Customer Behavior - Bakery

By: Urfi Nur Syamsiyah Yogabama

Last Updates: 03 June 2023

Overview

A bakery want to improve their sales through Market Basket Analysist with data from November 2016 to April 2017

Action

Bundled product can be combined with the discount of product besides, Coffee, Cake, and Tea

Tools

-Python
-Tableau
-Canva

METHODOLOGIES

01 Business Uderstanding

-Gather data from kaggle.com
-Understanding the problem from dataset

02 Data Preparation And Cleaning

-Define main problem
-Data Cleaning Using Python

03 Data Analysis and Modeling

-Define main problem
-Data Cleaning Using Python

04 Recommendation

-Propose Actionable action

Objective

- ☑ Identify association between products to increase selling
- ☑ Identify Customer Behavior based on the most time to buy

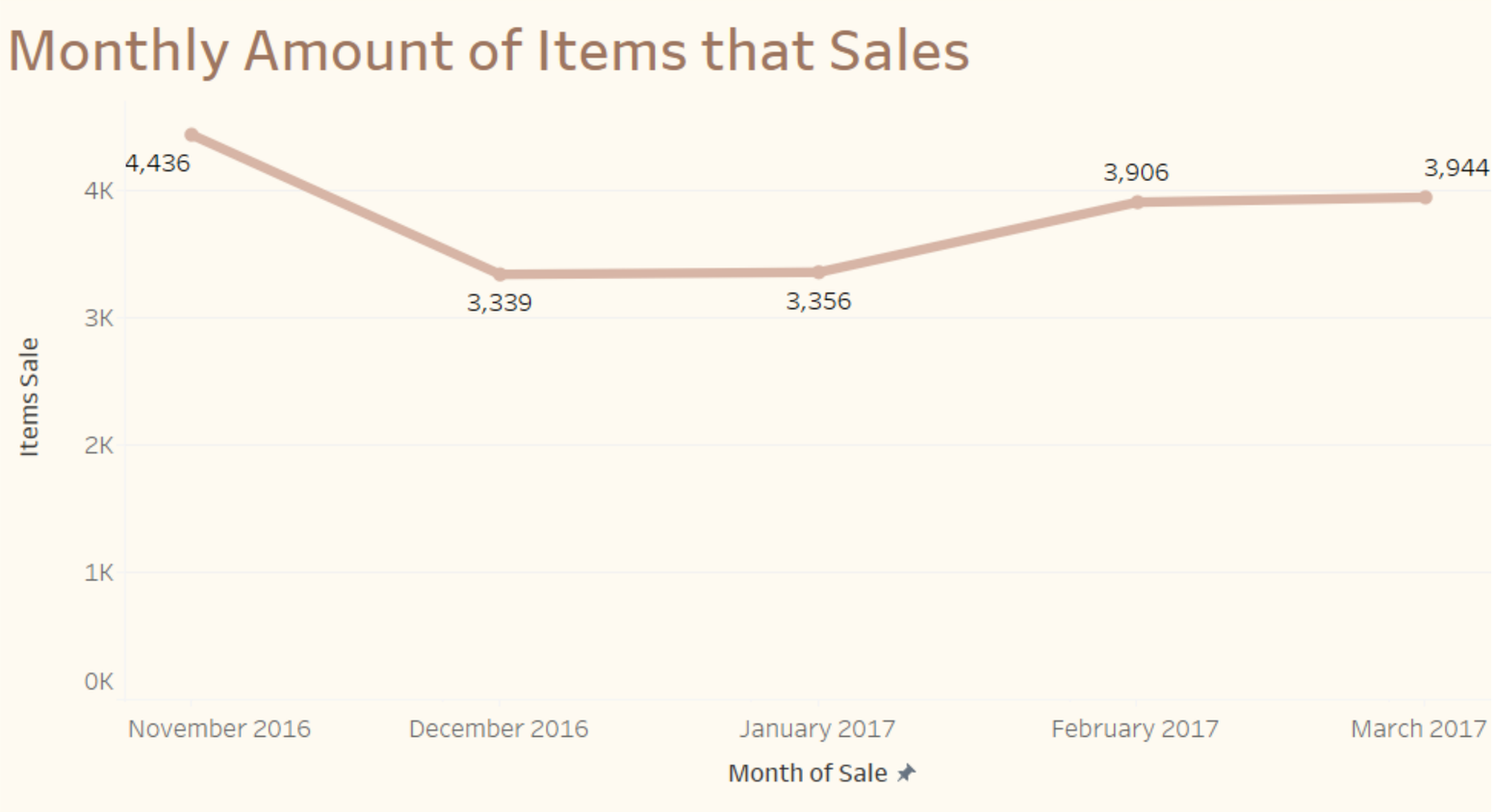
