_ID = 2051 of data bau _ID = 2052 of data bau _ID = 2053 of data bau

VIETNAM NATIONAL UNIVERSITY – HO CHI MINH CITY
INTERNATIONAL UNIVERSITY
DEPARTMENT OF MATHEMATICS

GRADUATION THESIS

MACHINE LEARNING FOR RETAIL PRICE OPTIMIZATION

Submitted in partial fulfillment of the requirements for the degree of
BACHELOR OF SCIENCE
IN APPLIED MATHEMATICS
SPECIALIZATION IN FINANCIAL ENGINEERING
AND RISK MANAGEMENT

Student's Name: Luong Do Van Quyen
Student's ID: MAMAIU18057
Thesis Supervisor: Assoc. Prof. TRAN VU KHANH, PhD.

Ho Chi Minh City, Vietnam
March 2023

MACHINE LEARNING FOR RETAIL PRICE OPTIMIZATION

By

Luong Do Van Quyen

Submitted to DEPARTMENT OF MATHEMATICS,
INTERNATIONAL UNIVERSITY, HO CHI MINH CITY
in partial fulfillment of requirements for the degree of
BACHELOR OF SCIENCE
IN APPLIED MATHEMATICS
SPECIALIZATION IN FINANCIAL ENGINEERING
AND RISK MANAGEMENT
March 2023

Signature of Student: _____

Luong Do Van Quyen

Certified by: _____

Assoc. Prof. TRAN VU KHANH, PhD.
Thesis Supervisor

Approved by: _____

Prof. Pham Huu Anh Ngoc, PhD.

Acknowledgements

First of all, I would like to express my deepest and most sincere gratitude to Assoc. Prof. Tran Vu Khanh, PhD - who wholeheartedly guided and guided me during the implementation of the thesis. He not only provided valuable materials but also guided me in the right direction so that my thesis could be completed in the best way. I think if it wasn't for his dedication during the past process, I probably wouldn't have done my thesis well. In short, from the bottom of my heart, I sincerely thank Mr.Khanh for agreeing to accompany me for the past few months.

Additionaly, I would like to express my sincere thanks to the teachers of Mathematics Department, International University, National University, Ho Chi Minh City for their dedicated teaching and equipping me with valuable knowledge during the four years. Thanks to that knowledge, I can apply it well in life and do better in my thesis.

Finally, I would like to sincerely thank my grandparents and parents who have always supported both materially and spiritually during the past time. I would like to thank my friends and colleagues for their help and support during the course of the thesis course as well as during my studies at IU.

Even though I made an effort to finish the thesis within my capacity, there will inevitably be errors. I look forward to receiving the sympathy, advice, and enthusiastic instruction of professors and friends.

Abstract

In today's developed economy, setting a selling price to maximize profits is always a top priority for businesses. Businesses cannot set a price they like for a certain product to sell to customers, but this must be based on many different factors. In this thesis, we use a variety of machine learning models to optimize prices and then compare their results to determine which one is most accurate. We use two basic methods to forecast the overall sales at Burger Coffee Shop: Linear Regression and Tree-based method. Then, we use four metrics to the evaluate models, in which, they are Mean Absolute Errors (MAE), Mean Squared Error (MSE), Root-mean-square error (RMSE) and R-squared (R^2). In the rest of this thesis, we calculate the elasticity and set the optimal price for each product.

Key words: Price optimization, Linear regression, Random forest, Decision tree, XGBoost, Burger coffee.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

Vietnam has emerged as one of the world's most appealing Food And Beverage (F&B) markets, according market research company BMI [1]. Statistics show that the food and beverage sector has contributed 15.8% of the country's GDP (in 2021). The largest percentage, at around 35%, of total spending was on food and drink. Particularly, Vietnam's F&B business is anticipated to expand due to the widespread availability of food services, tourism-stimulating regulations, and efforts to attract tourists. All will help to encourage the F&B sector's overall growth and the firms inside it in particular. Consequently, one of the greatest problems for all organizations is to determine the appropriate pricing for each of their items, so as to optimize revenue and profit. Because pricing is still one of the most crucial variables affecting a consumer's purchase choice, which directly impacts a retailer's income. Retailers are always aware that in order to draw customers, they must not only provide high-quality products, but also enticing prices and promotions at the appropriate times. For instance, they must choose attractive prices for goods with high purchasing power during the peak season. During the off-season, they must determine how large of a discount is necessary to ensure that the previous season's inventory is eaten while profit is maximized. Businesses must create a clever selling pricing strategy for this. However, because traditional pricing is frequently dependent on individual experience, it can be difficult for merchants to develop a wise pricing strategy. Due to several aspects, including competition analysis, weather, shopping season, operational expenses, regional demand, and corporate goals, this procedure typically takes a long time but is not very efficient [2].

Currently, with the strong development of technology in the digital age, especially the techniques of Machine Learning and Deep Learning [3], the need to optimize profits

based on the retailer's smart pricing plan was met with ease and efficiency [4]. Retailers can better set prices by using price optimization to predict how customers will respond to various pricing methods for goods and services. Machine learning models can find the optimal rates, even for huge catalogs of goods or services, that can meet the predetermined KPIs by taking important pricing factors into consideration (such as purchase histories, seasons, inventories, and competitors' pricing).

In comparison to conventional pricing methods, machine learning-based algorithms can evaluate far larger data sets and take into account a much greater number of variables [5]. Prior to now, pricing managers had to decide on pricing regulations manually. In contrast, machine learning models employ algorithms that semi-automatically learn from their outcomes over time. Retailers can employ machine learning models to determine prices in relation to sales targets as a result. They can accomplish this completely automatically, with far greater accuracy, and with a lot less work.

In addition to learning, machine learning-based pricing algorithms improve with time at determining the best price points for businesses by identifying the sweet spot between "too cheap" and "too costly." Additionally, the algorithms used by these machine learning-based pricing solutions can take into account both important internal and crucial external data. In addition to processing far larger and more varied datasets than prior technologies, this enables them to determine prices that are extremely precisely in respect to these important data points. Machine learning-based pricing algorithms may determine price elasticity using a variety of variables, which quantifies how demand will change in response to environmental changes [6]. The software then makes the necessary price adjustments. These technologies can help identify which items' demand is comparatively constant, making them suited for margin optimization, or those that are crucial to overall sales and should only be judiciously altered.

The purpose of this thesis is to determine how to set optimal price accuracy after calculating and choosing the most suitable model to determine the price. In this thesis, we will divide process into three parts: The Prediction Model, Elasticity Of Price, and The Price Optimization. The data we use from a Burger Cafe of Microsoft building in China, based on Xueshan Zhang's Research [7].

First, The Prediction Model part helps us to choose the most accurate model to apply the price elasticity calculation. Here, we choose 4 main approaches in this section, namely Linear Regression, Random Forest, Decision Tree, XGBoost [8]. Based on each

product's prior sales, the model is trained on it. The value of the quantity sold is on the output label. After applying machine learning models, we will use metrics to evaluate their accuracy. We will base on their results and the graphs they represent to select the best results. We use 4 metrics: MAE, MSE, RMSE, R^2.

Next, after we choose suitable model and data, we will calculate "Price Elasticity of Demand" (PED) - a concept of micro economics. Changes in demand as a result of a shift in price are captured by the concept of price elasticity. By doing this, we are able to derive many sets of price-demand relationships for each commodity. There must be a single pricing for every product where the total revenue is maximized by the specified configuration. In The Price Optimization section, the Linear Programming optimization approach is employed to solve it. In section [3.6], a thorough explanation of the formulation of linear programming problems and their constraints may be found.

The remaining parts of this thesis are arranged as follows. In Chapter 2, we briefly address the associated work. In Chapter 3, we present the methodology, which consists of The Prediction Model, Elasticity Of Price, and The Price Optimization. Data are discussed in Chapter 4 and the result are summarized in Chapter 5.

Chapter 2

LITERATURE REVIEW

Pricing optimization is becoming more important as competition heats up, driving insurers to improve ratings and take customer behavior into account in the majority of established insurance markets [13].

In this part, we conducted a quick literature review on Price Optimization. According to [9], authors use K-means clustering to divide customers into groups, and then use a regression model to set prices for each group. With the assistance of past user cluster and price range data, the writers are now attempting to determine the possibility that a client will purchase a product. The likelihood of a customer is computed using the logistic regression model. While in conventional fashion e-commerce the exact price point of each product is needed, this model provides a price range at the level of the user cluster.

The Carlson Rezidor Hotel Group is a another real-world example of sophisticated pricing optimization. In this instance, the pricing optimization model was determined by market (competitor) prices, room availability, predicted demand (based on historical data on hotel visitors' length of stay, rate segment, day of the week, and lead time), and business rules. According to the proposed pricing optimization method cited in [10], cooperating hotels witnessed an increase in revenue of 2-4% when compared to non-compliant hotels.

The authors of Paper [11] have described the ideal pricing suggestion mechanism for the Airbnb platform's hosts. First, they used a binary classification model to forecast each item's likelihood of being booked on the site; then, they used a regression model to

forecast the best pricing for each offering for a given night. To create the final pricing recommendation to the hosts for their property, they add a customisation layer last. In a traditional fashion e-commerce arrangement, we must select a certain price point for each product; but, in this case, there is a pricing range.

In order to more fully and accurately explore the housing price of second-hand homes, the authors of paper [12] studied and assessed 35417 pieces of data that Chengdu HOME LINK network had gathered. Cleaning and selecting the qualities from the gathered data came first. For each of these ten criteria, the predicted house price score curve was then fitted using multiple linear regression, decision tree, and XGboost models. The best prediction model was then selected by adjusting the model's parameters. The experimental results show that XGboost has the highest prediction accuracy, with a prediction accuracy score of 0.9251. In comparison to linear regression and decision tree models, the XGboost technique has higher generalization ability and robustness in data prediction, avoids overfitting problems, and offers a solid foundation for the future prediction of second-hand house price.

Hence, our solution to the mentioned challenges is a revolutionary machine learning technique that forecasts the precise pricing point for every product in the burger cafe business on a daily basis.

Chapter 3

MATERIALS AND METHODOLOGY

This chapter covers some methodologies that we use in this study. It includes Machine Learning Algorithms, Linear Regression, Tree-based method, Elasticity and Linear program as well as Metrics to evaluate models.

3.1 Machine Learning Algorithms

Machine learning is a branch of artificial intelligence (AI) that enables computers to "self-learn" from training data and improve over time without the need for explicit algorithms that are developed for them [14]. Machine learning algorithms are able to spot patterns in the data and learn from them, which enables them to make their own predictions. In a word, the models and algorithms used in machine learning get more experienced over time.

In typical programming, a computer engineer will develop a list of steps that a computer should carry out in order to transform the data that it receives into the result that the programmer requires. The majority of the instructions are structured in an IF-THEN format, which directs the computer to carry out a certain action only if a set of predetermined requirements have been satisfied.

In contrast, machine learning is an algorithmic process that lets computers learn from their own mistakes and make judgments based on their past performance.

Artificial intelligence and machine learning are two separate concepts, despite the fact that they are commonly used interchangeably with one another. Machine learning is a subset of artificial intelligence that enables autonomous learning from data, whereas artificial intelligence more generally refers to machines that can reason, learn new skills, and solve problems in ways that mimic those of humans.

We may choose to provide machine learning algorithms with examples of labeled data, which is also referred to as training data. This will enable the algorithms to automatically calculate, analyze the data, and recognize patterns in the data without human intervention.

Simply said, machine learning is a sophisticated labeling machine, according to Google's Chief Decision Scientist. Machines can be trained to identify items to show them be examples of fruit; after learning from appropriate and accurate training examples, the machines will eventually begin labeling apples and pears on their own.

On enormous volumes of data, machine learning can be put to work and is considerably more accurate than people. It may help us save money and time on tasks and analyses, for example, lowering customer annoyance in order to raise customer satisfaction, automating the process of submitting support tickets, and data mining from internal sources and all over the internet.

3.1.1 Supervised learning

Predictions are generated using labeled training data by supervised learning algorithms and models. Each training sample contains both inputs and intended outcomes. When choosing the labels for unseen data, a supervised learning algorithm evaluates this sample data and draws an inference, or, to put it another way, produces an informed guess.

The most typical and widely used approach to machine learning is this one. It is "supervised" because these models require manually labelled sample data to be fed into them in order to learn from. Data is labeled in order to instruct the computer what patterns (e.g., resemblances between words and images, data categories, etc.) to search for and identify relationships with.

To automatically identify spam, for instance, we would need to provide a machine learning system with instances of emails we want to be flagged as spam and emails that are important and shouldn't be.

Classification and regression are the two types of tasks that fall under the category of supervised learning. In which:

- **Classification in supervised machine learning**

Support Vector Machines (SVM) and Naive Bayes are two of the most widely used classification algorithms in supervised learning.

A category with a limited number of alternatives is the output value for classification tasks. As an illustration, we may automatically categorize data as good, negative, or neutral using this free pretrained sentiment analysis model.

- **Regression in supervised machine learning**

The predicted outcome in regression tasks is a continuous number. This model can give any numerical value within a specific range because it is used to predict quantities like the likelihood that an event will occur. Regression challenges also include things like trying to guess how much a house will sell for in a certain area or how far COVID-19 will spread in a certain region.

3.1.2 Unsupervised learning

Algorithms for unsupervised learning find patterns and correlations in data that has not been labeled. In this scenario, models are fed input data but, without prior knowledge of the projected outputs, must draw inferences using only circumstantial evidence. Because they weren't given the "correct answer," the models need to figure out how to spot patterns on their own.

One of the most common forms of unsupervised learning is called "clustering," and it entails grouping together pertinent data. This technique, typically used in exploratory analysis, helps uncover previously unseen tendencies or patterns.

The marketing department of an online store, for instance, may employ clustering to better categorize its customers. If you feed a machine learning algorithm a customer's income and expenditure information, it will discover groupings of consumers that exhibit similar habits.

Marketers can customize their strategies using segmentation for each important market. As a means to reward loyalty and increase retention, they might provide specials and discounts for low-income consumers who spend a lot of money on the website.

3.1.3 Machine Learning Workflow

Machine learning workflows specify the actions taken throughout a specific machine learning deployment. Its workflows differ per project, however five key processes are commonly included in figure 3.1:

Figure 3.1: Machine learning workflow

First, it is gathering machine learning data. Data gathering is a critical phase in machine learning operations. The quality of the data we obtain throughout data collection influences our project's potential usefulness and accuracy. To begin collecting data, we must first identify our sources and then aggregate the information they give into a single dataset. This might include gathering open-source information sets, streaming data from Internet of Things sensors, or creating a data lake out of various files, logs, and media.

Second, it is data pre-processing. We must pre-process our data after collecting it. The data must be cleaned, verified, and transformed into an useful dataset before preprocessing can begin. It's possible that this will be a very simple operation if the data we need comes from only one place. However, in order to successfully merge data from several sources, we need to ensure that the formats of the data can communicate with one another, that the data is equally reliable, and that any potential duplicates are

eliminated.

Third, it is building datasets. At this point, processed data is separated into three datasets for training, validation, and testing: The first training set is used to guide the algorithm in information processing. The validation set is used to determine the level to which the model is accurate, and this set contains parameters that reflect how the model is categorized. This dataset is utilized in the process of parameter tuning for the model. The performance of the models, as well as their precision, are evaluated using the test set. The purpose of this collection is to draw attention to any problems or deficiencies in the model's training.

Fourth, it involves training and refinement. After obtaining the appropriate datasets, we may train our model. We must input our training data to our algorithm in order for it to learn the appropriate parameters and properties for classification. After training, we may utilize our validation dataset to enhance the model. In order to do this, it is necessary to adjust model-specific settings (hyperparameters) until an acceptable degree of accuracy is attained. This may need the addition or deletion of variables.

