College of Computer Science
and Engineering

Supermarket Sales Analysis

IS372 - Data Warehouse and Data Mining

Student name:

Farha Ahmed Arief – 4050971

Fedhah Mesned Almutairi -4050392

Balqis Anwar Bakr - 4050751

Layal Emad Kherais – 4051179

Class: IA8G

Professor's Name: Mohammed Al-Sarem

Content:

Timeline..... 2

abstract..... 3

Introduction & method..... 4

Screenshot for code 5

Data Preprocessing…….. 5

Statistical Analysis……..... 7

Data Visualization …….. 10

Decision Tree..... 13

Kmeans..... 14

Naive Bayes..... 17

PCA……..... 19

Apriori Algorithm..... 20

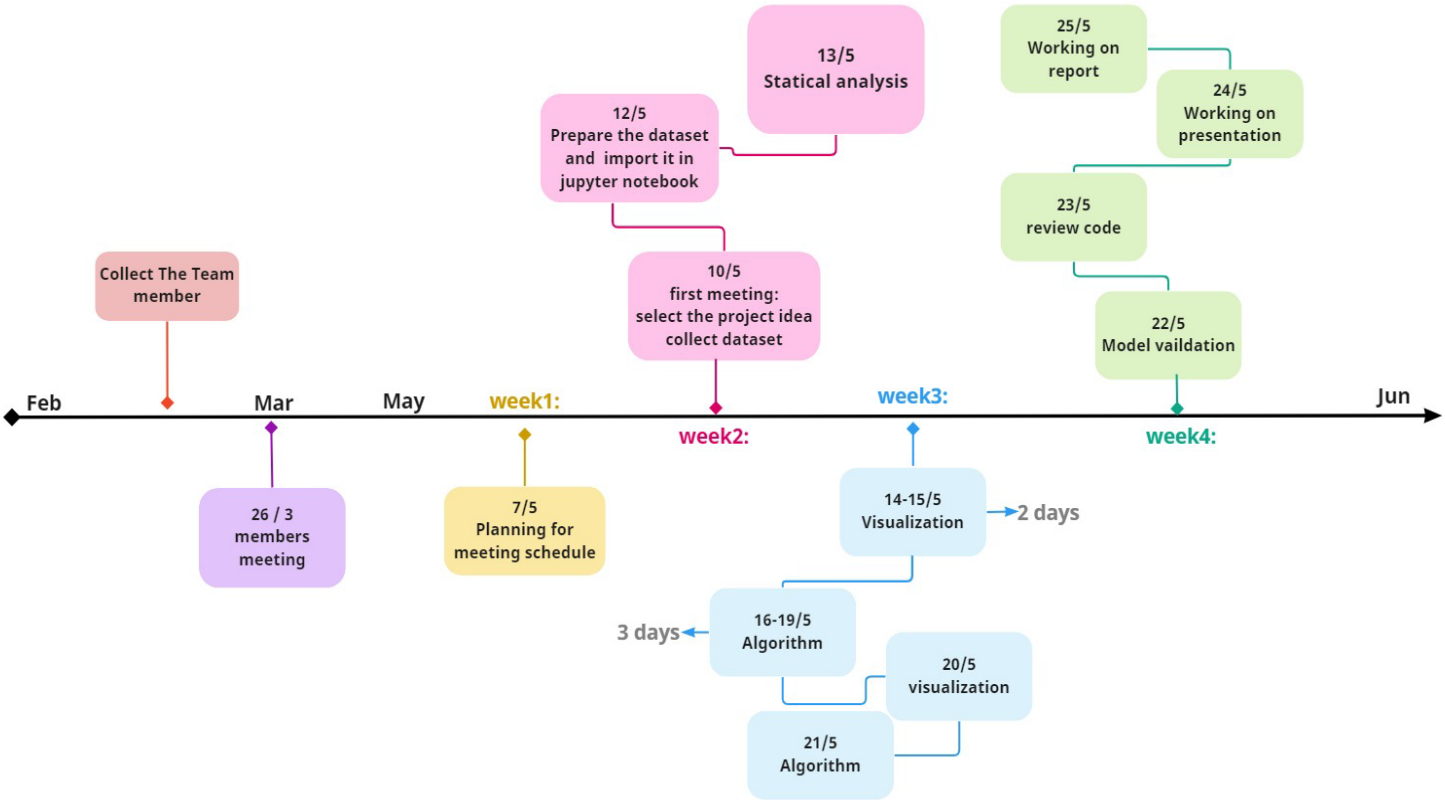
Linear Regression 21

Logistic Regression..... 21

Conclusion..... 22

Reference 22

Time Line:



