Market basket insights

Phase 3 submission document

Project tittle: Market Basket Insights

Phase 3 : *Development part 1*

Topic:

In this section begin building your project by loading and preprocessing the dataset.

Introduction

Market Basket Analysis is one of the key techniques used by large retailers to uncover associations between items. It works by looking for combinations of items that occur together frequently in transactions. To put it another way, it allows retailers to identify relationships between the items that people buy.

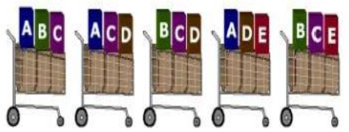
Association Rules are widely used to analyze retail basket or transaction data, and are intended to identify strong rules discovered in transaction data using measures of interestingness, based on the concept of strong rules.

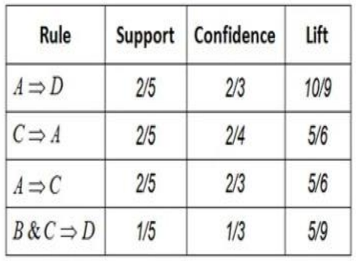
An example of Association Rules

- Assume there are 100 customers
- 10 of them bought milk, 8 bought butter and 6 bought both of them.
- bought milk => bought butter
- support = P(Milk & Butter) = 6/100 = 0.06
- confidence = support/P(Butter) = 0.06/0.08 = 0.75
- lift = confidence/P(Milk) = 0.75/0.10 = 7.5

$$\begin{tikzcd}
 & Support = \frac{frq(X,Y)}{N} \\
Rule:\ X \Rightarrow Y \arrow[ur] \arrow[r] \arrow[dr] & Confidence = \frac{frq(X,Y)}{frq(X)} \\
 & Lift = \frac{Support}{Supp(X) \times Supp(Y)}
\end{tikzcd}$$

Example:





Rule	Support	Confidence	Lift
$A \Rightarrow D$	2/5	2/3	10/9
$C \Rightarrow A$	2/5	2/4	5/6
$A \Rightarrow C$	2/5	2/3	5/6
$B \& C \Rightarrow D$	1/5	1/3	5/9

Load the packages

```
library(tidyverse)
library(readxl)
library(knitr)
library(ggplot2)
library(lubridate)
library(arules)
library(arulesViz)
library(plyr)
```

Data preprocessing and exploring

```
retail <- read_excel('Online_retail.xlsx')
retail <- retail[complete.cases(retail), ]
retail <- retail %>% mutate(Description = as.factor(Description))
retail <- retail %>% mutate(Country = as.factor(Country))
retail$Date <- as.Date(retail$InvoiceDate)
retail$Time <- format(retail$InvoiceDate,"%H:%M:%S")
retail$InvoiceNo <- as.numeric(as.character(retail$InvoiceNo))
glimpse(retail)
```

Each row of data represents a transaction for a particular item and the attributes correspond to the following:

InvoiceNo : Unique identifier for transaction

StockCode : Unique identifier for the stock item being purchased

Description : Description of item

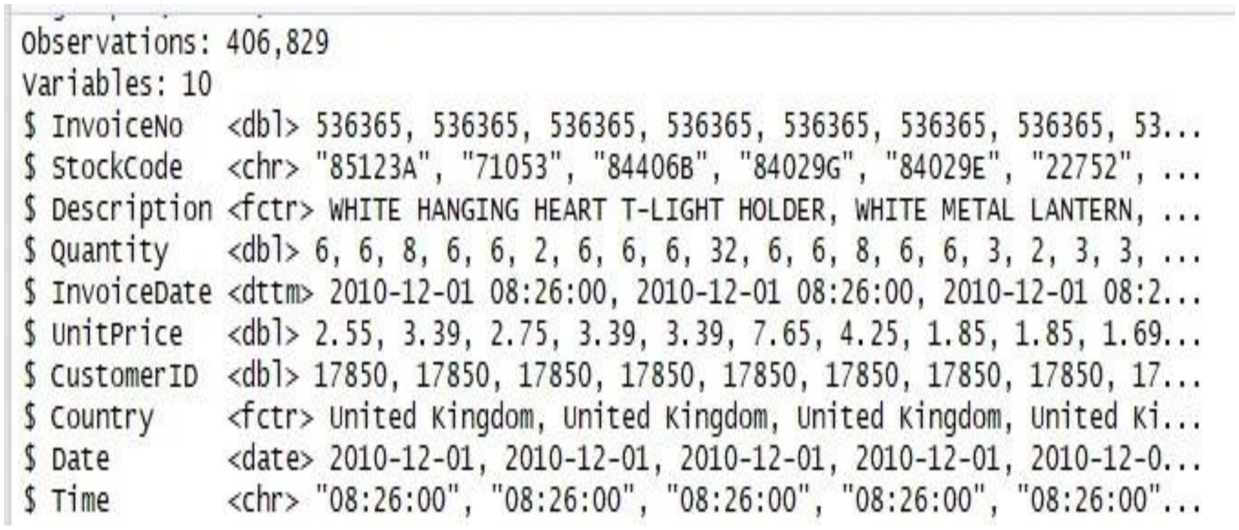
Quantity : Number of units purchased

InvoiceDate : Date of purchase

UnitPrice : Cost of one unit of the item

CustomerID : Unique Identifier for customer

Country : Country of transaction



```
Observations: 406,829
Variables: 10
$ InvoiceNo   <dbl> 536365, 536365, 536365, 536365, 536365, 536365, 536365, 53...
$ StockCode   <chr> "85123A", "71053", "84406B", "84029G", "84029E", "22752", ...
$ Description <fctr> WHITE HANGING HEART T-LIGHT HOLDER, WHITE METAL LANTERN, ...
$ Quantity    <dbl> 6, 6, 8, 6, 6, 2, 6, 6, 6, 32, 6, 6, 8, 6, 6, 3, 2, 3, 3, ...
$ InvoiceDate <dttm> 2010-12-01 08:26:00, 2010-12-01 08:26:00, 2010-12-01 08:2...
$ UnitPrice   <dbl> 2.55, 3.39, 2.75, 3.39, 3.39, 7.65, 4.25, 1.85, 1.85, 1.69...
$ CustomerID  <dbl> 17850, 17850, 17850, 17850, 17850, 17850, 17850, 17850, 17...
$ Country     <fctr> United Kingdom, United Kingdom, United Kingdom, United Ki...
$ Date        <date> 2010-12-01, 2010-12-01, 2010-12-01, 2010-12-01, 2010-12-0...
$ Time        <chr> "08:26:00", "08:26:00", "08:26:00", "08:26:00", "08:26:00"...
```

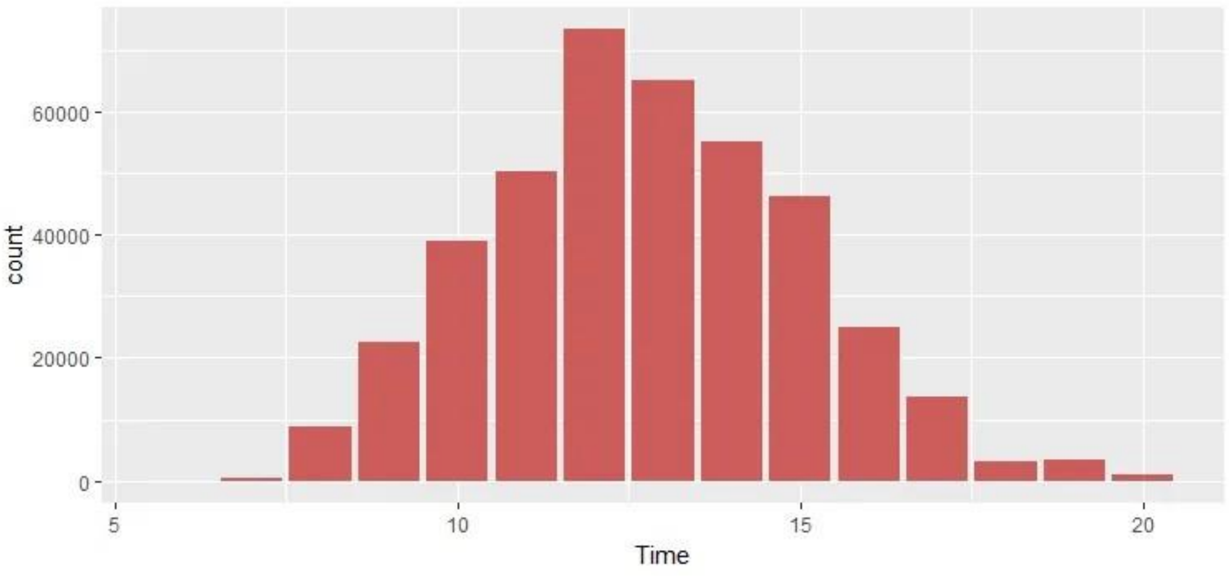
After preprocessing, the dataset includes 406,829 records and 10 fields

What time do people often purchase online?

In order to find the answer to this question, we need to extract “hour” from the time column.

```
retail$Time <- as.factor(retail$Time)
a <- hms(as.character(retail$Time))
retail$Time = hour(a)
    retail %>%
      ggplot(aes(x=Time)) +
      geom_histogram(stat="count",fill="indianred")
```

Figure 1. Shopping time distribution



There is a clear bias between the hour of day and order volume. Most orders happened between 10:00–15:00.