Finally, it is machine learning evaluation. We can test our model after its accuracy has been tuned and an appropriate set of hyperparameters has been determined. Our test dataset is used in testing to ensure that our models are using accurate attributes. Depending on the input, we may go back and retrain the model to improve accuracy, change output parameters, or deploy the model as needed.

3.2 Linear Regression

Regression, at its most fundamental level, seeks to characterize and evaluate the connection between a given variable and one or more other variables. Specifically, regression is an attempt to provide an explanation for shifts in one variable by making use of shifts in another variable (or variables).

In the simplest terms, Linear Regression is the supervised machine-learning model that identifies the best-fitting linear line between the independent variable and the dependent variable.i.e, it determines whether the dependent and independent variables have a linear relationship.

Simple regression and multiple regression are the two most common types of linear regression. Simple linear regression is used when there is just one independent variable to assess how linearly it is related to the variable that is being analyzed (the dependent variable). Multiple linear regression, on the other hand, makes use of a number of different independent variables in order to determine associations. In this thesis, we will focus on simple linear regression and this section will be based on Introductory Econometrics for Finance (Chris Brooks) [16].

To be more specific, let's call the variable whose changes are being explained by the regression y, and the variables $x_1, x_2, ..., x_k$ that are being utilized to do so. To put it another way, we may say that in this very straightforward scenario, changes in y are a result of shifts in k variables (the xs).

3.2.1 Simple Linear Regression

Assume that a researcher has some inkling that y and x should be related, and financial theory implies that when x goes up, y does too. There seems to be a positive linear connection among x and y here, with higher values of x generally coinciding with higher values of y, such that a straight line may be used to approximatively describe the relationship between them. The resulting formula is as follows:

$$y = \alpha + \beta x. \tag{3.1}$$

The researcher would then try to determine the coefficients or parameters values, α and β, to get the line to lie as near to the data points as feasible.

But this model is obviously not realistic. In terms of statistics, it equates to the situation when the model accurately predicted the data, meaning that every single data point fell within a perfectly straight line. As a result, a *random disturbance term*, represented by the symbol μ, is added to the equation to increase the model's realism. Then:

$$y_t = \alpha + \beta x_t + u_t, \forall t = \overline{1, n}. \tag{3.2}$$

Here, choosing values for α and β helps reduce the (vertical) gaps between the data and the lines as close as possible. Thus, they are selected to minimize the total distances

(vertical) between the data points and fitted line. As demonstrated in Figure 3.2, one may generate a scatter plot for each pair of variables y and x by 'eyeballing' the data and manually drawing a line that looks to fit the data well.

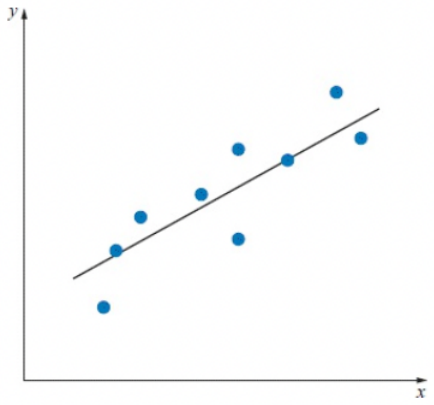

Figure 3.2: Simple linear regression

Keep in mind that *vertical distances* are frequently reduced, as opposed to horizontal or perpendicular lengths. Given (or conditioned on) the observed values of x, it is difficult to choose the right model for y since it is assumed that x remains the same across repeated samples.

If merely suggestive findings are needed, this 'eyeballing' process would be acceptable, but it is obvious that this technique will be inaccurate in addition to being time-consuming. Ordinary least squares is the most used technique for fitting a line to the data (OLS). This method, which serves as the backbone of econometric model estimate, will be covered in Chapters 3 and 4 [16].

Besides that, there are two more estimate techniques (The method of moments and the maximum likelihood) to establish the right values of the coefficients α and β. Despite being well known, Hansen's (1982) generalized version of the technique of moments is outside the purview of this work. Additionally popular, the maximum probability approach will be in-depth explained in Chapter 9 [16].

Suppose that there are five different observations in the data sample. As seen in Figure 3.3, the OLS requires computing each vertical distance that exists between a point and a line, then squaring that distance, and minimizing the overall sum of the squares' areas (it is 'least squares'). This is comparable to decreasing the overall area of the

squares that are produced while connecting the points to the line.

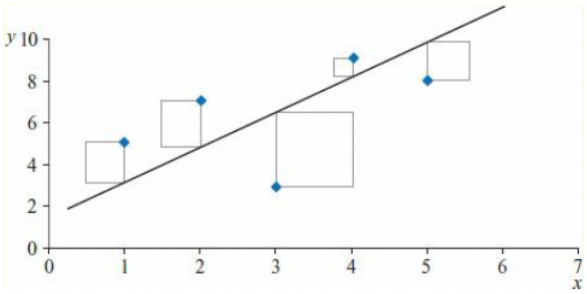

Figure 3.3: OLS fitting a line

Simply, let y_t represent the actual data point for observation t and $\hat{y}_t$ is the fitted value from the regression line, or, more precisely, let $\hat{y}_t$ represent the value for y that the model would have predicted for the given value of x in observation t. Then, let $y_t - \hat{y}_t$ represent the residual μ, which is the difference between the actual value of y and the value predicted by the model for this data point. This is represented for a single observation t in Figure 3.4.

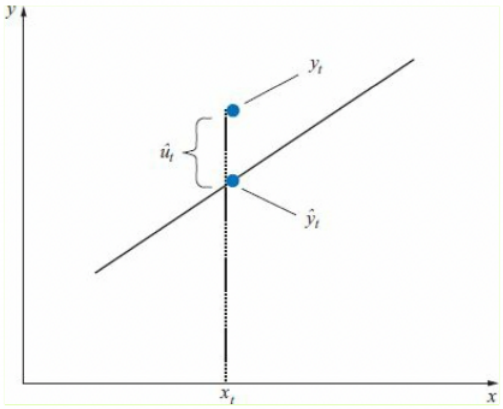

Figure 3.4: Single observation

Given that some points will be above the line and some below it, minimizing the sum of squared distances is preferred to computing the sum of $\hat{u}_t^2$ as near to 0 as feasible. Therefore, when the total is pushed as close to zero as possible, the values above the line are regarded positive, while the values below the line are considered negative. Since the lengths from points above and below the line often cancel each other out, it is

possible to fit the data to virtually any line. In this instance, the predicted coefficients would not have a single solution. The total of the $\hat{u}_t$ for each fitted line that crosses the data mean is really 0. (i.e., $\bar{x}, \bar{y}$). The squared distances, on the other hand, ensure that any deviations included in the computation are positive and do not cancel out.

Therefore, minimizing $\left(\sum_{t=1}^{5} \hat{\mu}_t^2\right)$ results in minimizing the total of squared distances. The *residual sum of squares* (RSS) or sum of squared residuals is the name given to this total. However, what is $\hat{u}_t$? Once more, the distinction between the actual point and the line, $y_t - \hat{y}_t$. Therefore, minimising $\sum_t \hat{\mu}_t^2$ is similar to minimising $\sum_t (y_t - \hat{y}_t^2)^2$.

The fitted line's equation is provided by $\hat{y}_t = \hat{\alpha} + \hat{\beta} x_t$, where $\hat{\alpha}$, $\hat{\beta}$ stand for the corresponding values of α, β chosen by minimizing the RSS. Let us now use the symbol L to denote the RSS, which is also known as a loss function. Take into consideration the overall number of observations T (from time t = 1 to time T).

$$L := \sum_{t=1}^{T} (y_t - \hat{y}_t)^2 = \sum_{t=1}^{T} (y_t - \hat{\alpha} - \hat{\beta} x_t)^2.$$

Making L's first order partial derivatives, w.r.t $\hat{\alpha}$ and $\hat{\beta}$, then we have:

$$\frac{\partial L}{\partial \hat{\alpha}} = -2 \sum_{t=1}^{T} (y_t - \hat{\alpha} - \hat{\beta} x_t) = 0,$$

$$\frac{\partial L}{\partial \hat{\beta}} = -2 \sum_{t=1}^{T} x_t (y_t - \hat{\alpha} - \hat{\beta} x_t) = 0.$$

So, OLS $\hat{\alpha}$ and $\hat{\beta}$ are as a result:

$$\hat{\beta} = \frac{\sum_{t=1}^{T} (x_t - \overline{x})(y_t - \overline{y})}{\sum_{t=1}^{T} (x_t - \overline{x})^2} \quad , \quad \hat{\alpha} = \overline{y} - \hat{\beta}\, \overline{x}, \tag{3.3}$$

and the sample means of the observations are shown by $\overline{x}, \overline{y}$.

Classical Linear Regression Model

We have

$$y_t = \alpha + \beta x_t + u_t, \forall t = \overline{1, n}. \tag{3.4}$$

, together with the following assumptions, is what statisticians call classical linear regression model (CLRM). Since both x_t and y_t depend on μ_t, it is essential to understand how μ_t are generated. A standard set of assumptions is established for the μ_t below, also known as the unobservable error or disturbance terms. Regarding their observable equivalents, the residuals of the estimated model, it is important to highlight that no assumptions are made. If assumption (1) is correct, assumption (4) may be expressed as $\mathrm{E}(x_t\ \mu_t) = 0$. The regressor is orthogonal to the error term, according to both formulations (i.e., unrelated to it). The marginally stronger alternative to assumption (4) is that the x_t are constant or non-stochastic in repeated samples. The value of x_t is determined independently of the model due to the absence of sampling variance in x_t.

- $E(u_t) = 0$.
- $Var(u_t) = \sigma^2 < \infty, \forall t = \overline{1, n}$.
- $\mathrm{Cov}(u_i, u_j) = 0, \forall 1 \leq i < j \leq n$.
- $\mathrm{Cov}(x_t, u_t) = 0, \forall t = \overline{1, n}$.
- $u_t \sim \mathcal{N}(0, \sigma^2), \forall t = \overline{1, n}$.

Making accurate conclusions about the population parameters (the real α and β) from the sample parameters ($\hat{\alpha}$ and $\hat{\beta}$) determined using a finite amount of data necessitates the fifth assumption, the disturbances have a normal distribution.

3.2.2 Multiple Linear Regression

As I mentioned above, we do not focus so much in Multiple linear regression. So, we just go through. Multiple linear regression makes use of a number of different independent variables in order to determine associations.

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + +\beta_n x_n + \epsilon, \tag{3.5}$$

where x_i is ith predictor and β_i quantifies association between variable and sponse.

3.3 Tree-based Model

Tree-based models are helpful for demand forecasting since they must not need the creation of a functional connection between input attributes and output labels. Trees can be utilized to appropriately convey the outcome to business users and are also more easier to interpret. There were several different tree-based model types used in this thesis, including Random Forest, Decision Tree and XGBoost.

An ensemble technique is a strategy that combines several straightforward 'building block' models to produce a single, potentially extremely potent model. Since they could provide poor predictions on their own, these basic building block models are frequently referred to as weak learners. Now we will discuss its ensemble approaches.

3.3.1 Regression Trees

There are two main step to build this model:

1. By designating $R_1, R_2, ..., R_N$ as independent and non-overlapping areas, we partition the predictor space, or the set of possible values for $X_1, X_2, .., X_n$, into N distinct and non-overlapping regions.

2. For every observation inside region R_n, we generate the exact same forecast. It's as simple as averaging the response values saw during the training observations that fell inside R_n to arrive at this forecast.

The regions can take on whatever form we want. On the other hand, in order to simplify things and make the final predictive model easier to understand, we decided to partition the predictor space into high-dimensional rectangles, which are also known as boxes. The objective is to locate the $R_1, ..., R_N$ boxes that provide the lowest RSS value given by:

$$\sum_{i=1}^{N} \sum_{i \in R_n} (y_i - \hat{y}_{R_n})^2, \tag{3.6}$$

where $\hat{y}_{R_n}$ is the average response for training observations within the n^{th} box.

The feature space might theoretically be divided into N boxes, but doing so would be computationally expensive. To do this, we employ a top-down, greedy technique

known as recursive binary splitting. The predictor space is partitioned in stages using the recursive binary splitting approach, with each stage represented by two new branches further down the tree. This means that the method is top-down in nature, starting at the root and working its way to the leaves (where all observations belong to a single area). When constructing a tree, greediness manifests itself when the best split is selected at each stage rather than a split that will produce a better tree later.

First, we select the predictor X_n and the cutpoints so that the RSS is minimized when the predictor space is divided into the subspaces $\{X|X_n < s\}$ and $\{X|X_n \geq s\}$. If X_n has a value smaller than s, then that portion of predictor space is denoted by the notation $\{X|X_n < s\}$. In other words, we look at every conceivable combination of predictors $(X_1, ..., X_p)$ and cutpoint values (for each predictor), and pick the one that yields the lowest relative standard deviation (RSD). More specifically, we give a definition for the pair of half-planes for any n and s.

$$\sum_{i=1}^{N} \sum_{i \in R_n} (y_i - \hat{y}_{R_n})^2, \tag{3.7}$$

where

$$R_1(n, s) = \{X|X_n < s\} \quad \text{and} \quad R_2(n, s) = \{X|X_n \geq s\}. \tag{3.8}$$

And we look for s, n values that minimize equation.

$$\sum_{i:x_i \in R_1(n,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(n,s)} (y_i - \hat{y}_{R_2})^2, \tag{3.9}$$

where $\hat{y}_{R_1}$ and $\hat{y}_{R_2}$ are the means of the responses to the training observations in $R_1(n, s)$ and $R_2(n, s)$, respectively.

When the number of features p is low, finding the values of n and s that minimize (3.9) binary split is easy. Each region's RSS is then minimized by finding the optimal predictor and cutpoint to further divide the data. This time, rather than dividing all of predictor space in half, we just choose to divide one of the two established subspaces. Currently, there are three distinct areas. Again, we want to lower the RSS by further subdividing one of these three areas. This is done until some predetermined condition is met; for instance, until no region has more than five observations.

After $R_1, ..., R_N$ regions have been defined, the predicted response for a test observation is calculated by averaging the responses of training observations in the region to which the test observation corresponds.

3.3.2 Bagging

The bootstrap is an extremely potent concept in Chapter 5 [15]. It is utilized most frequently in situations in which a direct estimation of the standard deviation of an important variable is either difficult or impossible. It seems clear to us that the bootstrap might be utilized in this scenario to enhance existing statistical learning strategies like decision trees.

A issue arises from the enormous variety of the decision trees (Section 8.1 [15]). This indicates that if we randomly split the training data in half and then perform a decision tree fitting process to each half, we may get drastically different results. If, on the other hand, the ratio of the number of observations to the sample size is relatively high, linear regression has a propensity to have a low variance. A technique with a low variance will, when used repeatedly to various data sets, produce findings that are consistent with one another. In the context of decision trees, bootstrap aggregation, which is also known as bagging, is a very helpful technique that is frequently used. It is a general strategy for lowering the variation in the performance of statistical learning systems.