Task	Deadline	Lead
Collect dataset	10/5/2022	all
Implementation code	12/5/2022	All, Fedhah's computer
Run test	23/5/2022	All, Fedhah's computer
Presentation	24/5/2022	Farha's computer and Fedhah
Report	25/5/2022	Balqis's computer , Layal's computer, Fedha, and Farha's coordinate
Upload code on GitHub	26/5/2022	Fedhah

Abstract

One of the problems facing the market in the retail sector is how to use the loyalty card for its customers to increase revenues and attract customers to subscribe. There is concern on the part of the retailer and the customer in developing advantages for the loyalty card for fear of reducing the company's revenues and losing customers, so by analyzing the sales data in the supermarket to study The experience of customer satisfaction in purchasing between a customer with a membership and a customer without a membership.

The customer satisfaction rate was used to devise a solution to encourage both parties to use loyalty cards. Through the use of the average rate and other algorithms by analyzing the data, the results showed that the average customer satisfaction with the service, in general, is 7 out of 10 in all categories and in particular in specific categories such as members obtaining a higher percentage of sales than normal, which from During which good results were obtained.

Introduction

Nowadays, there have been many changes in our lives, from development and changes in vision, and consequently the prices of all products have increased, taxes have risen, and consumers are looking for the best and safest ways to reduce expenses in order to be economical.

In terms of supermarkets, we found that subscribing to membership is a unique solution because it offers many advantages in terms of discounts and prices that suit everyone and helps them maintain their savings and get the best products at the best prices and offers. Therefore, it also has advantages that belong to the supermarket, as it raises revenues and increases the number of customers.

Method

The methods included in this project for analyzing sales data for "Supermarket sales "In phase preprocessing, we changed the data from the object to numerical data.
installed the libraries necessary to start the work procedures,

1. import pandas as pd : to show dataset structure make changes to basic data that is not arranged as raw data.
2. Import numpy as np: a fundamental package for scientific computing with Python mathematical library used in the mathematical operation.
3. import matplotlib.pyplot as plt: to interactive visualization in Python.
4. From sklearn.preprocessing import LabelEncoder
5. From sklearn.tree import DecisionTreeClassifier
6. From sklearn.model_selection import train_test_split
7. From .metrics import accuracy_score
8. From sklearn import datasets
9. import seaborn as sns: Python data visualization library based on matplotlib. It provides a high-level interface and informative statistical graphics.

GitHub Link: [DataMining/final Project at main · fedhah/DataMining (github.com)]

Screenshot for code

Data preprocessing

In [2]:
```
import matplotlib.pyplot as plt
import numpy as np
import os
import pandas as pd
```

In [3]:
```
!pip install opendatasets
```

```
Requirement already satisfied: opendatasets in /Users/fedhaalmutairi/opt/anaconda3/lib/python
3.9/site-packages (0.1.22)
Requirement already satisfied: kaggle in /Users/fedhaalmutairi/opt/anaconda3/lib/python3.9/si
te-packages (from opendatasets) (1.5.12)
Requirement already satisfied: tqdm in /Users/fedhaalmutairi/opt/anaconda3/lib/python3.9/site
-packages (from opendatasets) (4.62.3)
Requirement already satisfied: click in /Users/fedhaalmutairi/opt/anaconda3/lib/python3.9/sit
e-packages (from opendatasets) (8.0.3)
Requirement already satisfied: python-dateutil in /Users/fedhaalmutairi/opt/anaconda3/lib/pyt
hon3.9/site-packages (from kaggle->opendatasets) (2.8.2)
Requirement already satisfied: python-slugify in /Users/fedhaalmutairi/opt/anaconda3/lib/pyth
on3.9/site-packages (from kaggle->opendatasets) (5.0.2)
```