How many items each customer buy?

```
detach("package:plyr", unload=TRUE)retail %>%
  group_by(InvoiceNo) %>%
  summarize(n_items = mean(Quantity)) %>%
  ggplot(aes(x=n_items))+
geom_histogram(fill="indianred", bins = 100000) +
geom_rug()+
coord_cartesian(xlim=c(0,80))
```

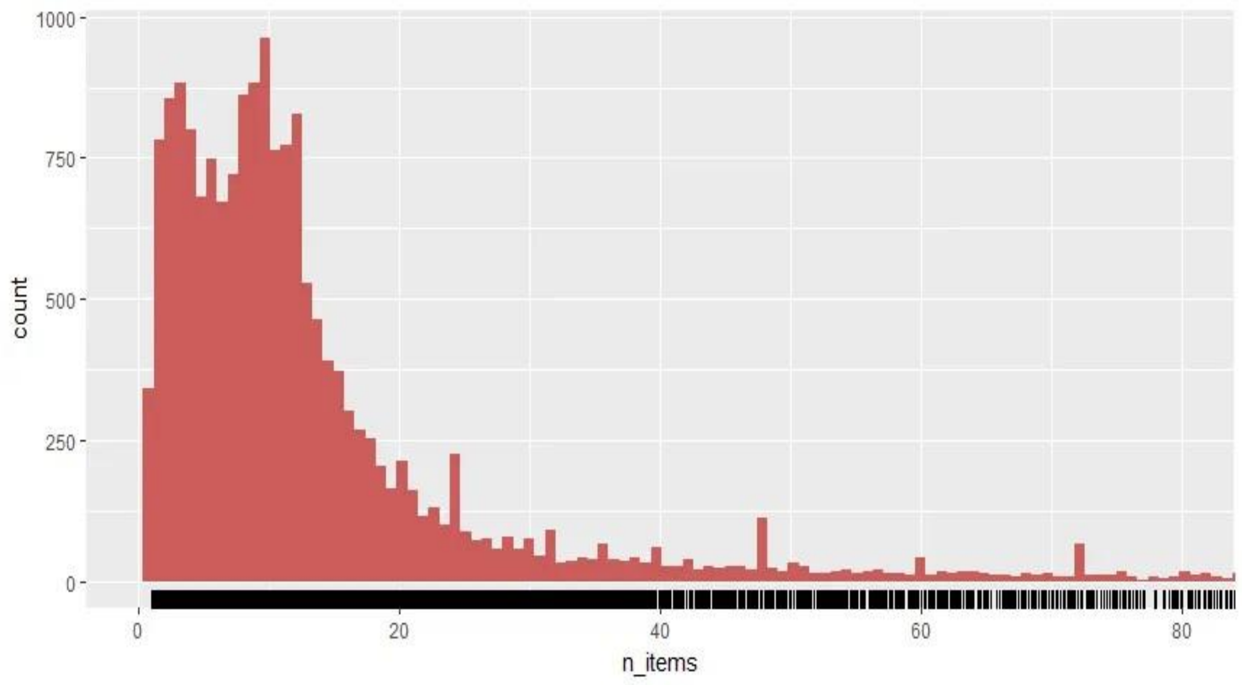


Figure 2. Number of items per invoice distribution

People mostly purchased less than 10 items (less than 10 items in each invoice).

Loading Libraries

First, you will load the libraries required. A short description of the libraries (taken from **Here**) is given in the following table, so you know what each library does:

Package	Description
`arules`	Provides the infrastructure for representing, manipulating and analyzing transaction data and patterns (frequent itemsets and association rules).

Package	Description
`arulesViz`	Extends package 'arules' with various visualization techniques for association rules and item-sets. The package also includes several interactive visualizations for rule exploration.
`tidyverse`	The tidyverse is an opinionated collection of R packages designed for data science
`readxl`	Read Excel Files in R
`plyr`	Tools for Splitting, Applying and Combining Data
`ggplot2`	Create graphics and charts
`knitr`	Dynamic Report generation in R
`lubridate`	Lubridate is an R package that makes it easier to work with dates and times.