The Data

	Date	Time	Transaction	Item	year_month
0	2016-11-01	07:51:20	178	Coffee	2016-11
1	2016-11-01	07:51:20	178	Pastry	2016-11
2	2016-11-01	08:20:50	179	Coffee	2016-11
3	2016-11-01	08:20:50	179	Pastry	2016-11
4	2016-11-01	08:22:28	180	Jam	2016-11
...	...	...	...	...	...
18976	2017-03-31	16:53:39	9157	Coke	2017-03
18977	2017-03-31	16:53:39	9157	Coffee	2017-03
18978	2017-03-31	16:53:39	9157	Soup	2017-03
18979	2017-03-31	17:23:57	9158	Coffee	2017-03
18980	2017-03-31	17:23:57	9158	Juice	2017-03

Items that sales is around 3,3K to 4,5K monthly

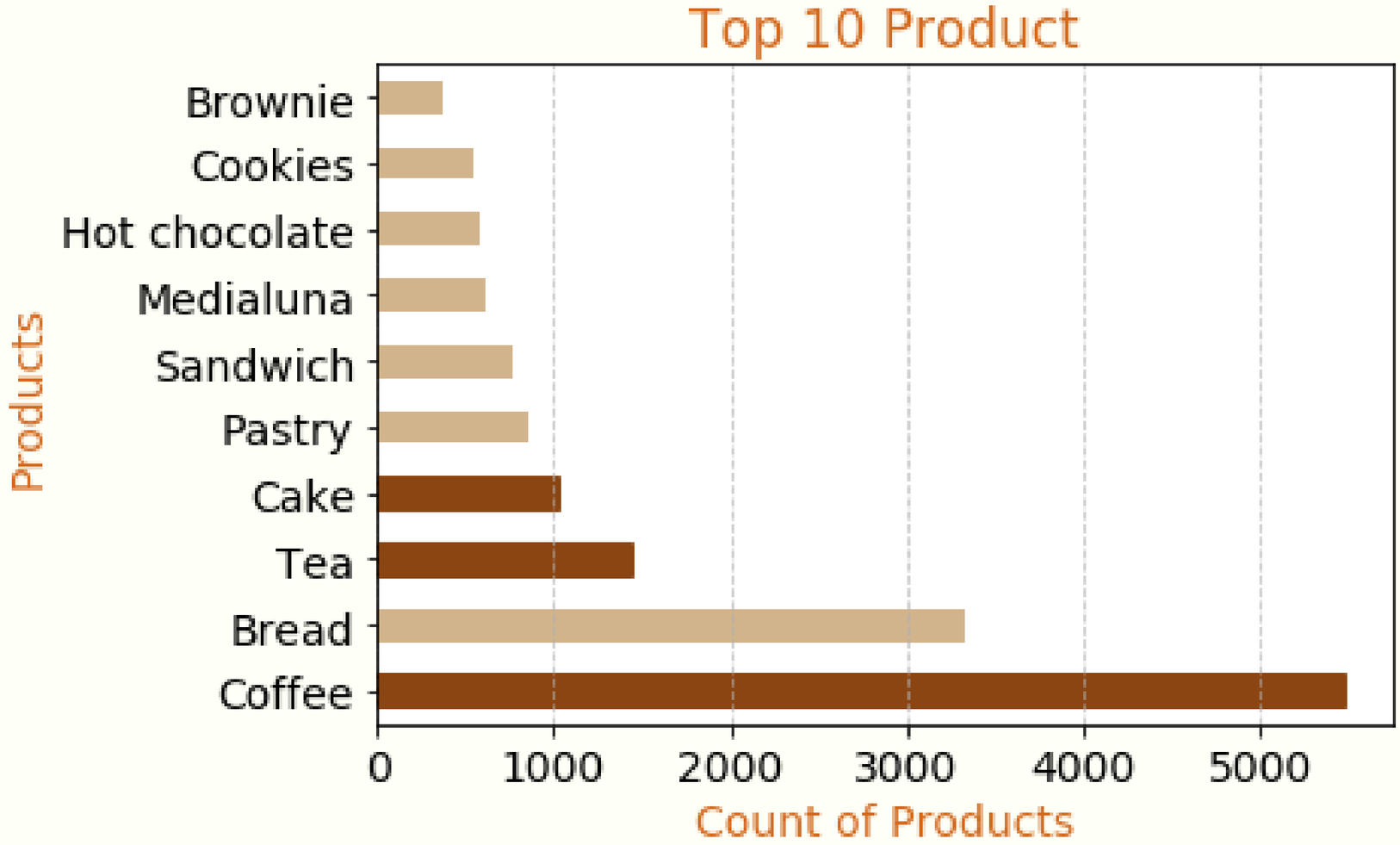
Data Overview

Our data shows list of Items Sales in a Bakery from November 2016 to March 2017



Products Overview

- Top Products for determining which product should be bundled
- Coffee, Tea, and Cake have good support and cofidence for bundling