Import os: to interact with macOS easily.

```
In [4]: import opendatasets as od

In [5]: dataset = 'https://www.kaggle.com/datasets/aungpyaeap/supermarket-sales'

In [6]: od.download(dataset)
        Skipping, found downloaded files in "./supermarket-sales" (use force=True to force download)

In [7]: data_dir = './supermarket-sales'

In [8]: os.listdir(data_dir)
Out[8]: ['supermarket_sales - Sheet1.csv']

In [81]: supermarket = pd.read_csv("./supermarket-sales/supermarket_sales - Sheet1.csv")

In [82]: supermarket
Out[82]:
```

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Paymen
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	1/5/2019	13:08	Ewalle

Load the Dataset from Kaggle online link.

Change Dataset from object to numerical datatype by using MAP function.

```
In [9]: supermarket_df = pd.read_csv("./supermarket-sales/supermarket_sales - Sheet1.csv")

In [10]: supermarket_df
         con = {'A': 0, 'B': 1, 'C': 2}
         supermarket_df['Branch'] = supermarket_df['Branch'].map(con)
         con = {'Yangon': 1, 'Naypyitaw': 0, 'Mandalay':2}
         supermarket_df['City'] = supermarket_df['City'].map(con)
         con = {'Member': 1, 'Normal': 0}
         supermarket_df['Customer type'] = supermarket_df['Customer type'].map(con)
         con = {'Female': 0, 'Male': 1}
         supermarket_df['Gender'] = supermarket_df['Gender'].map(con)
         con = {'Health and beauty': 0, 'Electronic accessories': 1, 'Home and lifestyle': 2, 'Sports an
         supermarket_df['Product line'] = supermarket_df['Product line'].map(con)
         con = {'Ewallet': 0, 'Cash': 1, 'Credit card': 2}
         supermarket_df['Payment'] = supermarket_df['Payment'].map(con)

In [11]: supermarket_df
Out[11]:
```

Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gross income	Rating
1	0	0	74.69	7	26.1415	548.9715	1/5/2019	13:08	0	522.83	4.761905	26.1415	9.1

```
In [12]: supermarket_df['Time']= pd.to_datetime(supermarket_df['Time'],format="%H:%M").dt.hour
         supermarket_df
Out[12]:
```

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs
0	750-67-8428	0	1	1	0	0	74.69	7	26.1415	548.9715	1/5/2019	13	0	522.83
1	226-31-3081	2	0	0	0	1	15.28	5	3.8200	80.2200	3/8/2019	10	1	76.40
2	631-41-3108	0	1	0	1	2	46.33	7	16.2155	340.5255	3/3/2019	13	2	324.31
3	123-19-1176	0	1	1	1	0	58.22	8	23.2880	489.0480	1/27/2019	20	0	465.76
4	373-73-7910	0	1	0	1	3	86.31	7	30.2085	634.3785	2/8/2019	10	0	604.17
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	233-67-5758	2	0	0	1	0	40.35	1	2.0175	42.3675	1/29/2019	13	0	40.35

Format Time column to int.

Statistical analysis

```
In [13]: supermarket_df.isnull().sum()
```

```
Out[13]: Invoice ID                0
         Branch                    0
         City                      0
         Customer type             0
         Gender                    0
         Product line              0
         Unit price                0
         Quantity                  0
         Tax 5%                    0
         Total                     0
         Date                      0
         Time                      0
         Payment                   0
         cogs                      0
         gross margin percentage   0
         gross income              0
         Rating                    0
         dtype: int64
```

Check if there are any missing data.