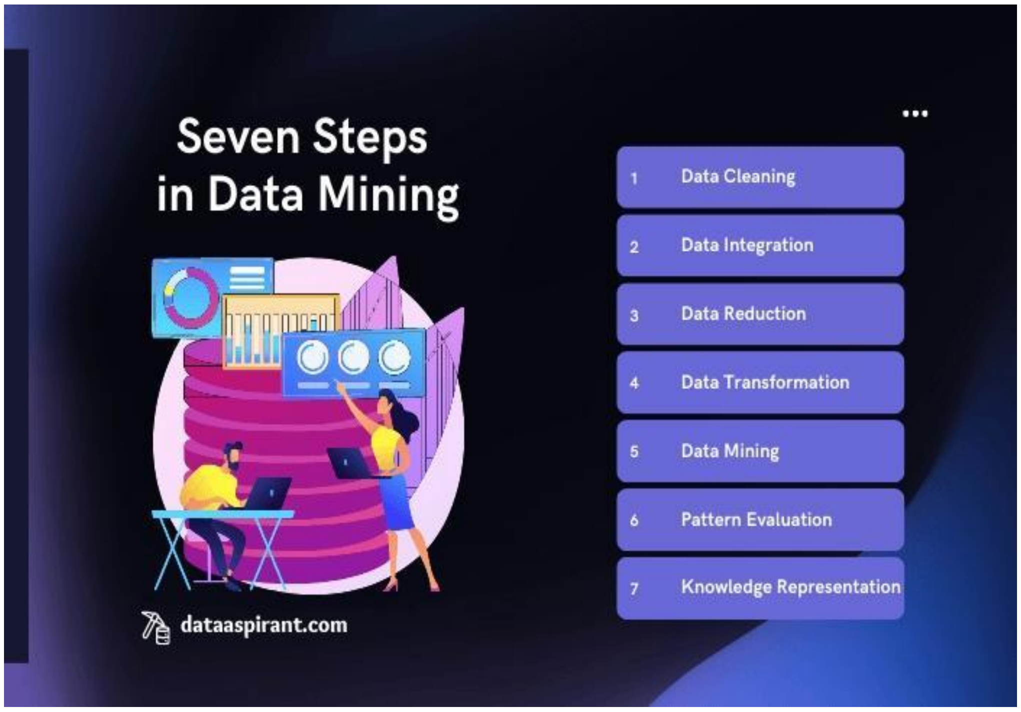
#install and load package arules

```
#install.packages("arules")

library(arules)

#install and load arulesViz

#install.packages("arulesViz")

library(arulesViz)

#install and load tidyverse

#install.packages("tidyverse")

library(tidyverse)

#install and load readxml

#install.packages("readxml")

library(readxl)

#install and load knitr

#install.packages("knitr")

library(knitr)

#load ggplot2 as it comes in tidyverse

library(ggplot2)

#install and load lubridate

#install.packages("lubridate")
```

```
library(lubridate)

#install and load plyr

#install.packages("plyr")

library(plyr)

library(dplyr)
```

Seven Steps in Data Mining Processing

Data mining is known as Knowledge Discovery in Databases or **KDD**. Identifying insightful knowledge from data includes various stages.



Gregory Piatetsky-Shapiro coined the term "Knowledge Discovery in Databases" in 1989.

On a high level, the knowledge discovery process has two main stages.

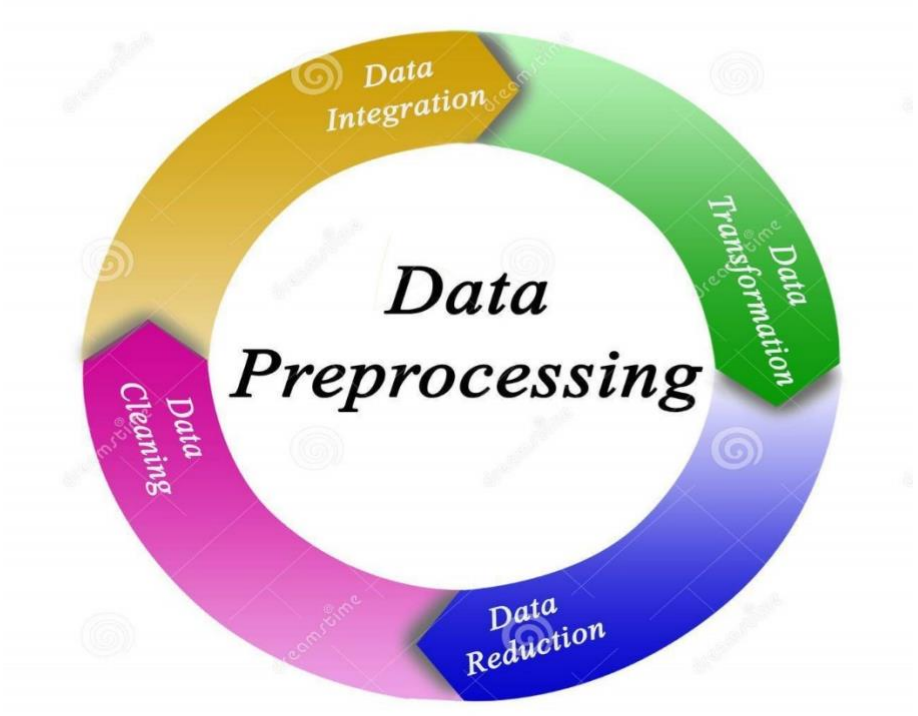
1. Data Preprocessing
2. Data Mining

Data Pre-processing

Use read_excel(path to file) to read the dataset from the downloaded file into R. Give your complete path to file including filename in read_excel(path-to-file-with-filename)