Python code for Top 10 Products

```
#the color
faint_brown='#D2B48C'
dark_brown='#8B4513'
light_brown='#D2691E'

#count the products and making bar graph
df['Item'].value_counts()[:10].plot(kind='barh',color=faint_brown,fontsize=14)

#make the coffee, tea, and cake are dark brown
specific_bars=[0,2,3]
for idx in specific_bars:
    plt.gca().get_children()[idx].set_color(dark_brown)

#giving the title, x-axis, and y-axis
plt.title("Top 10 Product", fontsize=17, color=light_brown)
plt.xlabel("Count of Products", fontsize=14, color=light_brown)
plt.ylabel("Products",fontsize=14, color=light_brown)

#giving grid to the chart
plt.grid(axis='x',linestyle='--',alpha=0.7)
```

Bundle Recommendation

- The most bundle product recommendation are Coffee, Cake, and Tea that are 10th most buy product, based on the highest support and confidence value
- Bundled product can be combined with the discount of product besides, Coffee, Cake, and Tea
- Bundled product must be offered by waitress

Bundle	Support	Confidence
(Tea, Coffee) (Cake)	0.010037	0.201271
(Cake) (Tea)	0.023772	0.228891
(Toast) (Coffee)	0.023666	0.704403
(Spanish Brunch) (Coffee)	0.010882	0.598837
(Medialuna) (Coffee)	0.035182	0.569231
(Pastry) (Coffee)	0.047544	0.552147
(Alfajores) (Coffee)	0.019651	0.540698
(Juice) (Coffee)	0.020602	0.534247
(Sandwich) (Coffee)	0.038246	0.532353
(Cake) (Coffee)	0.054728	0.526958

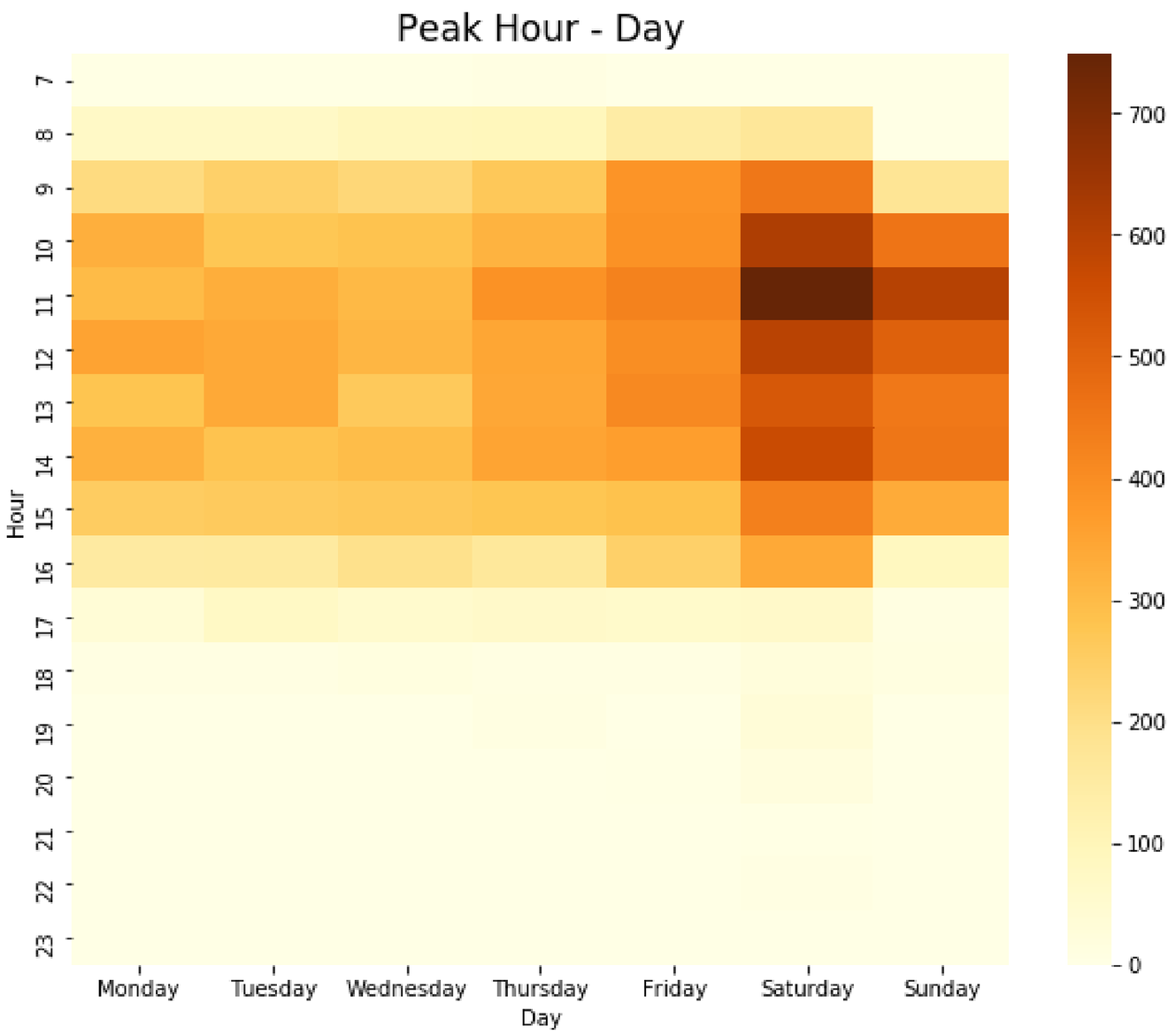
-Support is an indication of how frequently the item set appear in the dataset.
-Confidence is an indication of how often the rule has found to be true.