Keep in mind that the variance of the mean Z of the observations may be calculated using the formula $\frac{\sigma^2}{n}$. This formula is used when there are n independent observations $Z_1, ..., Z_n$, each of which has a variance of σ^2. As a consequence of this, reducing the variation of a set of data via averaging it out. Therefore, it makes sense to choose many training sets from the population, develop a new prediction model for each training set, and then average the resultant predictions in order to minimize variance and enhance test set accuracy using a statistical learning approach. To put it another way, we may compute functions such as $f^1(x), f^2(x), ..., f^B(x)$, and then take the average of all of these functions to produce a single statistical learning model with low variance. This model is denoted by:

$$\hat{f}_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^b(x).$$

Of course, the lack of availability of many training sets makes this impractical. Instead, by using additional samples from the (single) training data set, we may bootstrap. With this method, we produce B various bootstrapped training data sets. Then, using the b^{th} bootstrapped training, we train our algorithm to have $f^{*b}(x)$, and then take an

average of all the forecasts, to get

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x).$$

This is method of bagging.

Bagging can enhance predictions for various regression techniques, but decision trees benefit the most from its use. We simply construct B regression trees from B bootstrapped training sets, average the resulting predictions, and then bag regression trees. These trees are not maintained and have a deep root system. Consequently, each tree has a high amount of variation yet a low level of bias. These B trees are averaged to lessen volatility. By combining a large number of trees into a single process (bagging), accuracy can be dramatically improved.

To predict a quantitative result Y, we merely provided the bagging approach inside the framework of regression. In order to foretell the variable's future value, we did this. If Y is a qualitative variable, what effect does bagging have on a classification problem? There are a number of possible strategies to employ in this situation, but the simplest is as follows. In addition to noting which class was predicted by each B tree for a particular test observation, we may additionally note which class obtained the most predictions from the B trees as a whole. Here, we display the percentage of incorrect test results as a function of B, where B is the sample size of the training dataset used to train the trees. As a result of these factors, we can observe that the bagging test yields a somewhat lower error rate than the single-tree test. Because B, the number of trees, is optional in bagging, overfitting is not an issue even if B is set to very large values. To put this into reality, we pick a B value so astronomically large that the error is effectively eliminated.

3.3.3 Random Forest

Through a little modification that decorrelates the trees, random forests offer an improvement over bagged trees. Bootstrapped training samples are used to create multiple decision trees, which is similar to bagging. During the process of creating these decision trees, a random subset of m predictors from the whole set of p predictors are selected as split candidates for each potential split.

Random Forest Algorithms:

1. Samples of n training instances from X and Y with replacement: X_b and Y_b.

2. Build a regression tree f_b using the variables X_b, Y_b. When constructing decision trees, a random sample of m predictors is selected as split candidates from the entire collection of p predictors each time a split in a tree is taken into account.

3. A forecast for a sample x' after training may be generated by averaging predictions from all the different regression trees on x':

$$\hat{f} = \frac{1}{B} \sum_{b=1}^{B} f_b(x').$$

One of the m predictors can be used by the split, but not by any other operation. In many cases, the number of predictors examined at each split is close to being equal to the square root of the total number of predictors, or $m \simeq \sqrt{p}$.

To put it another way, when creating a random forest, the algorithm is not even permitted to take into account the majority of the predictors that are accessible at each branch in the tree. Although it may seem absurd, there is a good reason behind this. Assume that the data set contains a few predictors that are moderately strong and a few really strong predictors. The majority or all of the trees in the collection of bagged trees will then employ this reliable prediction in the top split. As a result, the bagged trees will all have a similar appearance. So, there will be a strong correlation between the predictions from the bagged trees. Unfortunately, there is not as much variance reduction from averaging several highly linked data as there is from averaging numerous uncorrelated numbers. This specifically indicates that in this case, bagging will not result in a significant decrease in variance over a single tree.

This issue is solved by random forests, which require that each split take into account just a portion of the predictors. Thus, on average $\frac{p-m}{p}$ of the splits will not even take the strong predictor into account, giving other predictors a better chance. By decorrelating the trees, we may reduce their variability and increase the reliability of the average of the resultant trees.

3.3.4 Boosting

Now we'll discuss boosting, another strategy for improving a decision tree's forecasts. Much like bagging, boosting is a general approach that may be used with other statistical learning methods for regression or classification. The use of boosting to decision trees is the sole focus of this essay.

To recap, in order to create a single prediction model from a bag of predictions, we must first create several copies of the original training data set using the bootstrap, each of which will be fitted with a distinct decision tree. This may be done by combining the trees. Notably, a bootstrap data set is used to create each tree in isolation from the others before they are combined. Boosting is a method of growing trees that works in a similar way to other methods in that it employs the knowledge gained from previously grown trees to assist in the growth of subsequent plants. Boosting is an alternative to bootstrap sampling that bases the fitting of each tree on an altered version of the starting data set.

First, think about the environment for the regression. Boosting is quite similar to bagging in that it also necessitates the union of several different decision trees ($\hat{f}^1, ..., \hat{f}^B$). So, what is the purpose of this process? Instead of trying to force-fit a single huge decision tree to the data, which leads to hard fitting and likely overfitting, the boosting technique takes its time to learn. We fitted a decision tree to the model's residuals using the present model. Instead of using the result Y as the response, we fit a tree using the existing residuals. The residuals are then modified by include this new decision tree within the fitted function. The size of each of these trees is defined by the algorithmic parameter d, which may only have a few terminal nodes in each tree. In locations where $\hat{f}$ does not perform well, we gradually improve it by fitting tiny trees to the residuals. The process is slowed down even more by the shrinkage parameter λ, enabling more trees of various shapes to make an assault on the residuals. Statistical learning approaches that take their time to improve their performance typically have excellent results. Be aware that each tree's construction in boosting depends heavily on the trees that have already been produced, unlike in bagging.

What we just did was a walkthrough of the procedure for boosting regression trees. As for how to improve classification trees, it's done in a way that's similar to the last but slightly more involved; the details are left out here for clarity.

Algorithm : Boosting for Regression Trees

1. $\hat{f}(x) = 0$, $r_i = y_i$ for every iteration in the training dataset.

2. For b = 1, 2,...,B, repeat:
   (a) Fit to the training data (X,r) a tree $\hat{f^B}$ with d splits (d + 1 terminal nodes).
   (b) Update $\hat{f}$ by including a reduced version of the new tree:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x). \tag{3.10}$$

   (c) Update residual values,

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i). \tag{3.11}$$

3. Out the boosted model,

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x). \tag{3.12}$$

Boosting has three tuning parameters:

1. The quantity of trees B. Boosting can overfit if B is too large, unlike bagging and random forests, however this overfitting usually happens gradually, if at all. Cross-validation is used to choose B.
2. The shrinkage parameter, which is a tiny positive value called λ. This regulates how quickly boosting learns. The appropriate decision can vary depending on the task and typical values of 0.01 or 0.001. For very small λ, using a very high value of B may be necessary for good performance.
3. The dth split in each tree, which determines how complicated the boosted ensemble is. When d = 1, each tree is a stump made out of a single split, and this solution is frequently successful. Since each term only contains one variable, the boosted ensemble is here fitting an additive model. Since d splits can only involve a maximum of d variables, d is the interaction depth, and determines the sequence in which the boosted model's interactions occur.

3.3.5 XGBoost

Extreme Gradient Boosting (XGBoost) is a scalable, distributed gradient-boosted decision tree (GBDT) machine learning framework.

A technique known as Gradient Boosting Decision Trees (GBDT) is an ensemble learning method for decision trees that functions similarly to random forest and may be used for classification and regression. Ensemble learning algorithms combine several distinct types of machine learning strategies in order to produce more accurate models.

The practice of 'boosting' or strengthening one weak model by merging it with a number of other weak models in order to obtain a model that is more dependable as a whole is referred to as gradient boosting. The term 'gradient boosting' refers to an extension of 'boosting' that formalizes the process of additively generating weak models as a gradient descent strategy over an objective function. Gradient boosting is a kind of 'boosting'. By determining the intended results for the next model, gradient boosting helps to reduce the number of errors that are made. Because the expected results in each scenario are defined by the gradient of the mistake in relation to the forecast, the technique is referred to as 'gradient boosting'.

The gradient boosting technique known as XGBoost pushes the limits of what can be accomplished in terms of processing power for boosted tree algorithms. It is also highly precise despite being scalable. It was designed in the first place to enhance the effectiveness of machine learning models and their overall performance. In contrast to the sequential method used by GBDT, XGBoost builds its trees using a parallel technique. It does this by employing a level-wise method, during which it scans through gradient values and assesses the quality of splits at each potential split in the training set by utilizing these partial sums.

We have to find a predicted value/gamma for which the loss function is minimum

$$F_0(x) = argmin \sum_{i=1}^{n} L(y_i, \gamma),$$

L : loss function,

γ : predicted value.

Since the target column is continuous our loss function will be :

$$L = \frac{1}{n} \times \sum_{i=0}^{n} (y_i - \gamma_i)^2,$$

y_i : observed value,

γ : predicted value.

Taking the derivative of loss function w.r.t the predicted value :

$$\frac{dL}{d\gamma} = -\sum_{i=0}^{n}(y_i - \gamma_i),$$

$$\gamma_m = argmin \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)),$$

$$F_m(x) = F_{m-1}(x) + \nu_m h_m(x).$$

3.4 Metrics

Clearly, a forecasting model cannot be evaluated only on a single forecast or reality. As a result, in practice, it is common to make predictions for the whole out-of-sample time, compare these forecasts with actual values, and then aggregate the difference between the two sets of values. Forecast error refers to the difference between observation i's actual value and the prediction made for it. The forecast error would be positive (negative) if the forecast was too low(high). Positive and negative mistakes cancel each other out, rendering it difficult to simply add up forecast errors. As a result, before adding up the forecast mistakes, they are frequently squared or given an absolute value, making them all positive.

Since MSE offers a quadratic loss function, it may be especially helpful when smaller forecast error rates are disproportionately higher than larger error rates. Despite the fact that the same critique might be leveled against the whole least squares approach, this may be viewed as a downside if large mistakes are not disproportionately more severe. In fact, Dielman (1986) [17] suggests that when there are outliers, the model's parameters should be preferred using least absolute values better to least squares. According to Makridakis (1993, p.528) [18], the mean absolute percentage error (MAPE) is a relative statistic that 'incorporates the finer characteristics of the various accuracy standards'. The MSE may also be defined as the difference between s-step-ahead predictions of a variable made at time t (f_t) and the variable's actual value at time t (y_t).

$$MSE = \frac{1}{T - (T_1 - 1)} \sum_{t=T_1}^{T} (y_{t+s} - f_{t,s})^2,$$

where T_1 is the first out-of-sample forecast observation and T is the entire sample size (in-sample + out-of-sample). As a result, the initial range of observations for in-sample

model estimation is observation 1 to (T_1-1), whereas observations T_1 to T are available for out-of-sample estimation, yielding a total holdout sample of $T-(T_1-1)$.
MAE measures the average absolute forecast error, and is given by

$$MAE = \frac{1}{T-(T_1-1)} \sum_{t=T_1}^{T} |y_{t+s} - f_{t,s}| \,.$$

And RMSD (Root-mean-square deviation) or RMSE (Root-mean-square error) is calculated by: $\sqrt{MSE}$.

3.5 Price Elasticity of Demand

Using the idea of price elasticity of demand, we can determine how much demand there is for a product at various price points [19].

Then, its formula is:

$$Ed = \frac{\Delta Q}{Q} . \frac{P}{\Delta P},$$

where Q and P stand for the product's initial demand and price, respectively. ΔQ and ΔP show how demand changes as prices fluctuate and how the cost of the product changes.

A product is highly elastic if demand changes significantly for a minor change in price, unitary elastic if demand changes significantly in direct proportion to price changes, and inelastic if demand changes significantly but prices do not move much.

The task of determining price elasticity of demand is difficult since we operate in a F&B world. Since a large variety of products are sold, we encounter various elasticity. Normally, the link between price and demand is inverse, meaning that when the price of a product lowers, demand for it rises and vice versa.

However, in the case of Giffen/Veblen commodities, this relationship is not upheld. An example of a product type whose demand rises in response to price increases is a Giffen/Veblen good. In addition to diverse elasticity kinds, we also encounter Giffen/Veblen commodities in our surroundings. For instance, a Channel bag is an example of a Veblen good since its demand rises as its price rises due to its increased standing as a status symbol.

Using price elasticity, we can predict how much interest there will be in a given product at different prices. From the perspective of the manager, the default discount can shift by no more than a user-specified amount (in this case, $\delta\%$). Keeping δ at a small positive integer is preferable since a large shift in product discounts would be counterproductive.

Demand values for all products tomorrow at the initial discount rate are calculated using the stated demand forecasting methodology. The price elasticity is also used to increase the basic discount by $\delta\%$ and decrease it by $\delta\%$ to obtain two additional demand values. So, using the concept of elasticity yields three distinct price points and corresponding values for each product. The price elasticity value is revised daily due to its changing nature.

3.6 Linear Program

With the help of the price elasticity of demand, we are able to determine the three pricing points for a product as well as the demand for it at each price point.

In order to maximize net revenue, we must now select one of these three pricing. There will be a total of 3^N permutations with N goods and 3 pricing points. Consequently, the issue boils down to an optimization problem, and we phrase it as follows: [20].

$$max \sum_{k=1}^{3} \sum_{i=1}^{n} p_{ik} x_{ik} d_{ik},$$

$$w.r.t$$

$$\sum_{k=1}^{3} x_{ik} = 1 \quad \forall i = \overline{1, n},$$

$$\sum_{k=1}^{3} \sum_{i=1}^{n} p_{ik} x_{ik} = c,$$

$$x_{ik} \epsilon \{0, 1\}.$$

In the equations above, p_{ik} represents the price and d_{ik} represents the demand for the i^{th} product if the k^{th} price-demand combination is selected. In this scenario, x_{ik} serves as a mask to choose a certain price for a product. x_{ik} is a boolean variable with the

values 0 or 1, where 1 indicates that the price has been selected and 0 indicates that it has not.
The restriction $\sum_{k=1}^{3} x_{ik} = 1$ ensures that just one k=1 price is selected for a product. And when c is a user-defined variable, $\sum_{k=1}^{3} \sum_{i=1}^{n} p_{ik} x_{ik} = c$ is calculated, guarantees that the sum of all specified prices equals c. The value of c can range between the lowest value obtained by picking the cheapest price for each product and the highest value obtained by selecting the most expensive price for each product. We reach the optimal outcome by locating a medium ground.

Given that price-demand combinations may be combined in many ways, the above formulation is computationally intractable. So, by considering x_{ik} as a continuous variable, we may transform the integer programming issue into a linear programming problem. Linear relaxation is the name of this approach. The answer to this problem can be thought of as a linear limit for the integer formulation problem. Here, the price that corresponds to the highest value of x_{ik} for the ith product is selected as the optimal price. Then:

$$\sum_{k=1}^{3} x_{ik} = 1 \quad \forall i = \overline{1, n},$$

$$\sum_{k=1}^{3} \sum_{i=1}^{n} p_{ik} x_{ik} = c,$$

$$0 \leq x_{ik} \leq 1 \ ,$$

In order to solve the aforementioned linear programming problem, we use the linprog procedure from the Scipy linear programming package. Following is the vector representation of the input:

$$max \quad r.x,$$

$$s.t \quad A.x = b.$$

Where r is a vector containing the revenue contribution of a product at each price point, A is a matrix containing product and pricing information.

Chapter 4

PROCESS AND ANALYSIS

To build a good model, we need to first analyze and process data, then remove unnecessary elements to evaluate the model most accurately. So, in this chapter, we will analyze and process data

4.1 Describe Data

In this thesis, we use transaction data (within 3 years) of a Burger Café restaurant in a Microsoft building in China, to show how to obtain price elasticity when adding product combo information, and outside information, including weather and vacations into the pricing model.

Input: The price and daily demand of the burger; Information about product combos; External features.
Model: Linear Regression, Random Forest, Decision Tree, XGBoost, Price Elasticity, Linear Programming.
Output: Optimal price for each product.