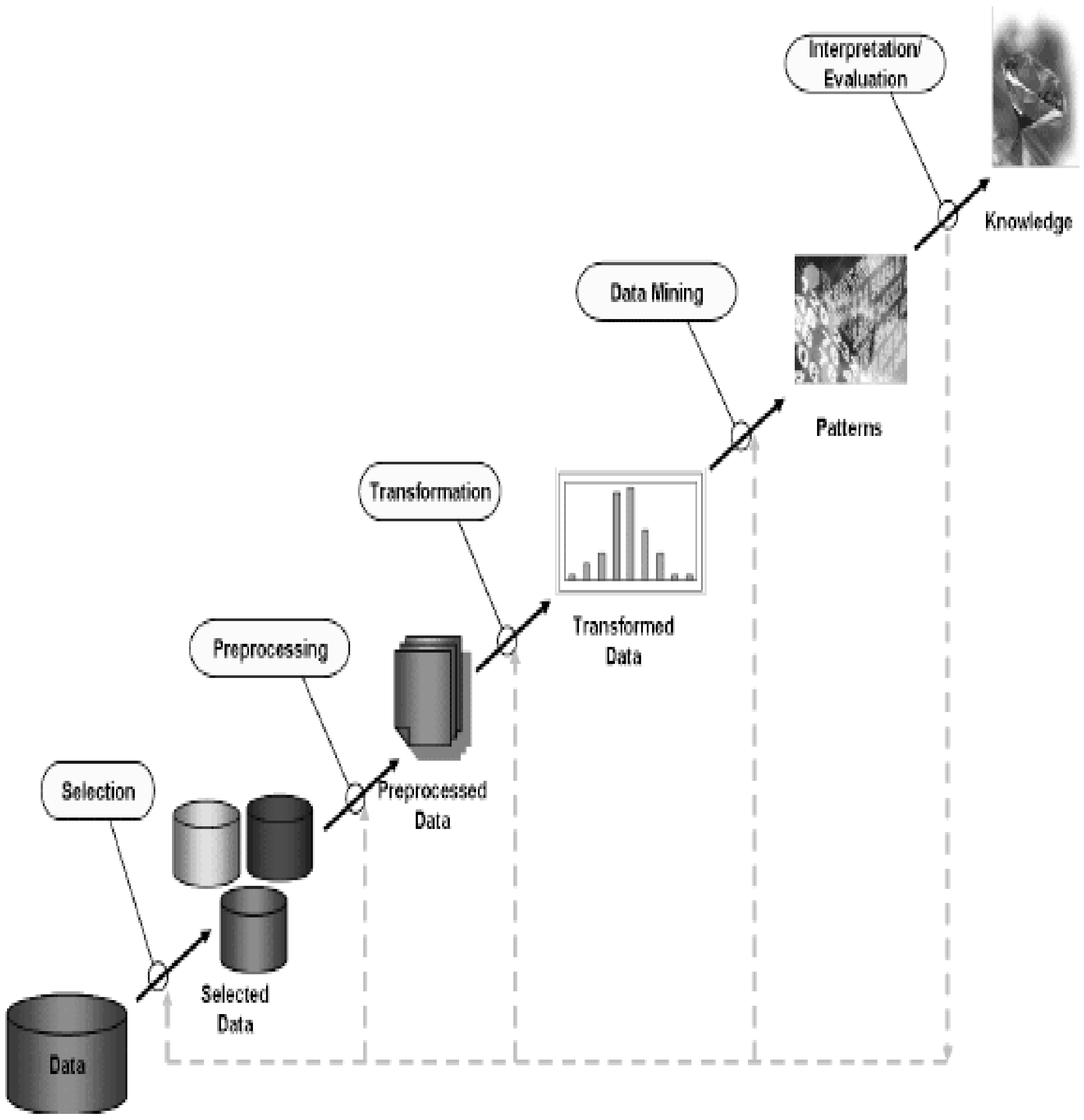
In the data preprocessing stage, we will perform the below steps.

- **Data cleaning**
    - To convert the raw data to a clean form of data by removing irrelevant data.
- **Data integration**
    - Combine the various data sources data to a centralized data source.
- **Data reduction**
    - Identify the relevant data and remove the data which are not helpful for any analysis.
- **Data transformation**
    - Transform the data to model-specific transformation, such as normalizing and converting the categorical data to numerical data.



Likewise, in the Data mining stage, we will perform the below steps

- **Data mining**
  - Apply various methods to get valuable insights out of the data.
- **Pattern evaluation**
  - Using the various statistical and machine learning models to identify the critical pattern inside the data.
- **Knowledge representation**
  - Represent the insights or the patterns we identified with **great visualizations**.



Data Processing

- Data Cleaning
- Transforming data to one transaction per row
- One Hot Encoding of purchases made

### Getting the list of transactions

Once we have read the dataset, we need to get the list of items in each transaction. SO we will run two loops here. One for the total number of transactions, and other for the total number of columns in each transaction. This list will work as a training set from where we can generate the list of association rules.

