Math 476 Project Report

Andy Chiv, Giovani Thai, Olivia Hartnett, Sarah Ellwein

July 26, 2022

Contributions

1. **Andy Chiv:** Data Cleaning, Statistical Analysis
2. **Giovani Thai:** Drafting Report, Data Cleaning
3. **Olivia Hartnett:** Statistical Analysis, Data Cleaning
4. **Sarah Ellwein**: Algorithm Implementation, Data Cleaning

Introduction

Purchasing Power Parity (PPP) is a method of calculating the amount of goods and services that a single unit of currency in one country can purchase in another. PPP efficiently compares prices between similar products purchased and consumed around the world. Traditionally, data collection for PPP came in the form of in-person identification across each major supermarket and for each product within said market. Having known this challenge, we pose the following research questions:

1. **How can products be classified using classification modeling?**
2. **How can we efficiently automate the calculation of PPPs across different stores and countries?**

These questions will be answered and achieved in three steps and explained more in depth under the Methodology section. Our algorithms will map onto a relevant data set provided by the Cal Poly DxHub. In partnership with Amazon AWS, Cal Poly's Digital Transformation Hub (DxHub) has provided a data set containing product information from four different stores: Walmart, Target, Noon, and Ubuy. These markets span North America and the Middle East, with product information ranging from ingredient lists to vendor names, to the occasional image and video urls.

The report is organized as follows: first, we introduce the approaches toward the problem, the methods for data cleaning, product classification, and unit standardization. Next, we describe the analysis conducted on the data set and display our findings. Finally, we conclude with major takeaways on key distinctions between products, as well as notes on improvement for further study.

Approaches

We observe that the data we have was scraped from two regions: North America and the Middle East. Majority of the products fall into food categories while some of them are appliances. Each store contains product description rather nicely that made data cleaning time-consuming. In order to efficiently calculate Purchasing Power Parity, we created a model to classify the products based on the product description (a.k.a product name), calculate the unit price of the product and convert them into a standardized metric form and lastly run a statistical analysis that compared price between the stores or regions.

To implement this pipeline, we start with the Naive Bayes algorithm based on Bayes theorem. This algorithm will train our model to associate certain words and vocabulary with a category.. Next we ensure that each product contains quantity and measurement, both of which are in metric form while price and unit price are in US Dollars. Last but not least, we deploy some statistical analysis such as t-test for two sample means and ANOVA for multiple sample means to compare product price between stores, countries, or regions. Since our data is scraped from 3 stores, we will solely focus on store-to-store comparison on their product prices.

Data Cleaning

In the *products.csv* file, it consists of four stores, one of which contains a very small sample size so we decided to only look at Walmart (USA), Target (USA), and Noon (UAE). For each store, we used **JSON, Pandas, and NumPy** to create a data frame that captured the information on total price, unit, quantity, and unit price. However each store has different variables and types of product descriptions, posting some challenges to clean them all up. To address these, we looked at one store at a time and found the specific variables we needed, using regular expression to find patterns of key characters and strings. In addition to obtaining the same type of information for each store, we also made sure to standardize units (metric) and currency (USD). In general, we were interested in the following features of the products:

Feature (Variable)	**Description**
uuid	product identification
category	product category
product name	product informative description
price	product total price ($)
unit	product measurement (kg or L) in metric
unit price	Product price ($/kg) or ($/L)

- **Walmart** has about 1362 products and is well-formatted, thus easy to clean. Of all the variables in this data set, the variable "product details" is in JSON format that store our desired variables such as product name, unit price, display price, amount, and displayed unit price. The inconsistent format in those variable required us to use **regular expression** to extract the **unit** and the **unit price** of the products.

- **Target** has 670 products, and we found the two variables (description 1 and description 2) were formatted as a list of JSON dictionary. Description 1 contains many adjectives that could help our descriptor pool whereas Description 2 has the important feature of the product: **net weight**. By looping a list of this JSON dictionary, we located the index of net weight and pulled out the amount and unit of measurement of the product.

- **Noon** has 1389 products and it is stored in an unorganized manner. We found that "product-details" provided all other relevant descriptors including the quantity and the unit of measurement. Even after we used **regular expression** to pull out that information, the quantity and the unit of measurement are not consistently formatted, giving some products like appliances unreliable.