We have three files data csv, include: Cafe-Transaction-Store, Cafe-Sell-Meta-Data, Cafe-DateInfo. Now we will desribe each dataset:

1. Cafe-Transaction-Store: The dataset contains statistics on the selling price and quantity of burger items in the store by day. It includes the following fields:

	Attribute	**Dtype**	**Description**
0	CALENDAR_DATE	datetime64[ns]	date
1	PRICE	float64	price of product
2	QUANTITY	int64	quantity of product sold
3	SELL_ID	int64	id of product sold
4	SELL_CATEGORY	int64	category of the product

Table 4.1: Attributes of Cafe-Transaction-Store file

2. Cafe-Sell-Meta-Data: This dataset describes information about burgers and burger combos in the store.

	Attribute	**Dtype**	**Description**
0	SELL_ID	int64	id of product sold
1	SELL_CATEGORY	int64	category of the product
2	ITEM_ID	int64	item is contained in product
3	ITEM_NAME	object	name of the item

Table 4.2: Attributes of Cafe-Sell-Meta-Data file

3. Date-Infodata: This set contains calendar information and other data such as holidays, weekends,... In details:

	Attribute	**Dtype**	**Description**
0	CALENDAR_DATE	datetime64[ns]	date
1	YEAR	date	year
2	HOLIDAY	int64	holiday
3	IS_WEEKEND	int64	weekend
4	IS_SCHOOLBREAK	int64	school break
5	AVERAGE_TEMPERATURE	float64	temperature
6	IS_OUTDOOR	int64	outdoor

Table 4.3: Attributes of Date-Infodata file

4.2 Prepocessing

We have 3 individual data files. First, we'll upload and briefly rename them: transactions (Cafe-Transaction-Store), sold (Cafe-Sell-Meta-Data), date_info (Cafe-DateInfo). Then we will combine these three files into one (merge_data) via SELL_ID and CALENDER_DATE. We merge files like this to get information about the data that is more cohesive for later analysis. The combination of these 3 files can be represented by the following figure:

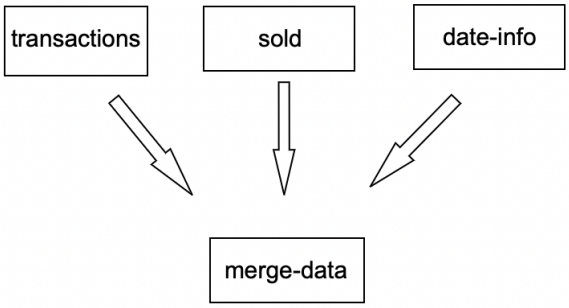

Figure 4.1: Combining data sets

Before combining the datas, we need to go through the following steps:

First, the transactions csv file contains the variable SELL_ID as a reference, so we need to have all the data about the SELL_ID in the additional sold set. Therefore, the data set must be converted from long format to wide format. This is done by

binary normalizing the composition information of the item codes via SELL_ID. We then combine the two data sets sold and transactions by column SELL_ID and name the new dataset sold_trans_data.

Next, we'll look at the date_info set. In this file, the HOLIDAY column has the value NaN, so we will add the value 'No Holiday' to it. Then we will separate the day, month, and year in the CALENDER_DATE column. Finally, we merge the 3 individual files together and name the new file merge_data.

Data after being merged has the same value as the tables below. We will quote the first 7 rows of data for reference:

	SELL_ID	**BURGER**	**COFFEE**	**COKE**	**LEMONADE**
1	1070	1	0	0	0
2	2051	1	0	1	0
3	2052	1	0	0	1
4	2053	1	1	1	0
5	1070	1	0	0	0
6	2051	1	0	1	0
7	2052	1	0	0	1

Table 4.4: Merge_data table

	CALENDAR_DATE	PRICE	QUANTITY	SELL_CATEGORY	YEAR
1	2012-01-01	15.50	46	0	2012
2	2012-01-01	12.73	22	2	2012
3	2012-01-01	12.75	18	2	2012
4	2012-01-01	12.60	30	2	2012
5	2012-01-02	15.50	70	0	2012
6	2012-01-02	12.73	22	2	2012
7	2012-01-02	12.75	16	2	2012

Table 4.5: Merge_data table

	MONTH	DAY	DAYOFWEEK	HOLIDAY	IS_WEEKEND
1	1	1	Sunday	New Year	1
2	1	1	Sunday	New Year	1
3	1	1	Sunday	New Year	1
4	1	1	Sunday	New Year	1
5	1	2	Monday	New Year	0
6	1	2	Monday	New Year	0
7	1	2	Monday	New Year	0

Table 4.6: Merge_data table

	IS_SCHOOLBREAK	AVERAGE_TEMPERATURE	IS_OUTDOOR
1	0	24.8	0
2	0	24.8	0
3	0	24.8	0
4	0	24.8	0
5	0	24.8	0
6	0	24.8	0
7	0	24.8	0

Table 4.7: Merge_data table

4.3 Exploratory Data Analysis

In this section, we will explore univariate and the relationship between them. Then, we will filter out unnecessary variables (outliers) to remove and create new datasets.

4.3.1 Univariate Analysis

1. SELL_ID: There are 4 main types of IDs which are 1070, 2051, 2052, 2053. In detail:

SELL_ID = 1070 : BURGER
SELL_ID = 2051 : BURGER, COKE
SELL_ID = 2052 : BURGER, LEMONADE
SELL_ID = 2053 : BURGER, COFFEE, COKE

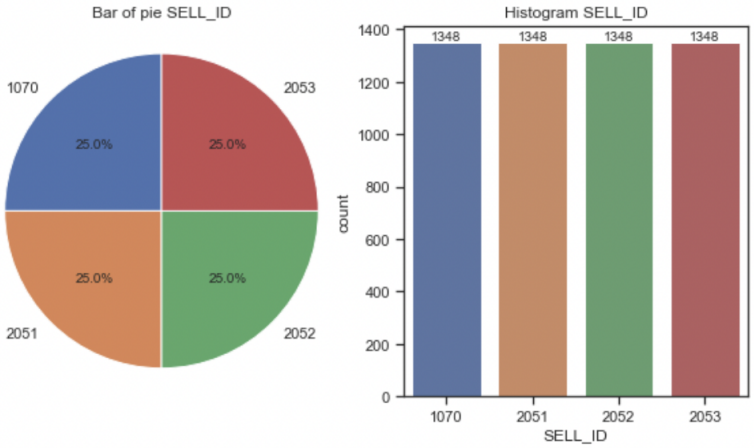

Figure 4.2: SELL_ID data

As seen in the figure above, the variable in the data set captures comprehensive sales data for all 4 product kinds per day. Thus, the date information will be duplicated in the aggregated data set. For the convenience of observing the data behind, we divide the set into 4 subsets according to each SELL_ID, and name each data as burger_1070, burger_2051, burger_2052, burger_2052, burger _2053.

```
burger_1070 = merge_data[merge_data['SELL_ID']==1070]
burger_2051 = merge_data[merge_data['SELL_ID']==2051]
burger_2052 = merge_data[merge_data['SELL_ID']==2052]
burger_2053 = merge_data[merge_data['SELL_ID']==2053]
```

2. HOLIDAY: There are 1348 days in total and 1243 days as No holiday. In which:

New Year 2015 is only 1 day off.
Mid-Autumn Day 2013 has only 1 day because it coincides with National Day.
WWII Celebration started in 2015 due to new regulations of China.

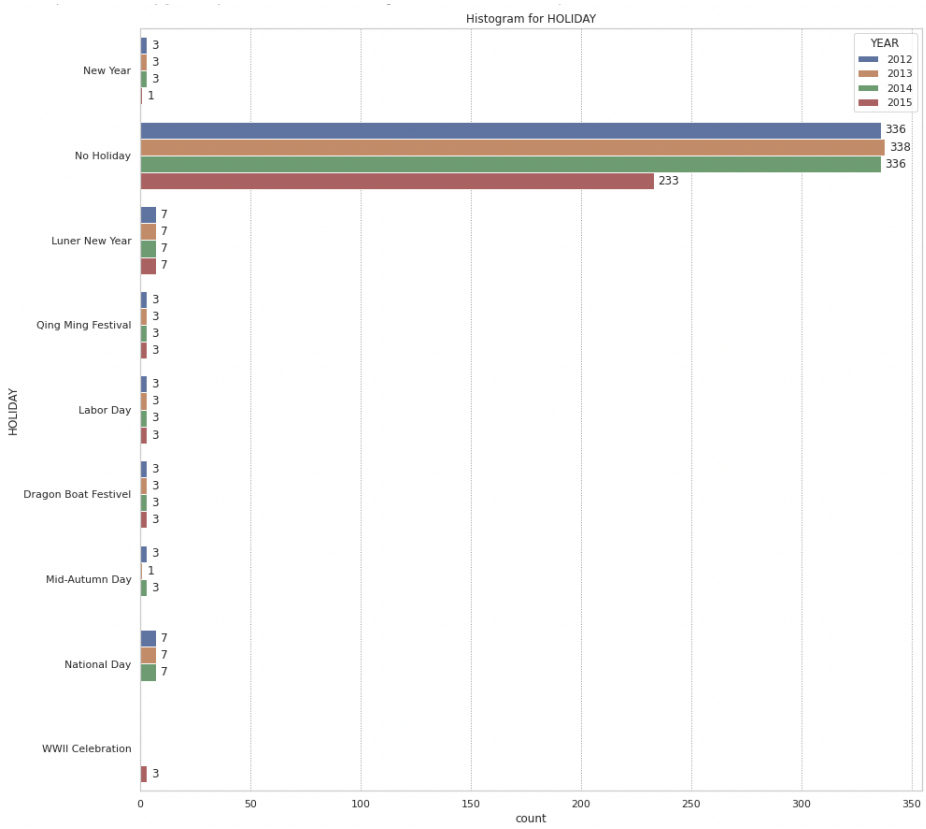

Figure 4.3: Day counting and classification chart

3. IS_WEEKEND: The variable IS_WEEKEND indicates that the weekend is Saturday, Sunday. Weekends make up 40% of weekdays.

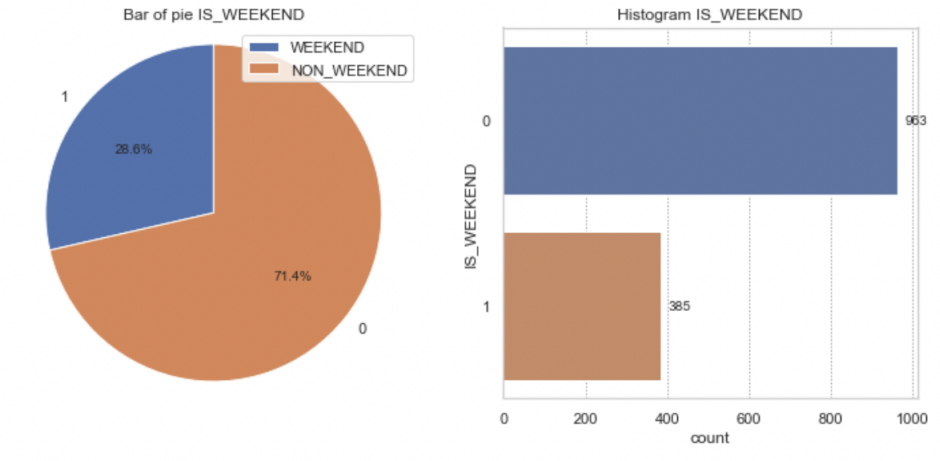

Figure 4.4: IS_WEEKEND chart

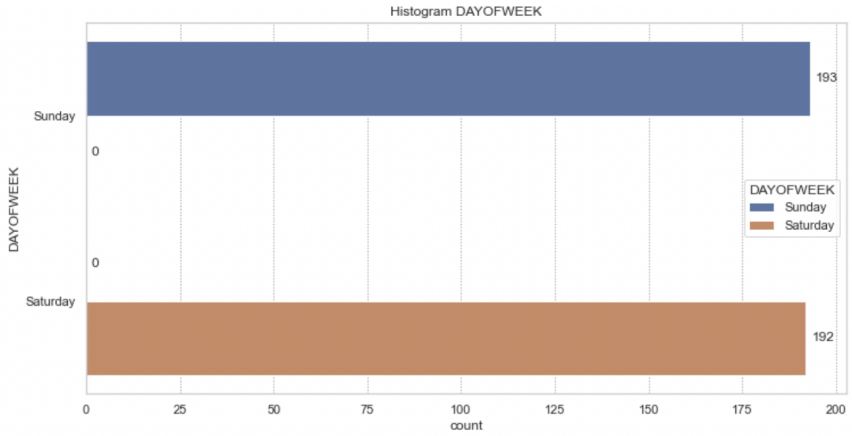

Figure 4.5: IS_WEEKEND chart

3. IS_SCHOOLBREAK: Students are absent from school in January, February, June, August, August. It is observed that school holidays are on the lunar new year in January and February, and summer holidays and other holidays in June, July, and August. Moreover, absence from school accounts for 27.1% of the total number of days considered. It is showed in figures 4.6, 4.7 below:

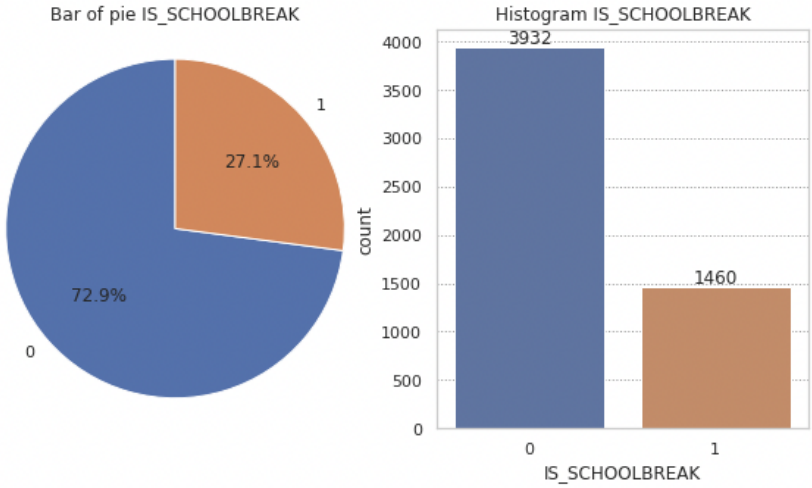

Figure 4.6: IS_SCHOOLBREAK chart

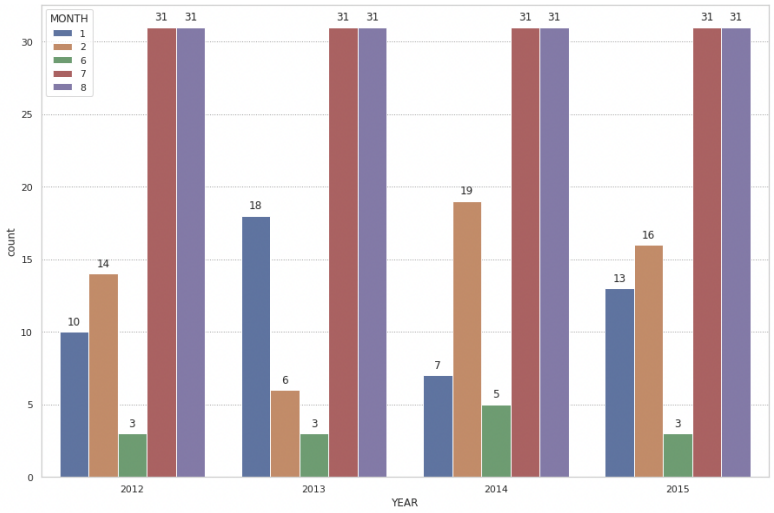

Figure 4.7: IS_SCHOOLBREAK chart

4. IS_OUTDOOR: Whether customers buy more or less will also depend on whether they go out or not. Here, customers going out accounted for 86.2%. From figure 4.9, we can see that most days at home are below 32 degrees F.