# Getting the list of transactions from the dataset

```
transactions = []
for i in range(0, len(data)):
    transactions.append([str(data.values[i,j]) for j in range(0, len(data.col
umns))])
```

```
transactions[:1]
```

```
[['shrimp',
  'almonds',
  'avocado',
  'vegetables mix',
  'green grapes',
  'whole weat flour',
  'yams',
  'cottage cheese',
  'energy drink',
  'tomato juice',
  'low fat yogurt',
  'green tea',
  'honey',
  'salad',
  'mineral water',
  'salmon',
  'antioxydant juice',
  'frozen smoothie',
  'spinach',
  'olive oil',
'Food']]
```

One-hot encoding transaction data

Throughout we will use a common pipeline for preprocessing data for use in market basket analysis. The first step is to import a pandas DataFrame and select the column that contains transactions. Each transaction in the column will be a string that consists of a number of items, each separated by a comma. The next step is to use a lambda function to split each transaction string into a list, thereby transforming the column into a list of lists. Then we will transform the transactions into a one-hot encoded DataFrame, where each column consists of TRUE and FALSE values that indicate whether an item was included in a transaction.

```python
# Import the transaction encoder function from mlxtend
from mlxtend.preprocessing import TransactionEncoder

# Instantiate transaction encoder and identify unique items
encoder = TransactionEncoder().fit(transactions)

# One-hot encode transactions
onehot = encoder.transform(transactions)

# Convert one-hot encoded data to DataFrame
onehot = pd.DataFrame(onehot, columns = encoder.columns_).drop('nan', axis=1)

# Print the one-hot encoded transaction dataset
onehot.head()
```

asparagus	Food	almonds	antioxydant juice	asparagus	avocado	babies food	bacon	barbecue sauce	black tea	blueberries	body spray	bramble	brownies	bugspray	burger sauce	burgers	butter	cake	candy bars	carrots	cauliflower	cra
0	False	True	True	True	False	True	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	F l

asparagus	Food	almonds	antioxydant juice	asparagus	avocado	babies food	bacon	barbecue sauce	black tea	blueberries	body spray	bramble	brownies	bug spray	burger sauce	burgers	butter	cake	candy bars	carrots	cauliflower	cra				
1	False	True	False	False	False	False	False	False	False	False	False	False	False	False	False	False	True	False	False	False	False	F				
2	False	True	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	F				
3	False	True	False	False	False	True	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	F				
4	False	True	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False

5 rows × 121 columns

Scatterplots

- Scatter plots will help us to evaluate general tendencies and behaviors of rules between many antecedents and consequents but, without isolating any rule in particular.

- **A scatterplot displays pairs of values.**
  - Antecedent and consequent support.
  - Confidence and lift.
- **No model is assumed.**
  - No trend line or curve needed.
- **Can provide starting point for pruning.**
  - Identify patterns in data and rules.

What can we learn from scatterplots?

- **Identify natural thresholds in data.**
  - Not possible with heatmaps or other visualizations.
- **Visualize entire dataset.**
  - Not limited to small number of rules.
- **Use findings to prune.**
  - Use natural thresholds and patterns to prune.

Pruning with scatterplots

After viewing your streaming service proposal from the previous exercise, the founder realizes that her initial plan may have been too narrow. Rather than focusing on initial titles, she asks you to focus on general patterns in the association rules and then perform pruning accordingly. Our goal should be to identify a large set of strong associations.

