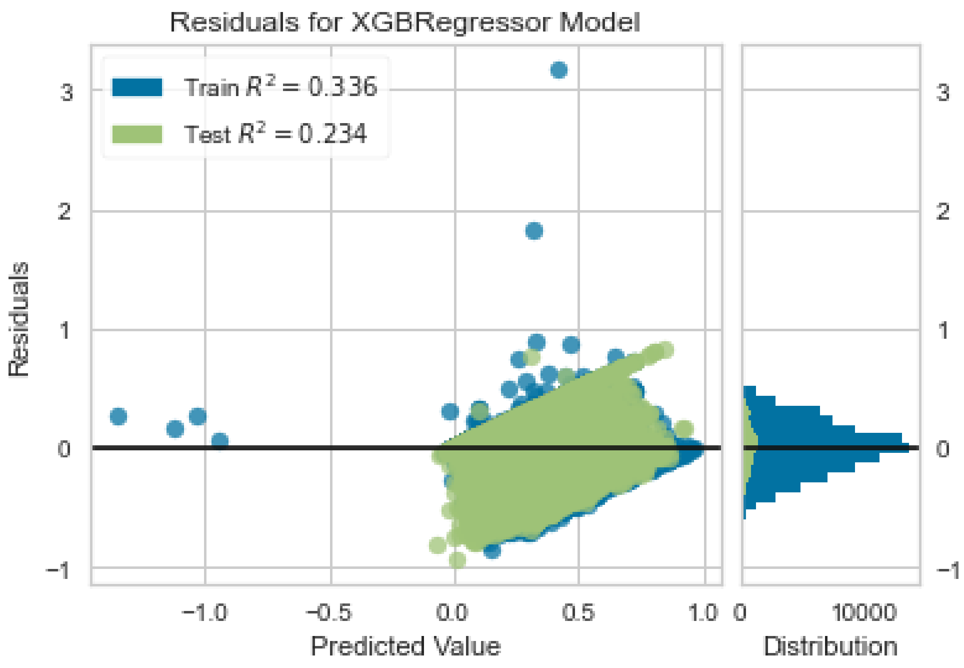


Figure 8. Residuals Plot for XGBoost (final model)


Residuals for XGBRegressor Model
Train $R^2 = 0.336$
Test $R^2 = 0.234$
Residuals
Predicted Value
Distribution