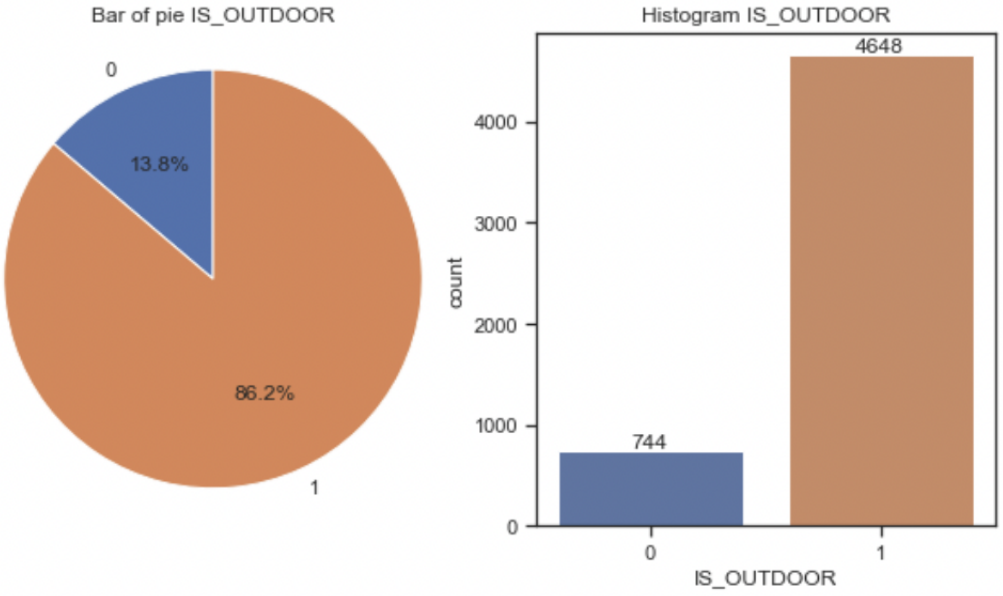

Figure 4.8: IS_OUTDOOR chart

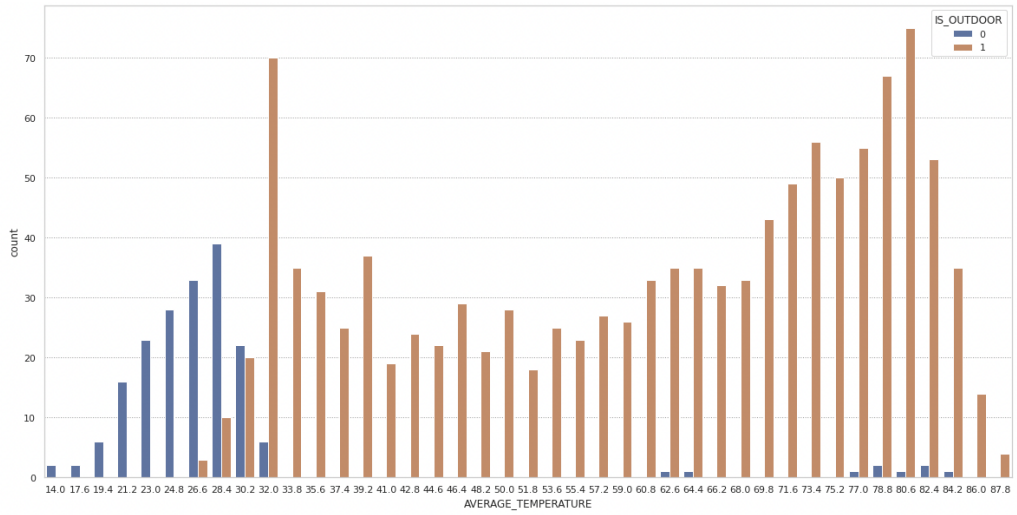

Figure 4.9: AVERAGE_TEMPERATURE chart

5. AVERAGE_TEMPERATURE: Average temperature is a key factor in determining whether or not a customer will go out. If the temperature is too low they will not go out (less than 32 degrees F), at 32 degrees F customers will go out more.

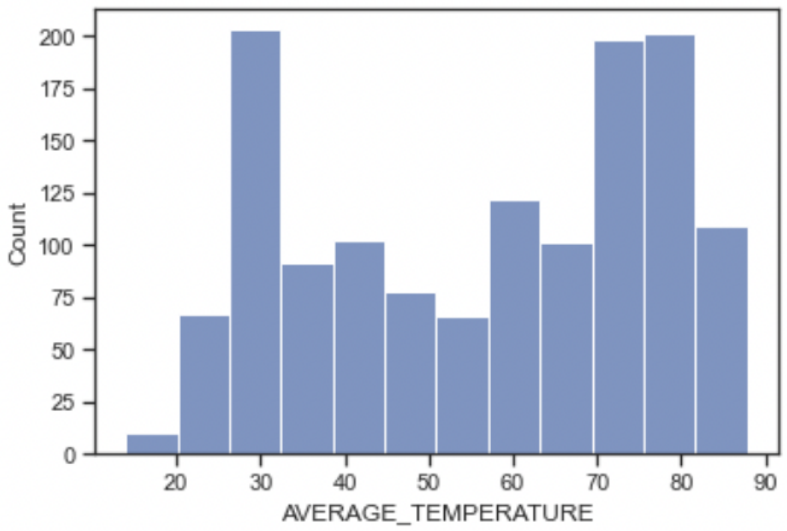

Figure 4.10: AVERAGE_TEMPERATURE chart

6. QUANTITY: Looking at the chart, we can see that the selling volume ranges from 50-100 servings/day. Customers give priority to buying burgers individually (SELL_ID = 1070) rather than buying combos (SELL_ID = 2051, 2052, 2053). Most purchases of combo 2052 (including BURGER, LEMONADE) are not prioritized by customers.

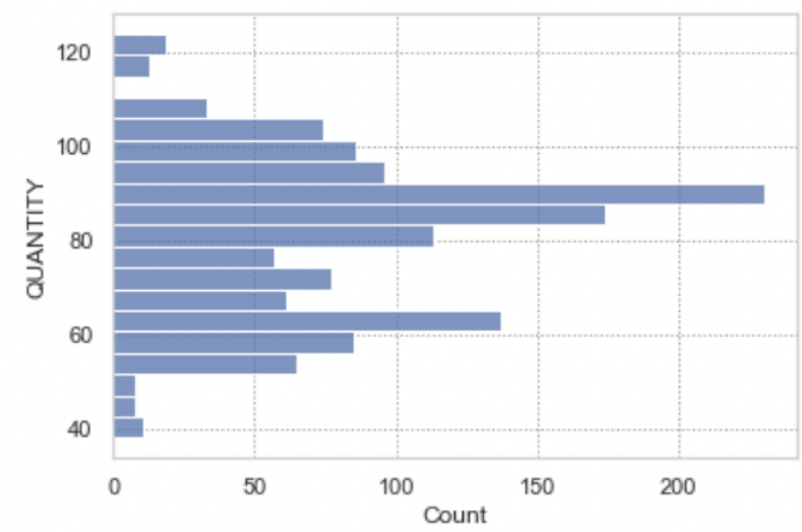

Figure 4.11: burger_1070 chart

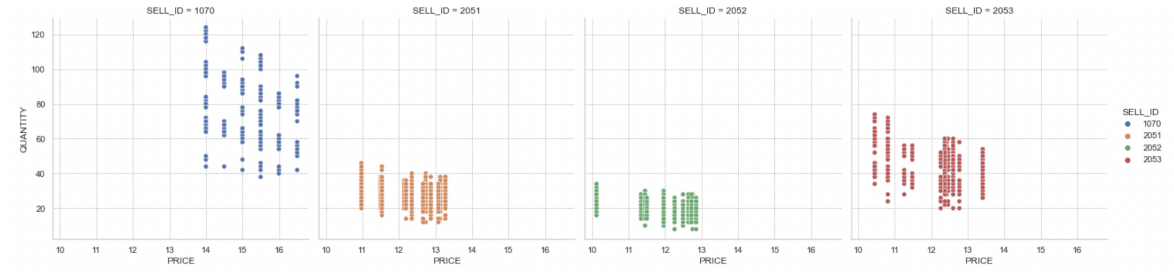

Figure 4.12: QUANTITY chart

7. PRICE: The retail price (SELL_ID = 1070) is always slightly higher than the combo sale, but this still does not affect the quantity, customers still buy the burger retail the most (figure 4.14).

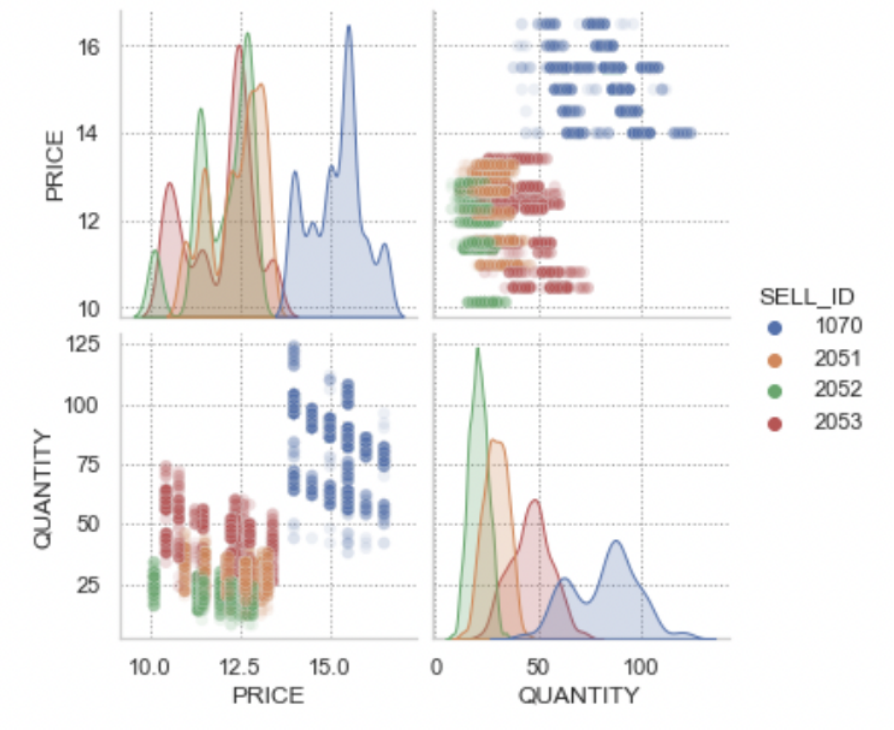

Figure 4.13: PRICE chart

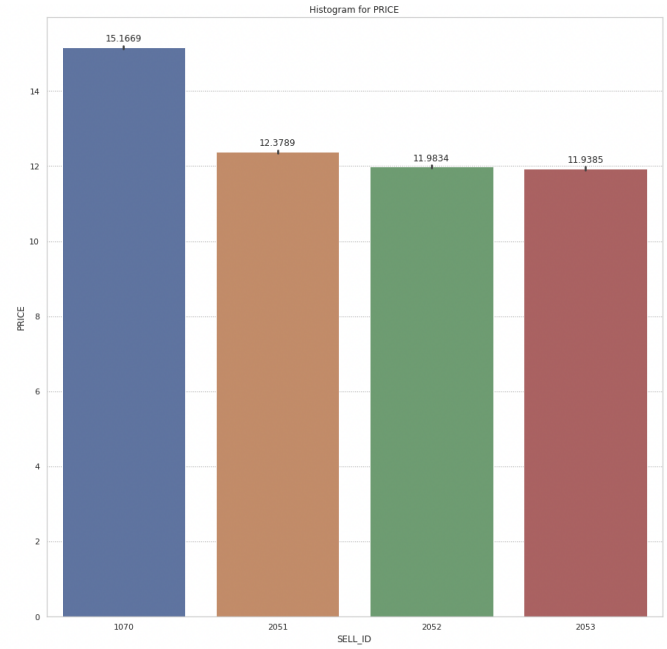

Figure 4.14: PRICE chart

4.3.2 Bi-variate Analysis

Next, we will examine the relationship of more than two variables to see how they affect each other.

Figure 4.15: Correlation graph

In figure 4.15, we see that our target variable QUANTITY is heavily influenced by the variable SELL_ID. So we will conduct analysis on each different SELL_ID. We divide the dataset into 4 parts based on SELL_ID (1070, 2051, 2052, 2053).

1. SELL_ID & QUANTITY

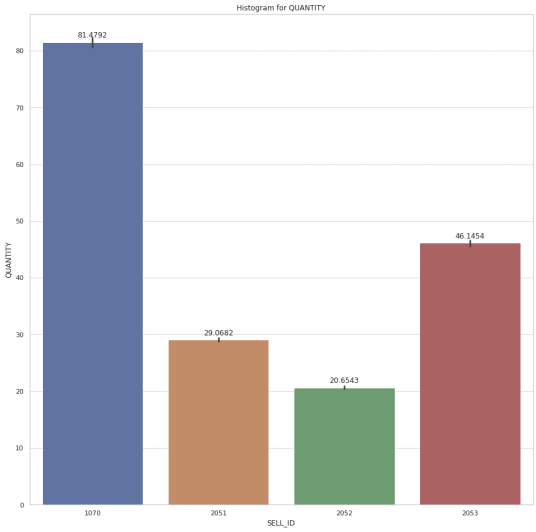

Figure 4.16: Quantity for each SELL_ID

From figure 4.16, SELL_ID = 1070 has the biggest quantities. Then, we will show correlation between variables of 1070 by figure 4.17 below:

Figure 4.17: Correlation graph for SELL_ID = 1070

Looking at the correlation chart 4.17, we see that our target variable QUANTITY

is correlated with PRICE and is also affected by the variables IS_WEEKEND, AVERAGE_TEMPERATURE, IS_OUTDOOR. This will produce outliers to our linear model. So, here we will see how it correlates and deal with the next steps.