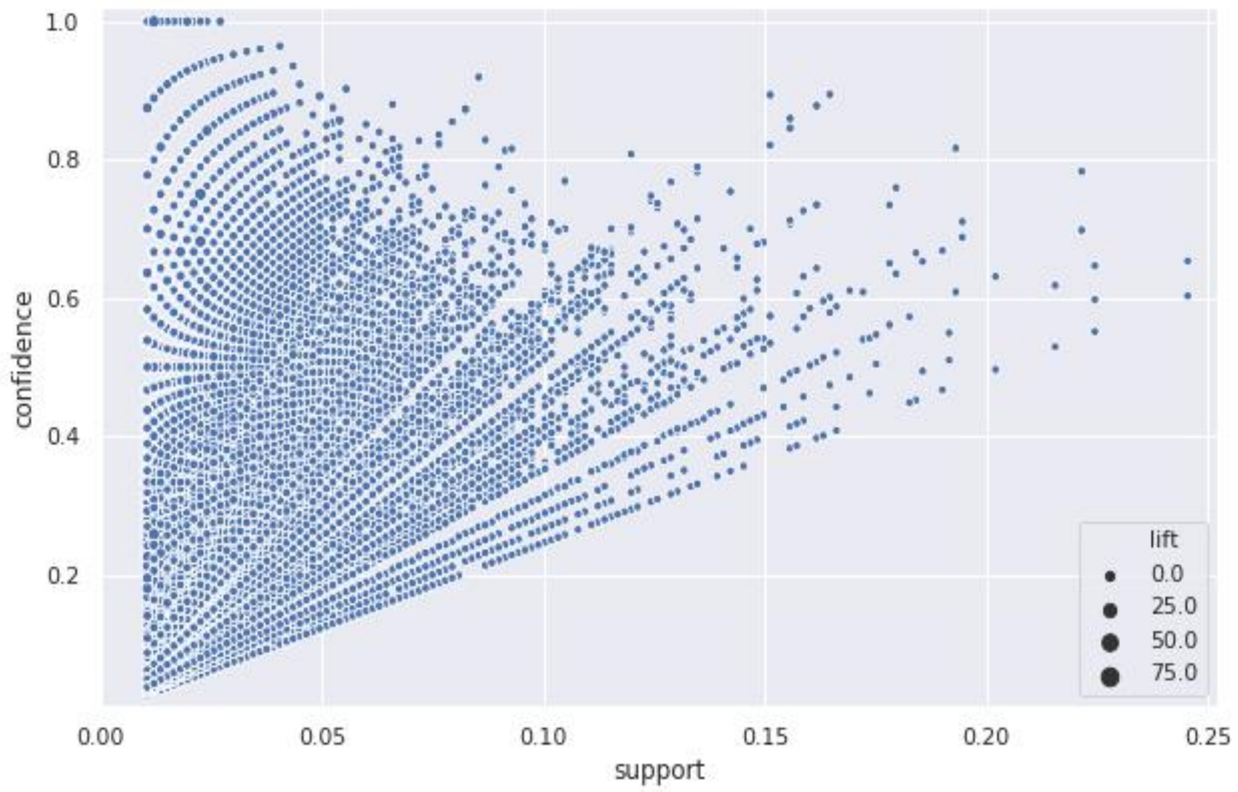
Fortunately, we've just learned how to generate scatterplots. We decide to start by plotting support and confidence, since all optimal rules according to many common metrics are located on the confidence-supply border.

```
## Apply the Apriori algorithm with a support value of 0.0095
frequent_itemsets = apriori(onehot, min_support = 0.0095,
                            use_colnames = True, max_len = 2)

# Generate association rules without performing additional pruning
rules = association_rules(frequent_itemsets, metric='support',
                          min_threshold = 0.0)
# Generate scatterplot using support and confidence
plt.figure(figsize=(10,6))
sns.scatterplot(x = "support", y = "confidence", data = rules)
plt.margins(0.01,0.01)
Plt.show()
```

Notice that the confidence-support border roughly forms a triangle. This suggests that throwing out some low support rules would also mean that we would discard rules that are strong according to many common metrics.

Optimality of the support-confidence border

We return to the founder with the scatterplot produced in the previous exercise and ask whether she would like us to use pruning to recover the support-confidence border. We tell her about the Bayardo-Agrawal result, but she seems skeptical and asks whether we can demonstrate this in an example.



Recalling that scatterplots can scale the size of dots according to a third metric, we decide to use that to demonstrate optimality of the support-confidence border. We will show this by scaling the dot size using the lift metric, which was one of the metrics to which Bayardo-Agrawal applies.

```
# Generate scatterplot using support and confidence
plt.figure(figsize=(10,6))
sns.scatterplot(x = "support", y = "confidence",
                size = "lift", data = rules)
plt.margins(0.01,0.01)
plt.show()
```

If you look at the plot carefully, you'll notice that the highest values of lift are always at the support-confidence border for any given value of confidence.

Parallel coordinates plot

- The parallel coordinates plot will allow us to visualize whether a relationship exist between an antecedent and consequent. We can think of it as a directed network diagram. The plot shows connections between 22 objects that are related and indicates the direction of the relationship.