Python code for Bundle Recommendation

```
#transaction matrix
df_transaction=pd.DataFrame({'Transaction':df['Transaction'],'Item':df['Item']})
transaction_matrix=pd.pivot_table(data=df_transaction,
                                  index='Transaction',
                                  columns='Item',
                                  aggfunc=lambda x:1 if len(x)>0 else 0,
                                  fill_value=0)

#make the association rules table
frequent_items=apriori(transaction_matrix, min_support=0.01, use_colnames=True)
rules=association_rules(frequent_items,metric='lift',min_threshold=1)
rules=rules[rules['confidence']>0.2]
rules=rules[rules['lift']>1.1]
rules=rules.sort_values(by='lift',ascending=False).reset_index(drop=True)

#adding a column, called 'bundle'
rules['antecedents']=rules['antecedents'].astype('str').str.replace('frozenset','').str.replace('{','').str.replace('}','').str
rules['consequents']=rules['consequents'].astype('str').str.replace('frozenset','').str.replace('{','').str.replace('}','').str
rules['bundle']=rules.apply(lambda row: row['antecedents'] + ' ' + row['consequents'], axis=1)

#makes the rules table ready
rules_table=pd.DataFrame({'Bundle':rules['bundle'],'Support':rules['support'],'Confidence':rules['confidence']})
def set_background_color(value):
    return f'background-color: #D2B48C'
style_rules=rules_table.style.applymap(set_background_color)
style rules
```

Timing Recommendation

- The bakery should close at 6pm since the transaction is too litle
- Pushing notification for product recommendation related on every 11am and promo at friday, saturday, and sunday at 11 am



Python code for Timing Recommendation

```
#separate day and time data
df['Time']=pd.to_datetime(df['Time'])
df['hour']=df['Time'].dt.hour
df['day_name']=df['Date'].dt.day_name()

#make the day-hour matrix
day_hour_matrix=pd.pivot_table(data=df,
                               index='hour',
                               columns='day_name',
                               aggfunc='size',
                               fill_value=0)

#Make the day in order
day_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
day_hour_matrix=day_hour_matrix.reindex(columns=day_order)

#plot the heatmap
plt.figure(figsize=(10,8))
sns.heatmap(day_hour_matrix,cmap='YlOrBr')
plt.title("Peak Hour - Day",fontsize=17)
plt.xlabel("Day")
plt.ylabel("Hour")
```

Recommendation

- [x] Product that bundled with Coffee, Cake, and Tea as the top 10th Product bought, should be discount, so customer buy more

- [x] Pushing notification for product recommendation related on every 11am and promo at friday, saturday, and sunday at 11 am

Bohery

Thank You.