Methodology

Step 1: Product Classification

Choosing a Model

The first step in product classification is analyzing our dataset and choosing our features. We decided to use the product's name/description as our feature, as it makes the most sense for people to classify products with. Next, we choose our model. Since we are classifying based on data, we must use **a supervised learning algorithm** to train our model to associate a product's features with its category. This rules out unsupervised techniques like K-means clustering and community detection. We also want to consider a model that works well with our feature. Because our feature is text data, we rule out models that won't make sense in the context of our problem, such as linear regression or topological data analysis. Within the classification algorithm, we finally chose Naive Bayes Classifier as our algorithm.

A Brief Overview of Naive Bayes: *Naive Bayes* is a supervised learning algorithm that applies Bayes' rule of probability to predict a class given a set of features, assuming the features are independent of each other. Suppose we have categories $c = \{c_1, ..., c_n\}$ and data point $\mathbf{x} \in d$-dimensional space. Then Naive Bay Theorem is given by

$$P(c_i|\mathbf{x}) = \frac{P(\mathbf{x}|c_i)P(c_i)}{P(\mathbf{x})}$$

We want to identify the class c_i for $\mathbf{x}$ based on the highest probability of c_i given features from $\mathbf{x}$. In other words, we want to find ${}_{c_i}\{P(c_i|\mathbf{x})\}$. In the context of our problem, our objective is to train our model to associate certain words and vocabulary with a category. From there, our model should take in a products name/description, parse its features, and find the best fit category based on probabilities.

Data Pre-processing

Next, we want to assure the features from the product name is standardized to be comparable. We implement the following procedure for preprocessing (refer to appendix for code):

1. Split the product name into a list of words.
2. Remove numbers and stop words (e.g. "the", "is", "are", etc.) from list.
3. For each word in the list, remove punctuation.
4. Extract the stem of the word (e.g. "crunching", "crunchy", "crunchable" $\rightarrow$ "crunch")

Implementation and Evaluation

After processing the features, we randomly shuffled our data then made a 80:20 split for training and testing. We resulted with approximately 73% accuracy with little variation.

Step 2: Unit Price Standardization

One of the goals in this project is to compare price differences across different stores or countries. Since each country has their own currency and products have different units, a standard unit of measurement is required. Since dollars is the most traded currency, we converted all price values to USD (dollars). Furthermore, we also chose to look at products that are of weight or volume and disregarded any measurement in length since most of our products are food and appliances. The following are the simplified steps:

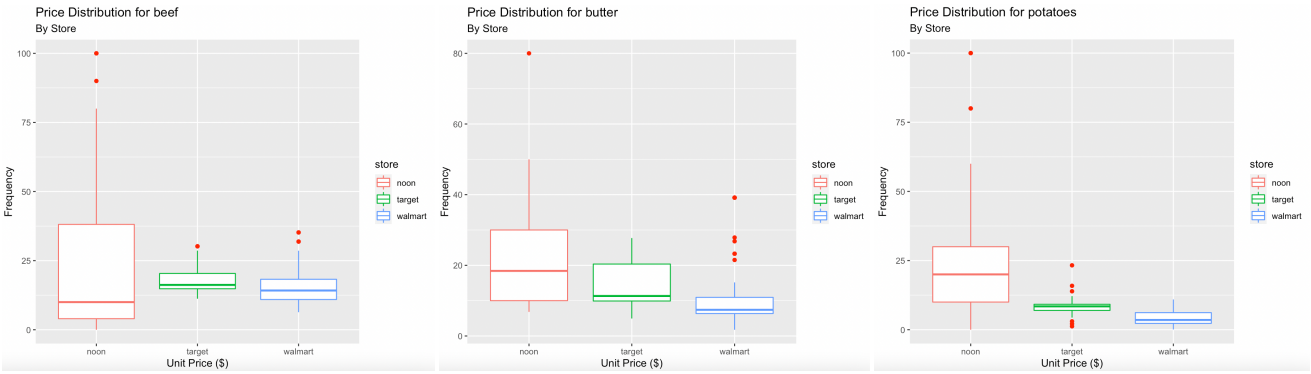
- Unit of Measurement $\rightarrow$ kilogram (kg) or liter (L)
- Unit Price = $\frac{Price}{Quantity}$ measured in ($ per unit)

Step 3: Statistical Analysis

Since we have a wide range of products, we decided to look at three specific products: beef, butter, and potatoes. In order to analyze and compare the prices for these three products across the three stores (Noon, Walmart, and Target), we performed analysis of variance (ANOVA) and plotted the distribution of the three products.