The result no missing data.

```
In [14]: print(supermarket_df.describe())
```

```
              Branch       City  Customer type       Gender  Product line  \
count  1000.000000  1000.0000    1000.000000  1000.000000   1000.000000
mean      0.988000     1.0040       0.501000     0.499000      2.574000
std       0.817633     0.8128       0.500249     0.500249      1.707466
min       0.000000     0.0000       0.000000     0.000000      0.000000
25%       0.000000     0.0000       0.000000     0.000000      1.000000
50%       1.000000     1.0000       1.000000     0.000000      3.000000
75%       2.000000     2.0000       1.000000     1.000000      4.000000
max       2.000000     2.0000       1.000000     1.000000      5.000000

         Unit price      Quantity       Tax 5%        Total         Time  \
count  1000.000000  1000.000000  1000.000000  1000.000000  1000.000000
mean     55.672130     5.510000    15.379369   322.966749    14.910000
std      26.494628     2.923431    11.708825   245.885335     3.186857
min      10.080000     1.000000     0.508500    10.678500    10.000000
25%      32.875000     3.000000     5.924875   124.422375    12.000000
50%      55.230000     5.000000    12.088000   253.848000    15.000000
75%      77.935000     8.000000    22.445250   471.350250    18.000000
max      99.960000    10.000000    49.650000  1042.650000    20.000000

             Payment        cogs  gross margin percentage  gross income  \
count  1000.000000  1000.00000             1.000000e+03   1000.000000
mean      0.966000   307.58738             4.761905e+00     15.379369
std       0.809629   234.17651             6.131498e-14     11.708825
min       0.000000    10.17000             4.761905e+00      0.508500
25%       0.000000   118.49750             4.761905e+00      5.924875
```

Use describe function to show the main statistical result.

```
min       0.000000     0.0000       0.000000     0.000000      0.000000
25%       0.000000     0.0000       0.000000     0.000000      1.000000
50%       1.000000     1.0000       1.000000     0.000000      3.000000
75%       2.000000     2.0000       1.000000     1.000000      4.000000
max       2.000000     2.0000       1.000000     1.000000      5.000000

         Unit price      Quantity       Tax 5%        Total         Time  \
count  1000.000000  1000.000000  1000.000000  1000.000000  1000.000000
mean     55.672130     5.510000    15.379369   322.966749    14.910000
std      26.494628     2.923431    11.708825   245.885335     3.186857
min      10.080000     1.000000     0.508500    10.678500    10.000000
25%      32.875000     3.000000     5.924875   124.422375    12.000000
50%      55.230000     5.000000    12.088000   253.848000    15.000000
75%      77.935000     8.000000    22.445250   471.350250    18.000000
max      99.960000    10.000000    49.650000  1042.650000    20.000000

             Payment        cogs  gross margin percentage  gross income  \
count  1000.000000  1000.00000             1.000000e+03   1000.000000
mean      0.966000   307.58738             4.761905e+00     15.379369
std       0.809629   234.17651             6.131498e-14     11.708825
min       0.000000    10.17000             4.761905e+00      0.508500
25%       0.000000   118.49750             4.761905e+00      5.924875
50%       1.000000   241.76000             4.761905e+00     12.088000
75%       2.000000   448.90500             4.761905e+00     22.445250
max       2.000000   993.00000             4.761905e+00     49.650000

            Rating
count  1000.00000
mean      6.97270
std       1.71858
min       4.00000
25%       5.50000
50%       7.00000
75%       8.50000
max      10.00000
```

```
In [15]: print('Display the Total for several columns:')
         print('')
         # the total sales in each branch
         b = supermarket_df.groupby(by=['Branch']).sum()[['Total']].sort_values(by='Total')
         print('The total for branch:')
         print(b)
         #the total sales for each gender
         g = supermarket_df.groupby(['Gender']).sum()[['Total']].sort_values(by='Total')
         print('')
         print('The total for gender:')
         print(g)
         # sum total for each product line from low to high
         p = supermarket_df.groupby(by=['Product line']).sum()[['Total']].sort_values(by='Total')
         print('')
         print('The total for product line:')
         print(p)
         #the total in each hour
         h = supermarket_df.groupby(by=['Time']).sum()[['Total']].sort_values(by='Total')
         print('')
         print('The total in each hour:')
         print(h)
```

output

```
Display the Total for several columns:

The total for branch:
              Total
Branch
1       106197.6720
0       106200.3705
2       110568.7065

The total for gender:
             Total
Gender
1       155083.824
0       167882.925

The total for product line:
                   Total
Product line
0             49193.7390
2             53861.9130
5             54305.8950
1             54337.5315
3             55122.8265
4             56144.8440

The total in each hour:
           Total
Time
20     22969.5270
17     24445.2180
16     25226.3235
18     26030.3400
12     26065.8825
11     30377.3295
14     30828.3990
15     31179.5085
10     31421.4810
13     34723.2270
19     39699.5130
```