2. PRICE & QUANTITY

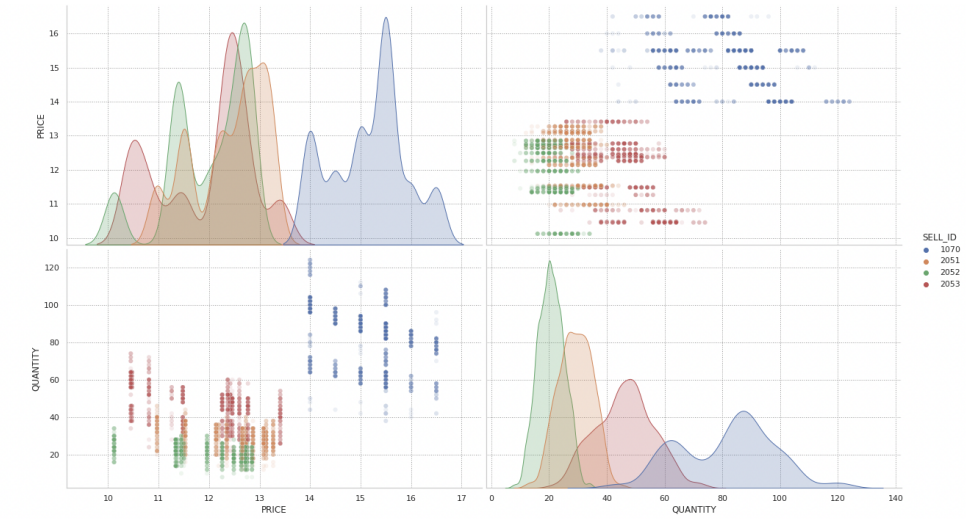

Figure 4.18: PRICE & QUANTITY graph

3. IS_WEEKEND & QUANTITY

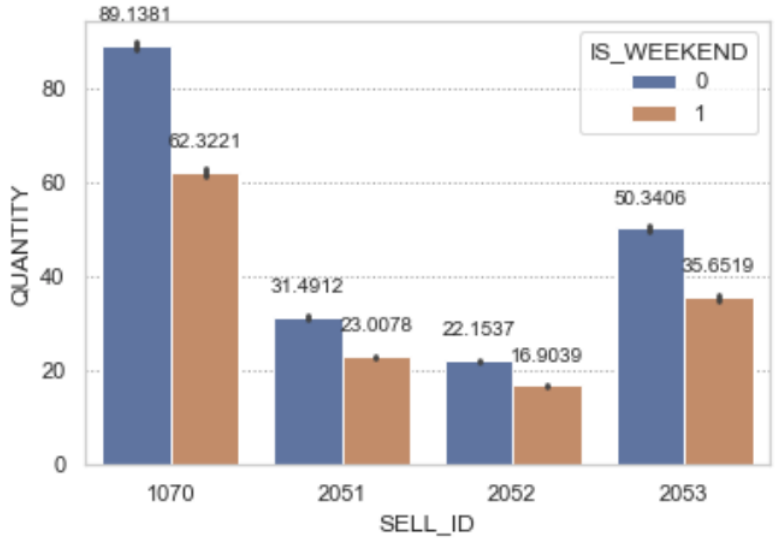

Figure 4.19: Correlation between QUANTITY, SELL_ID, IS_WEEKEND

According to Figure 4.19, we can see that customers buy more on weekdays, the most

is ID 1070 and the lowest is ID 2052. Also, on weekends, the number of sales is greatly reduced, in which:

- 30% for SELL_ID 1070
- 27% for SELL_ID 2051
- 23.7% for SELL_ID 2052
- 29.2% for SELL_ID 2053

Thus, we will only take non-weekend data, i.e. IS_WEEKEND = 0 to consider in our linear model.

4. IS_OUTDOOR & QUANTITY

This burger cafe is located in the building, so the fact that customers don't go out much will increase the number of burgers the shop sells. Because when not going out, they will use the service in the building more. This is shown in Figure 4.20 below:

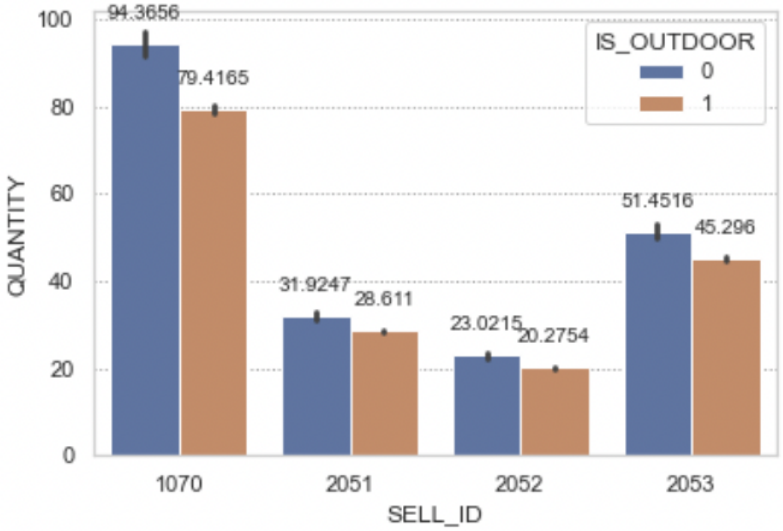

Figure 4.20: Correlation between QUANTITY, SELL_ID, IS_OUTDOOR

In terms of price, ID 1070 has the highest price and the highest volume of purchases, the selling price is almost unchanged when customers go out more or less (Figure 4.21). Similar in combos 2051, 2052, 2053 have a lower price and the selling price has not

changed significantly.

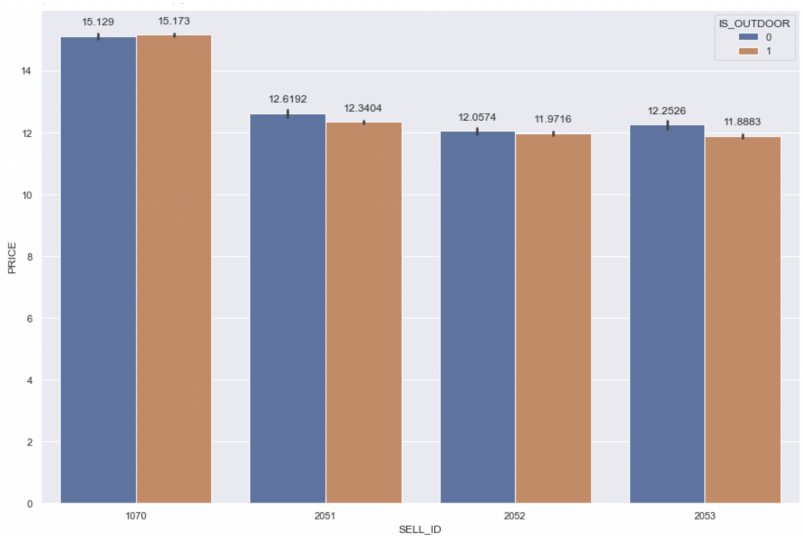

Figure 4.21: Correlation between PRICE, SELL_ID, IS_OUTDOOR

As shown in Figure 4.22, when the price increases in 4 products, the customer's demand will also decrease. Rising prices will also discourage customers from going out to buy less because they can choose another dish to eat. This makes perfect sense in economics, burger is just a normal product, with no income increase, when the price of burger increases, the quantity demanded for burgers will decrease.

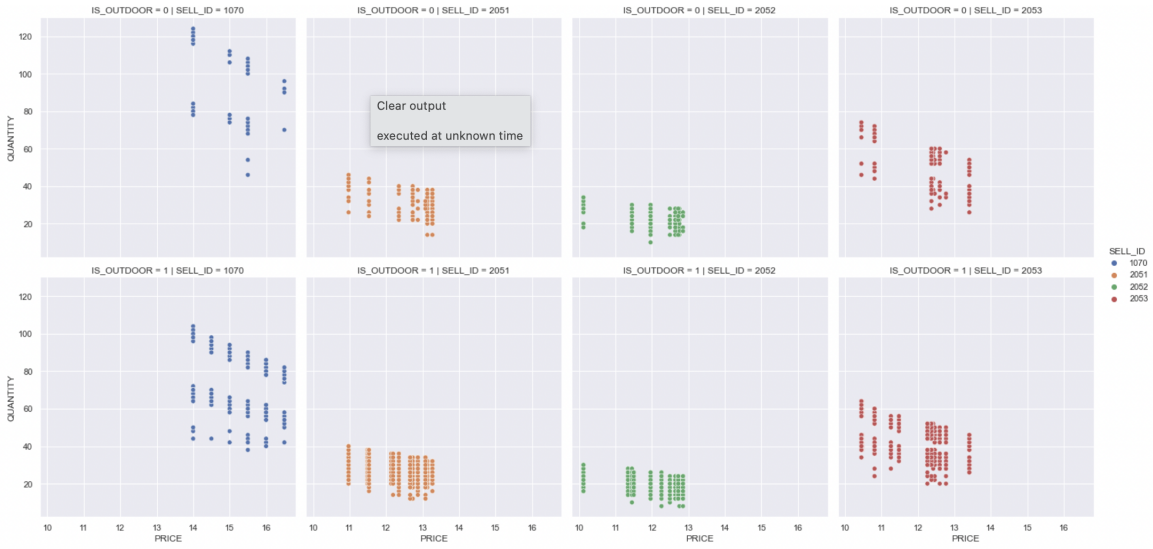

Figure 4.22: Correlation between QUANTITY, PRICE, IS_OUTDOOR

5. AVERAGE_TEMPERATURE & QUANTITY

Average temperature has little effect on the change in quantity sold, as shown in Figure 4.23:

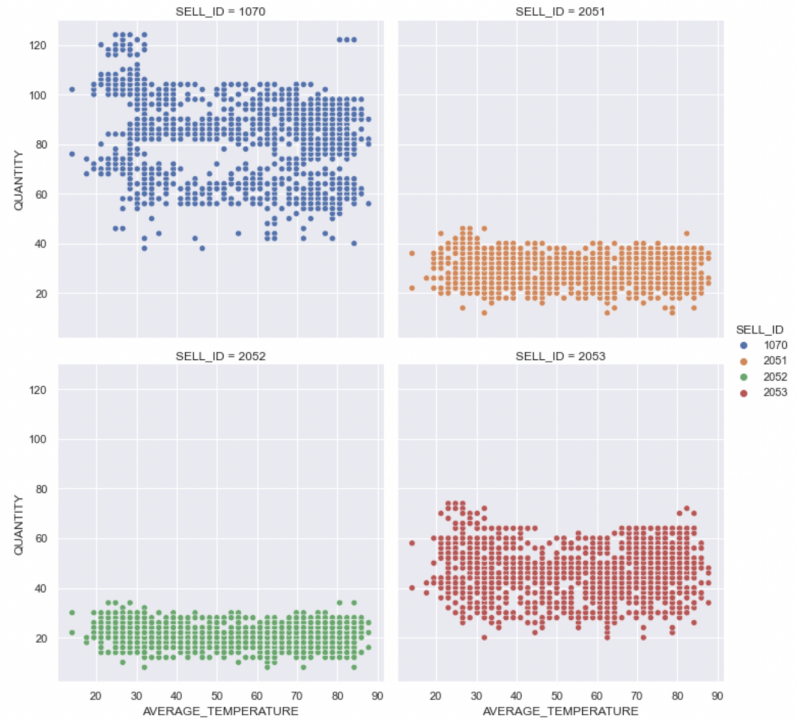

Figure 4.23: QUANTITY, SELL_ID, AVERAGE_TEMPERATURE chart

6. IS_SCHOOLBREAK & QUANTITY

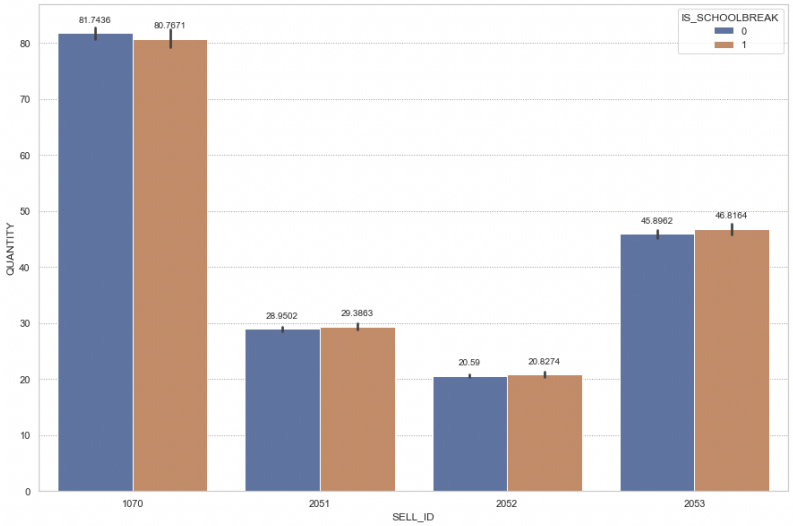

Figure 4.24: QUANTITY, SELL_ID, IS_SCHOOLBREAK chart

7. HOLIDAY & QUANTITY

Next, we will consider the variable HOLIDAY, as shown in Figure (4.25) below, No Holiday has a sudden increase in purchases compared to Holidays, the main reason is that on holidays customers will travel, stay at home, or eat other special food, and do on. So, the number of sales on Holiday is not much. Hence, we can consider omitting Holiday in our model, i.e. taking HOLIDAY = 'No Holiday' to make model better.

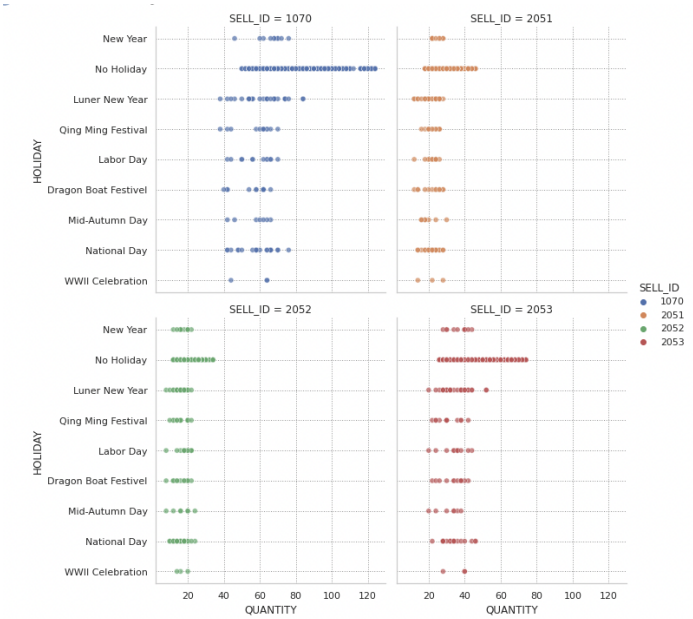

Figure 4.25: Correlation between HOLIDAY and QUANTITY

4.3.3 Outlier treatment

After analyzing the above correlations, we realized that in order to build a good regression model, we need to remove the factors affecting the quantity. Because when we remove like that, we get rid of the outliers. So, we proceed to build a new dataset by choosing variables as follows:

- HOLIDAY = 'No Holiday'
- IS_WEEKEND = 0
- IS_OUTDOOR = 1
- IS_SCHOOLBREAK = 0

4.4 Data Splitting

The train-test split is a way for measuring an algorithm's performance in machine learning. It may be utilized for problems regarding classification or regression, in addition to being applied to any supervised learning strategy. The dataset is going to be split up into two different groups using this procedure. The first subset to be used in the model-fitting process is called the training dataset. The second dataset is not used for model training; rather, it is used as an input to the model, and the model's predictions are then compared to the actual values in the dataset. In this context, 'test set' refers to the second data set that was examined.

Train Dataset: For fitting the machine learning model.
Test Dataset: For evaluating the fit machine learning model.

The goal is to evaluate the machine learning model's performance on fresh data—data that were not used to train the model. We expect using the model in this way. Specifically, to fit it to accessible data with known inputs and outputs, and then to generate predictions for future cases when we do not know the anticipated output or goal values. When a sufficiently large size dataset is provided, the train-test technique is acceptable.

In this thesis, we will divide our data with 80% for train and 20% for test.

Chapter 5

RESULTS

In this chapter, we will get results from three main parts: The Prediction Model, Elasticity Of Price and Price Optimization, as I mentioned in Introduction. First, The Prediction Model helps us to choose the best model for Elasticity Of Price part, then we apply it into Price Optimization and get the final result (the optimal price for each product).

5.1 The Prediction Model

Here, we will apply machine learning models to 3 main data sets: Data merge, Data bau and Data full. In which, Data merge is the original data (consider variables PRICE and QUANTITY); Data bau will consider PRICE and QUANTITY in special factors: No Holiday, IS_WEEKEND = 0 (weekdays), IS_OUTDOOR = 1 (people go out), IS_SCHOOLBREAK = 0 (students are not off); Data full is a model that will use factors other than PRICE (including IS_OUTDOOR, IS_WEEKEND, IS_SCHOOLBREAK) to predict QUANTITY, with the desire to find the optimal price in all conditions, without need to be considered separately for special cases.