When to use parallel coordinate plots

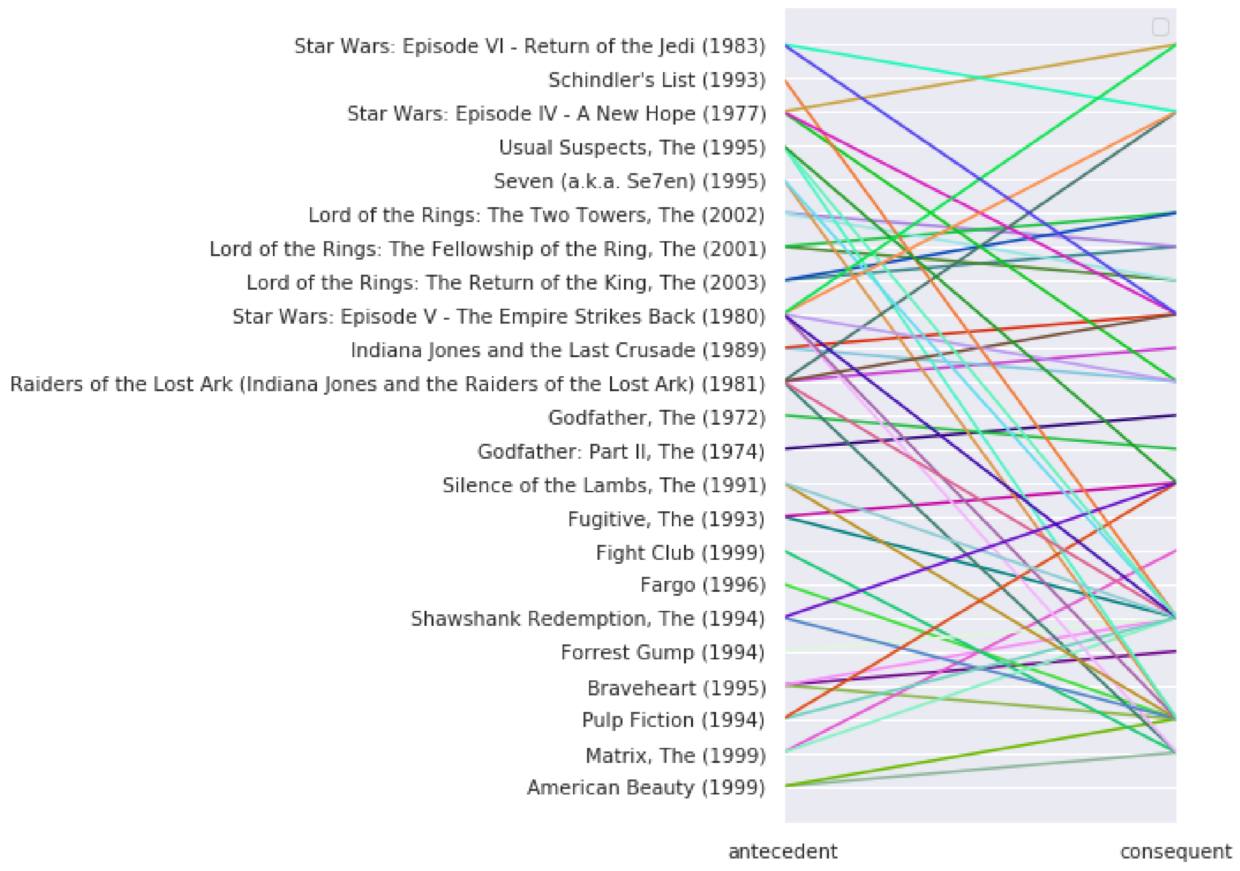
- **Parallel coordinates vs. heatmap.**
    - Don't need intensity information.
    - Only want to know whether rule exists.
    - Want to reduce visual clutter.

- **Parallel coordinates vs. scatterplot.**
    - Want individual rule information.
    - Not interested in multiple metrics.
    - Only want to examine final rules.

Using parallel coordinates to visualize rules

Our visual demonstration in the previous exercise convinced the founder that the supply-confidence border is worthy of further exploration. She now suggests that we extract part of the border and visualize it. Since the rules that fall on the border are strong with respect to most common metrics, she argues that we should simply visualize whether a rule exists, rather than the intensity of the rule according to some metric. We realize that a parallel coordinates plot is ideal for such cases.

```
# Function to convert rules to coordinates.
def rules_to_coordinates(rules):
    rules['antecedent'] = rules['antecedents'].apply(lambda antecedent: list(
antecedent)[0])
    rules['consequent'] = rules['consequents'].apply(lambda consequent: list(
consequent)[0])
    rules['rule'] = rules.index
    return rules[['antecedent','consequent','rule']]
from pandas.plotting import parallel_coordinates

# Compute the frequent itemsets
frequent_itemsets = apriori(onehot, min_support = 0.15,
                            use_colnames = True, max_len = 2)

# Compute rules from the frequent itemsets
rules = association_rules(frequent_itemsets, metric = 'confidence',
                          min_threshold = 0.55)

# Convert rules into coordinates suitable for use in a parallel coordinates p
lot
coords = rules_to_coordinates(rules)

# Generate parallel coordinates plot
plt.figure(figsize=(4,8))
parallel_coordinates(coords, 'rule')
plt.legend([])
plt.grid(True)
plt.show()
```



conclusion

In the quest to build a market basket insights model ,we have embarked on a critical journey that begins with loading and preprocessing the dataset. We have traversed through essential steps, starting with importing the necessary libraries to facilitate data manipulation and analysis.

- Understanding the data's structure, characteristics, and any potential issues through exploratory data analysis (EDA) is essential for informed decision-making.
- Data preprocessing emerged as a pivotal aspect of this process. It involves cleaning, transforming, and refining the dataset to ensure

that it aligns with the requirements of machine learning algorithms.

- With these foundational steps completed, our dataset is now primed for the subsequent stages of building and training a market basket insights model.

Thank you