In [16]:
```
print((supermarket_df[['Tax 5%','Quantity']]).max())
```

```
Tax 5%       49.65
Quantity     10.00
dtype: float64
```

In [17]:
```
print('Shows  the total sales and its average')
print('')
total_sales = supermarket_df['Total'].sum()
avg_sales= round(supermarket_df['Total'].mean(),1)
print('Total Sales:',total_sales,'USD')
print('Average sales:',avg_sales)
```

```
Shows  the total sales and its average

Total Sales: 322966.749 USD
Average sales: 323.0
```

In [18]:
```
print('Shows the average of customer satisfaction rate: ')
Avg_rate= round(supermarket_df['Rating'].mean(),1)
print('Average Rating:',Avg_rate)
```

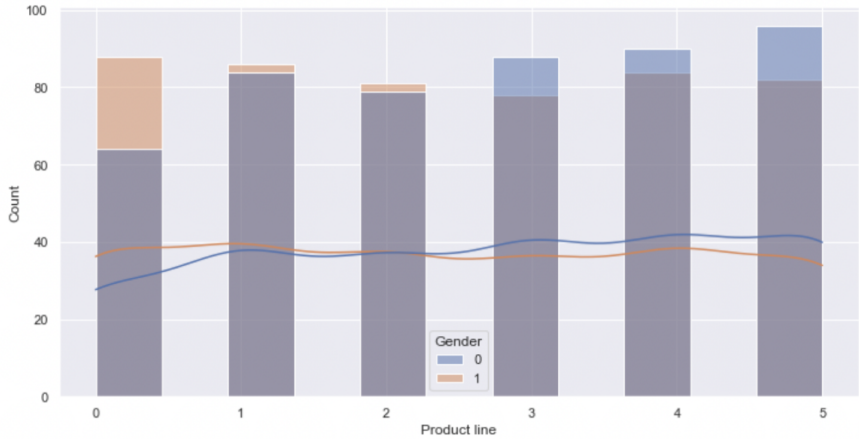
```
Shows the average of customer satisfaction rate:
Average Rating: 7.0
```

Data Visualisation ¶

In [19]:
```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(color_codes=True)
```

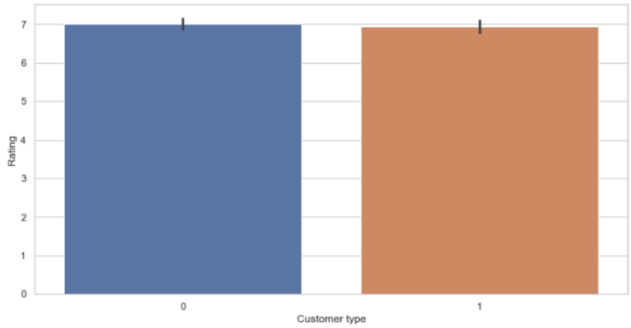
In [20]:
```
#represent which gender are more buy in each product line.
plt.figure(figsize = (12,6))
sns.histplot(x='Product line', hue='Gender', data=supermarket_df , kde=True )
```

Out[20]: <AxesSubplot:xlabel='Product line', ylabel='Count'>



In [124]:
```
plt.figure(figsize = (12,6))
sns.barplot(y='Rating', x='Customer type', data=supermarket_df  )
```