5.1.1 Data merge

Table 5.1 in this section discusses the results of several models. The model was evaluated and compared mostly between two different categories, in which, they are Linear Regression and Tree-based method. The evaluation metrics used to contrast the mod-

els are the mean absolute error, the mean squared error, the root mean squared error, and the R^2 statistic. On the first 80% of historical data, all models were trained. The previous 20% data were utilized for the test data. We can see that Random Forest, XGBoost and Decision Tree gave better resutls than Linear Regression. Here, the results of the R-squared are quite low, the accuracy is just under 60% in all models. This is also shown in Figure 6.1, 6.2, 6.3,6.4.

Type	**Linear Regression**	**Random Forest**	**XGBoost**	**Decision Tree**
MAE	0.586	0.377	0.373	0.373
MSE	0.509	0.226	0.227	0.227
RMSE	0.713	0.476	0.476	0.476
R^2	0.118	0.606	0.606	0.606

Table 5.1: Evaluate each model in Data merge

5.1.2 Data bau

With data bau, consider the following facts: HOLIDAY='No Holiday', IS_WEEKEND = 0, IS_OUTDOOR = 1, IS_SCHOOLBREAK = 0. We have better results in table 5.2 for R^2 than results of Data merge in table 5.1. Most of which are above 80%, this is also clearly shown in Figures 6.5, 6.6, 6.7,6.8. In particular, the data is almost linear and fits the linear regression model. The indicators that evaluate this model are also very good (Figure 6.5).

Type	**Linear Regression**	**Random Forest**	**XGBoost**	**Decision Tree**
MAE	2.525	2.426	2.425	2.425
MSE	8.520	8.253	8.242	8.242
RMSE	2.918	2.872	2.870	2.870
R^2	0.817	0.823	0.823	0.823

Table 5.2: Evaluate each model

5.1.3 Data full

Data full for us to consider factors other than PRICE for predicting QUANTITY, with the desire to find the most optimal price without considering special cases. We get R^2

results in table 5.3 which are also quite good in the models. This demonstrates the power of the above algorithms with data.

Type	**Random Forest**	**XGBoost**	**Decision Tree**
MAE	4.600	4.589	4.589
MSE	57.896	57.845	57.845
RMSE	7.608	7.605	7.605
R^2	0.789	0.789	0.789

Table 5.3: Evaluate each model

5.2 Elasticity Of Price

In our goal of building a product's price elasticity, we will use linear regression model with outliers removed (linear regression of data bau). Because according to the results considered in tables 5.1, 5.2, 5.3, this model has the best results of R^2. So we'll use it to find price elasticity. We will consider each product separately by its SELL_ID (1070, 2051, 2052, 2053) and get metric scores in table 5.4.

Type	**MAE**	**MSE**	**RMSE**	R^2
SELL_ID = 1070	2.525	8.520	2.918	0.817
SELL_ID = 2051	2.563	8.746	2.957	0.369
SELL_ID = 2052	2.312	7.455	2.703	0.414
SELL_ID = 2053	2.443	8.120	2.849	0.811

Table 5.4: Evaluate each SELL_ID

With price elasticity lines $q_{QUANTITY} = a.p_{PRICE} + b$, we calculate the price elasticity (EpD) at each value point on the line, where a, b are in OLS Regression Result. Then, we have following results:

- Linear model of SELL_ID 1070: $q = -8.83p + 222.93$
- Linear model of SELL_ID 2051: $q = -3.60p + 75.93$
- Linear model of SELL_ID 2052: $q = -2.55p + 52.43$

- Linear model of SELL_ID 2053: $q = -5.89p + 120.67$

An absolute value of Price Elasticity of Demand (parameter α) greater than 1 means that a 1% change in price causes a change in quantity demanded by more than 1%. Here the absolute values are all greater than 1, meaning that when the price increases, people will switch to buy other products. Here, SELL_ID 1070 will have the biggest change in quantity sold if the store changes its selling price. This makes sense in economics.

5.3 Linear Programming

We apply the formula $profit = (sellingprice - cost) \times quantity$ with quantity from the above section. We fix the cost of making a product at \$9 because we find the minimum price in this data is \$10.12 and maximum is \$16.5. Then, we set the selling price from the lowest current selling price of -1 (here we use \$9.12) to the lowest current selling price of +10 (\$20.12). We get the following output for each SELL_ID:

Type	**Price**	**Quantity**	**Profit**
SELL_ID = 1070	17.1	72.009	583.273
SELL_ID = 2051	15.07	21.709	131.776
SELL_ID = 2052	14.82	14.711	85.618
SELL_ID = 2053	14.75	33.810	194.410

Table 5.5: Optimal price for each SELL_ID

Here, we can understand that with item number 1070, when the selling price is \$17.1, we will sell at most 72 products, and make a profit of \$583. Similar to other products, if we raise the price or lower the price, our profit will not be as optimal as the price at the highest point as shown in Figure 6.13, 6.14, 6.15, 6.16 .

Chapter 6

CONCLUSION AND DISCUSSION

With the problem of finding the optimal price based on the relationship of selling price and sales volume of retail products, we rely on the theory of price elasticity to find the optimal price with the determined cost. The machine learning model was used to linearize sales history data of sales price and sales volume over 1350 days of burger products. Through the steps of building a machine learning model, we found that linear regression is still a good and data-friendly method to apply to price elasticity content. In addition, we research combo related data and extended sales date data, including CALENDAR_DATE, DAY, MONTH, YEAR, HOLIDAY, IS_WEEKEND, IS_SCHOOLBREAK, AVERAGE_TEMPERATURE, IS_OUTDOOR. From there, we have eliminated outliers in the data when building the linear regression model by choosing the more common extended data values. Besides, we try to build a regression model when considering the above extended variables, using more optimal machine learning models and having tabular data with many outliers such as XGBoost, RandomForest, Decision tree. Through testing different machine learning models with different processed datasets, we conduct evaluation through metris MAE, MSE, RMSE, R^2 with very good results.

Currently, Food & Beverage finds it incredibly difficult to determine the optimal pricing and discounting for all of their items on a daily basis, which would increase their total net sales and profitability. To provide an accurate estimate of tomorrow's need, a model for demand forecasting is first implemented. The notion of price elasticity is then utilized to establish the product's demand at various price points. Finally, an op-

timization strategy using linear programming is used to determine the optimal pricing for each product to maximize overall revenue.

However, such results are not completely convincing as well as applicable in practice, to further develop the quality and features of the model requires more and better data. With experimental data as above, we try to clearly show how to conduct data analysis and build models as a basis and premise for future research and development.

Now, prices are chosen and changed every day, but in the future, we would like to broaden our efforts so that we may incorporate intra-day signals into our model and carry out intra-day dynamic pricing as a direct outcome of these efforts.

Bibliography

[1] https://viracresearch.com/xu-huong-va-trien-vong-nganh-fb-tu-cuoi-nam-2022-d

[2] Joel E. Urbany, Peter R. Dickson, Rosemary Kalapurakal. Price Search in the Retail Grocery Marke, 2018. Volume 60, Issue 2. https://journals.sagepub.com/doi/pdf/10.1177/002224299606000207

[3] MPSTME, Shah and Anchor Kutchhi Engineering College, Mumbai. A Review of Machine Learning and Deep Learning Applications, 2019. https://ieeexplore.ieee.org/abstract/document/8697857

[4] Giorgio Alfredo Spedicato, Christophe Dutang (1), Leonardo Petrini. Machine Learning Methods to Perform Pricing Optimization. A Comparison with Standard GLMs, 2018. https://hal.science/hal-01942038/

[5] Christian O'Leary, Conor Lynch, Rose Bain, Gary Smith, Diarmuid Grimes. A Comparison of Deep Learning vs Traditional Machine Learning for Electricity Price Forecasting (2014), https://ieeexplore.ieee.org/abstract/document/9476947

[6] https://arxiv.org/abs/2007.05216

[7] https://gallery.azure.ai/Experiment/Modeling-Price-Elasticity-Part-3-Price-El

[8] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. ACM, 785–794. https://dl.acm.org/doi/10.1145/2939672.2939785

[9] Rajan Gupta and Chaitanya Pathak. 2014. A Machine Learning Frame- work for Predicting Purchase by Online Customers based on Dynamic Pricing. In Complex Adaptive Systems. https://www.sciencedirect.com/science/article/pii/S187705091401309X

[10] Pekgün, P.; Menich, R.P.; Acharya, S.; Finch, P.G.; Deschamps, F.; Mallery, K.; Van Sistine, J.; Christianson, K.; Fuller, J. Carlson Rezidor Hotel Group Maximizes Revenue Through Improved Demand Management and Price Optimization. Interfaces 2013, 43, 21–36. [Google Scholar] [CrossRef]. https://www.jstor.org/stable/23481527

[11] Peng Ye, Julian Qian, Jieying Chen, Chen-hung Wu, Yitong Zhou, Spencer De Mars, Frank Yang, and Li Zhang. 2018. Customized Regression Model for Airbnb Dynamic Pricing. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining. ACM, 932–940. https://www.kdd.org/kdd2018/accepted-papers/view/customized-regression-model-for-airbnb-dynamic-pricing

[12] Zhen Peng; Qiang Huang; Yincheng Han. Model Research on Forecast of Second-Hand House Price in Chengdu Based on XGboost Algorithm, 2019. https://ieeexplore.ieee.org/abstract/document/8935894

[13] Giorgio Alfredo Spedicato, Christophe Dutang (1), Leonardo Petrini. Machine Learning Methods to Perform Pricing Optimization. A Comparison with Standard GLMs. https://hal.science/hal-01942038/

[14] Divya Gangwani Pranav Gangwani. Applications of Machine Learning and Artificial Intelligence in Intelligent Transportation System: A Review, 2021. https://link.springer.com/chapter/10.1007/978-981-16-3067-5_16

[15] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. An Introduction to Statistical Learning: with Applications in R. https://books.google.com.vn/books/about/An_Introduction_to_Statistical_Learning.html?id=qcI_AAAAQBAJ&printsec=frontcover&source=kp_read_button&hl=en&redir_esc=y#v=onepage&q&f=false

[16] Chris Brooks. Introductory Econometrics for Finance

[17] Dielman, T. E. (1986) A Comparison of Forecasts from Least Absolute Value and Least Squares Regression, Journal of Forecasting 5, 189–95

[18] Makridakis, S. (1993) Accuracy Measures: Theoretical and Practical Concerns, International Journal of Forecasting 9, 527–9

[19] MBabar, PHNguyen, VCuk, andIGKamphuis.2015.The development of demand elasticity model for demand response in the retail market

[20] Ralph E Gomory and William J Baumol. Integer programming and pricing. Econometrica: Journal of the Econometric Society (1960)

[21] Main Code `https://colab.research.google.com/drive/1EBZOC_DDdLJouyXFMxFEXv5eV_9ghH7Z#scrollTo=JnQ6tqMKis7Z`