Before actually conducting the statistical analysis, we had to further clean some of the data that Naive Bayes Classification was unable to do. Because of the complex data set, we did have some extreme outliers in our data. To combat this, we decided to remove any product whose unit price is greater than $100. In addition, we also decided to not consider unit-less products labeled as "each", which were mostly appliances.

When we looked at the distribution for prices, we noticed Noon had some extreme outliers, as seen in the boxplots depicting the price distributions of the products. For all three products, Walmart and Target were close in price, with Target being slightly more expensive as expected. Noon had the most variability, and it did not consistently price its products higher or lower than Walmart and Target as speculated.



Performing ANOVA on the prices for beef, butter, and potatoes showed that Noon is most expensive and Walmart is least expensive. Our large F-statistics (5.2, 6.9, and 30.79, respectively) and small p-values (all ¡¡ 0.05) indicate that there is indeed a difference in price in at least one of the stores. But due to limitations in our data, we cannot trust these results.

Results

After analyzing this data set, we have found that Naive Bayes classifier was able to group the products based on the production description with up to 73% accuracy rate on average. This accuracy rate explains that for every 100 products in 33 categories, we are able to obtain about 73 products correctly classified. In addition to product classification, we created a pipeline that automates calculating the product prices between stores across multiple product categories.

Conclusion and Further Discussion

Overall, we have shown that the automation of PPP calculations can be broken down into three fundamental steps. This lays down a foundational outline of product classification, unit standardization, and statistical analysis. Within that, we have shown that the Naive Bayes algorithm can classify products into their intended categories with a high accuracy rate.

However, a main weakness of our implementation was that we were unable to validate the model due to limitations in available data. The largest obstacle in our way of tackling the automation of PPP was how messy the data

set was. Thus, further analysis could implement a wider variety of supervised learning algorithms for text analysis. Some examples are TF-IDF, which would weight defining product features more heavily; and Decision Trees, which would map product groups into branches, aiding in excluding unimportant features from product names. This would remove irrelevancies like brand names from product names to help the NB algorithm move along smoother.

In addition, delving deeper into product data can give our algorithm a larger pool of descriptors to work with. Some next steps would be to use more information from each store's product details section, as well as take a closer look at product sub-categories for analysis.

We hope that the next group of students who decide to take on this challenge by DxHub will take our suggestions into consideration when developing their model.

References

[1] Cal Poly DXHub Exploring methods to improve Purchasing Power Parity (PPP) calculations for the World Bank's International Comparison Program. *https://dxhub.calpoly.edu/challenges/improving-world-bank-ppp/*

Appendix

Data Preprocessing

```
import nltk
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
ps = PorterStemmer()
puncts = ['-', '"', "'", '(', ')', ',', '#', '@', ':', ';', '*', '  ', '    ', '.', '+', '!']

def has_numbers(inputString):
    return any(char.isdigit() for char in inputString)

def getFeatures1(name):
  features = {}
  words = [w.lower() for w in name.split() if w.lower() not in stopwords.words('english')]
  for word in words:
    feature = ps.stem(''.join([char for char in word if char not in puncts]))
    if len(feature) > 2 and not has_numbers(feature):
      features[feature] = True
  return features

def getData(row):
  return [getFeatures1(row['name']), row['product']]

allData = list(store.apply(getData, axis=1))
```

Naive Bayes Implementation

```
random.shuffle(allData)
testRatio = 5
splitpoint = len(allData) // testRatio
test, train = allData[:splitpoint],allData[splitpoint:]

nb = nltk.NaiveBayesClassifier.train(train)
print("NB accuracy: ", nltk.classify.accuracy(nb, test))
```