Out[124]: <AxesSubplot:xlabel='Customer type', ylabel='Rating'>

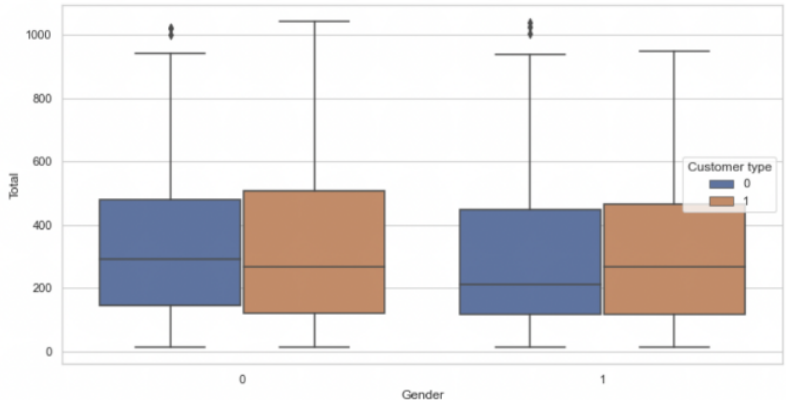


1: refers to member

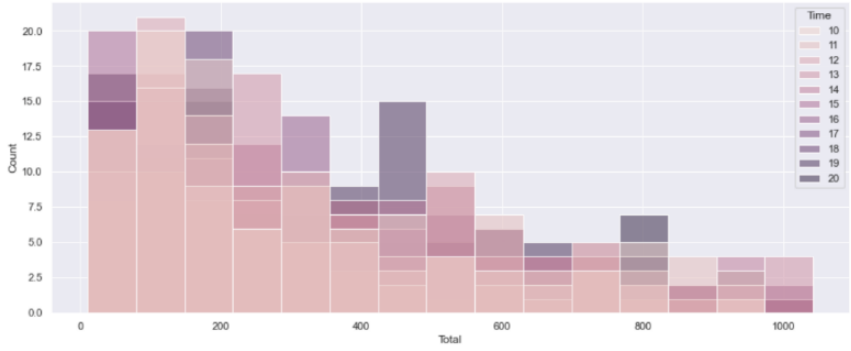
0: refers to normal

Shows the customer satisfying .

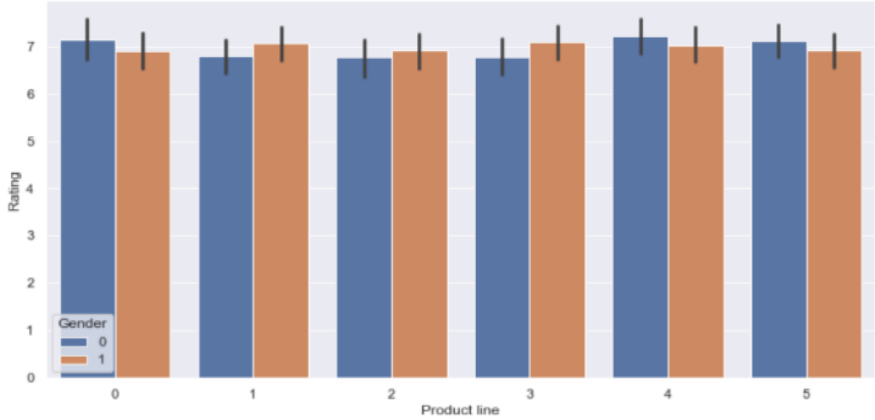
In [144]:
```
# represent the total in each branch
plt.figure(figsize = (12,6))
sns.boxplot(x='Gender',y='Total',hue='Customer type',data=supermarket_df)
plt.show()
```



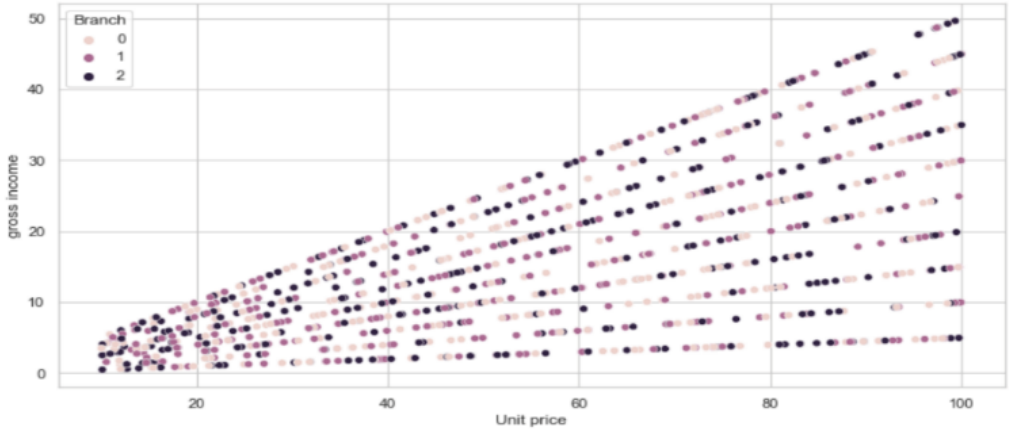
```
In [22]: # shows when customer more buy in each hour
         plt.figure(figsize = (15,6))
         sns.histplot(x='Total', hue='Time', data=supermarket_df )
Out[22]: <AxesSubplot:xlabel='Total', ylabel='Count'>
```