Appendix A

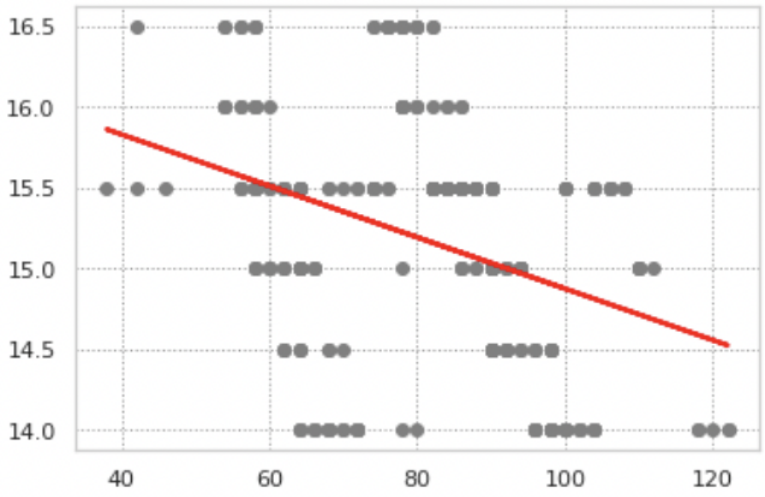

Figure 6.1: Linear regression of data merge

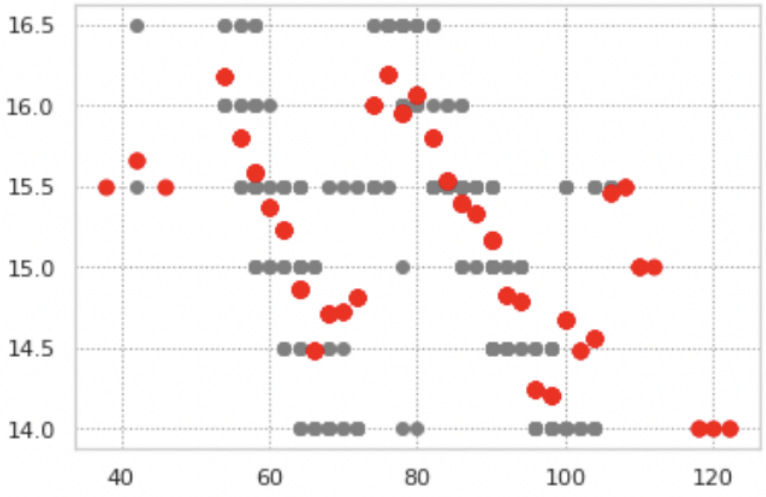

Figure 6.2: Decision tree of data merge

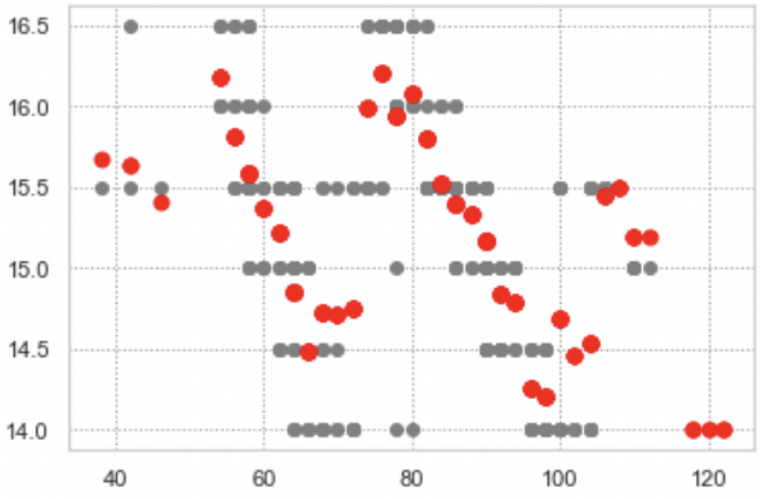

Figure 6.3: Random forest of data merge

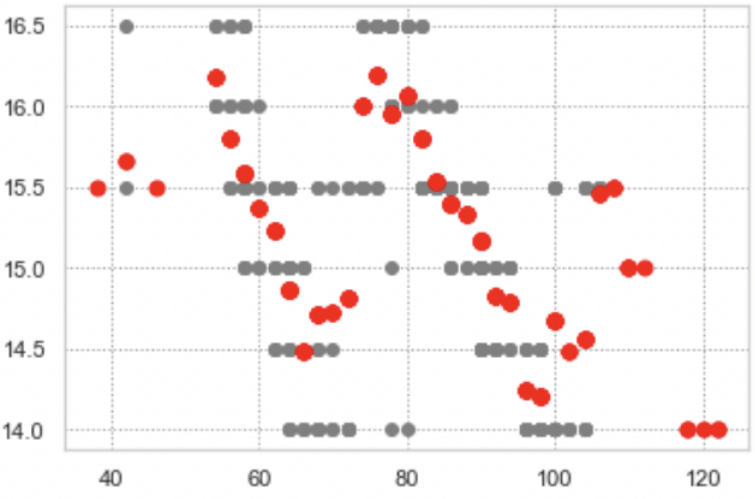

Figure 6.4: XGBoost of data merge

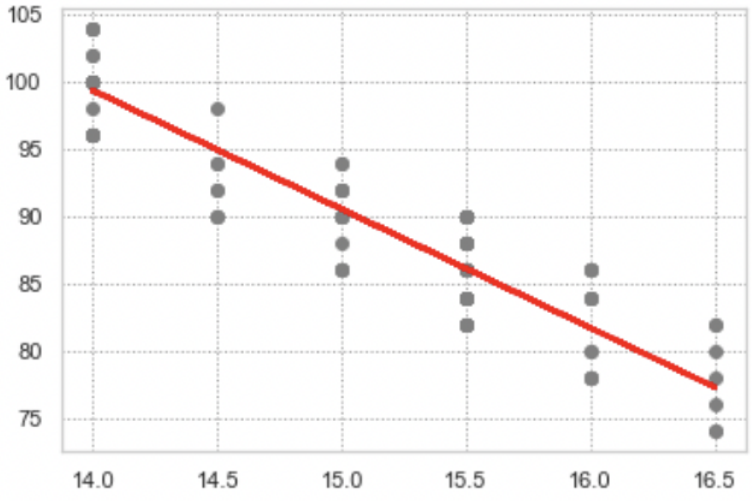

Figure 6.5: Linear regression of data bau

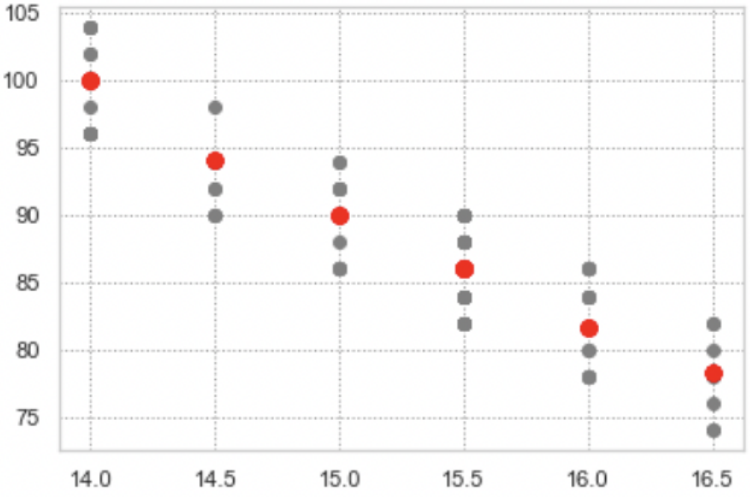

Figure 6.6: Decision tree of data bau

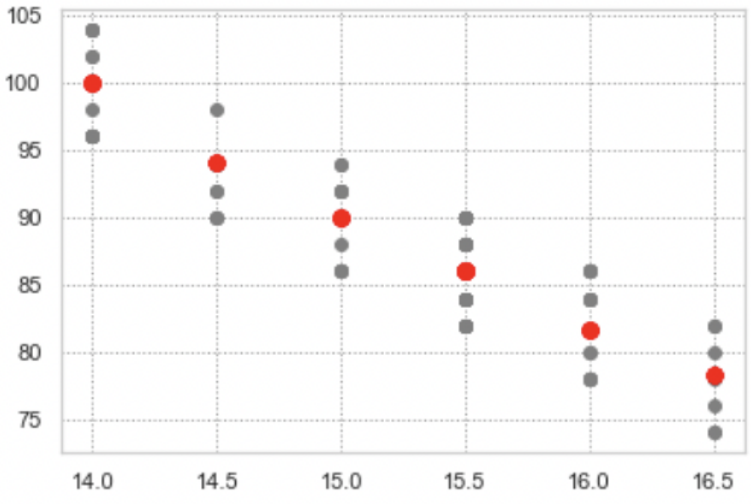

Figure 6.7: Random forest of data bau

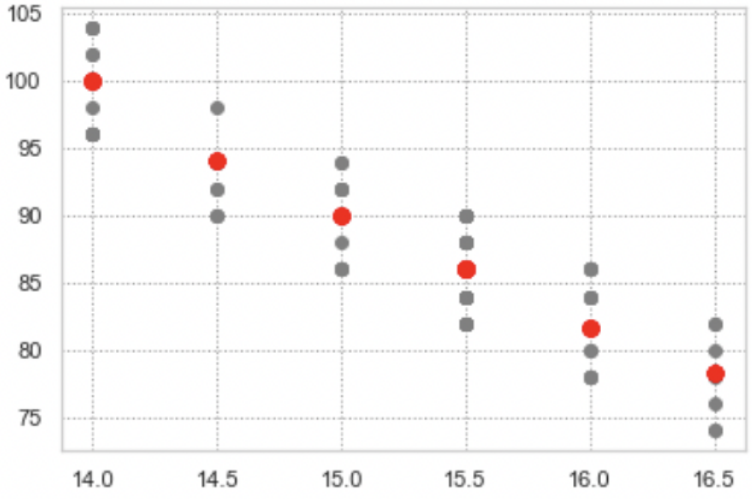

Figure 6.8: XGBoost of data bau

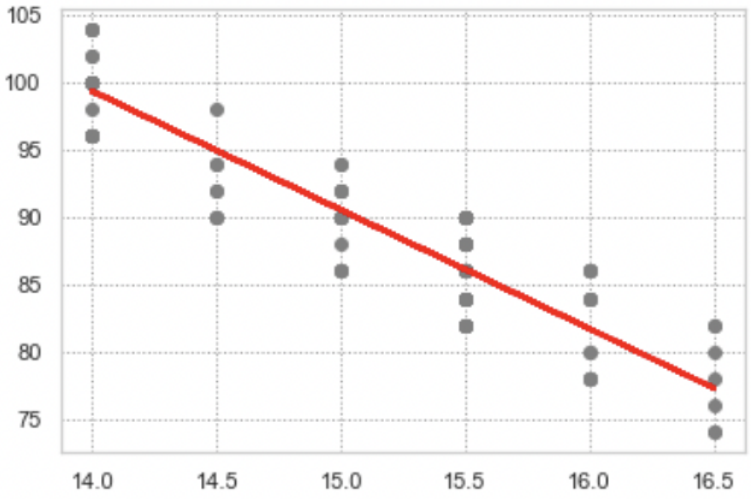

Figure 6.9: Linear regression of SELL_ID = 1070 of data bau

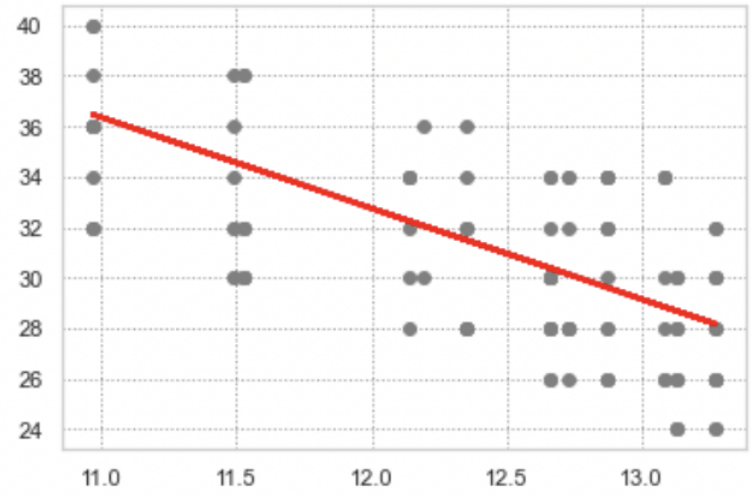

Figure 6.10: Linear regression of SELL_ID = 2051 of data bau

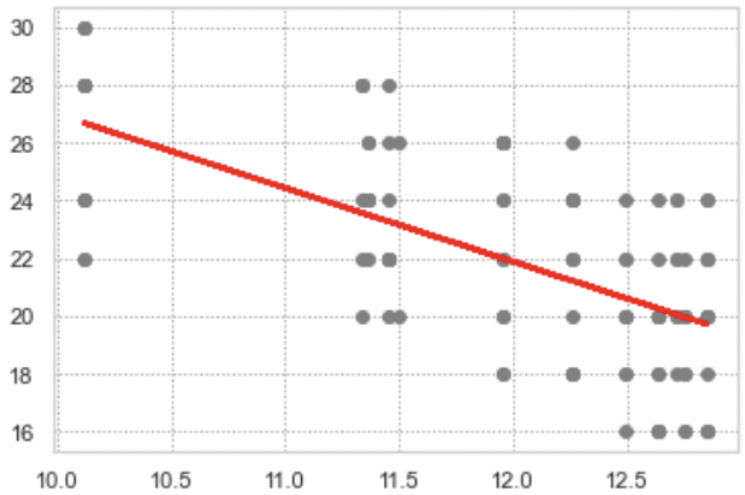

Figure 6.11: Linear regression of SELL_ID = 2052 of data bau

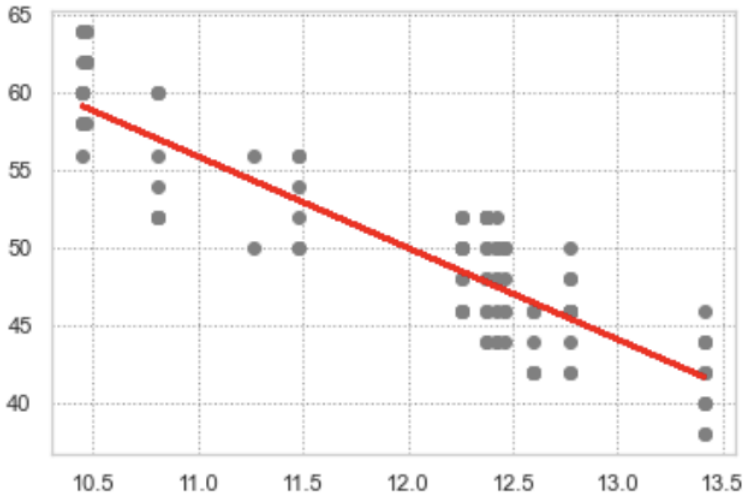

Figure 6.12: Linear regression of SELL_ID = 2053 of data bau

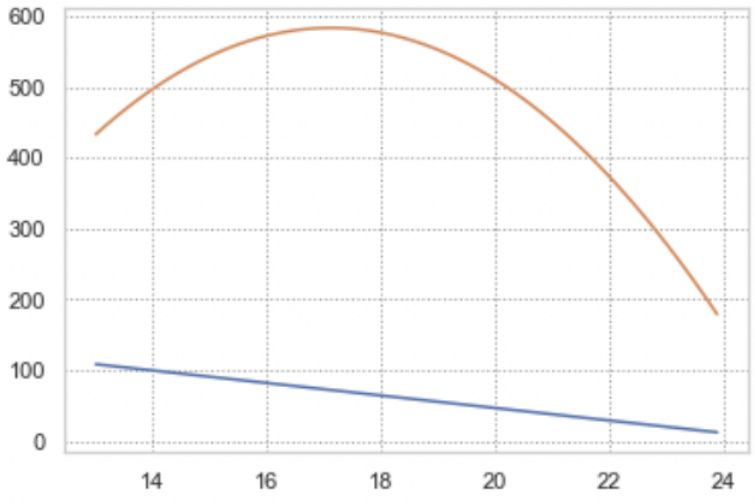

Figure 6.13: Optimal price and quantity of SELL_ID = 1070

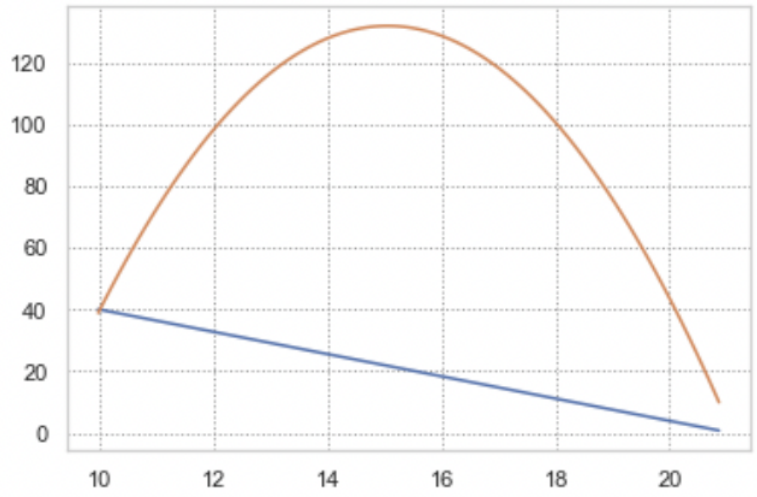

Figure 6.14: Optimal price and quantity of SELL_ID = 2051

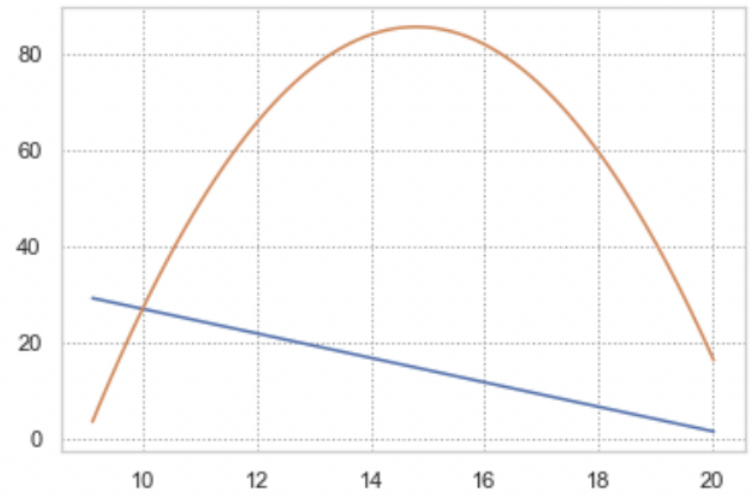

Figure 6.15: Optimal price and quantity of SELL_ID = 2052

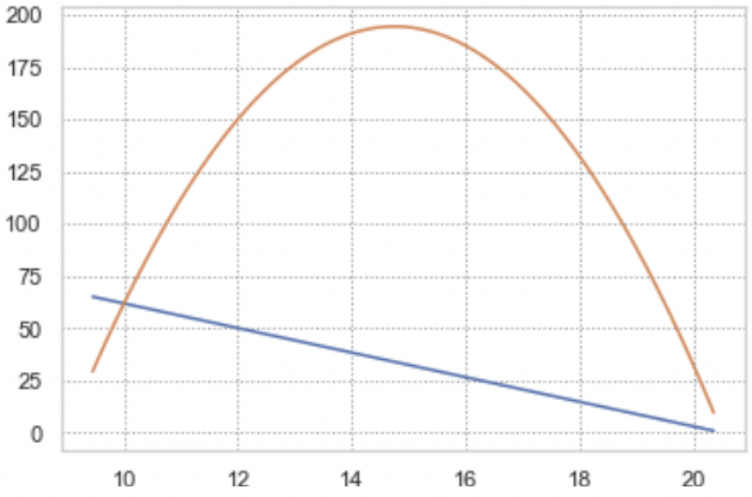

Figure 6.16: Optimal price and quantity of SELL_ID = 2053