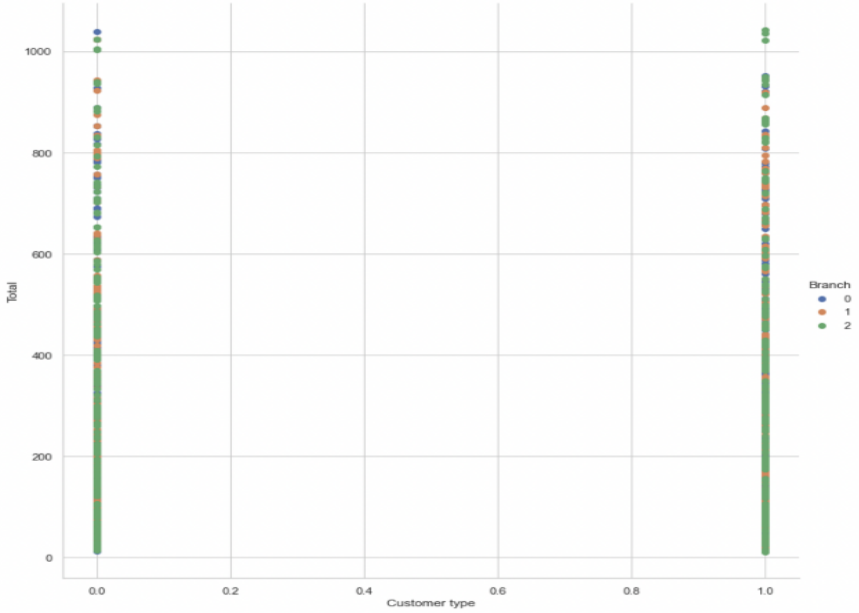

```
In [23]: #represents the highest rate that gets to the product line by each gender
         plt.figure(figsize = (12,6))

         sns.set(style='darkgrid')
         sns.barplot(data=supermarket_df,  y='Rating',x='Product line',hue='Gender')

         plt.show()
```



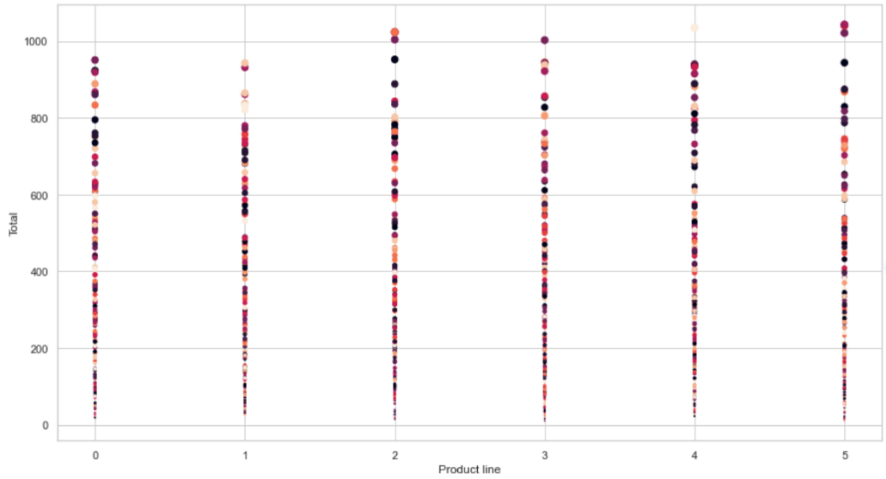

```
In [24]: # shows the gross income for each branch
         sns.set_style('whitegrid')
         plt.figure(figsize = (12,6))
         sns.scatterplot(x ='Unit price' , y = 'gross income',hue='Branch' , data = supermarket_df )
         plt.show()
```



In [132]:
```
# shows the most way for pay that used in each branch
sns.FacetGrid(supermarket_df,hue='Branch' , height=10).map(plt.scatter,'Customer type','Total')\
.add_legend();
plt.show();
```



In [26]:
```
plt.figure(figsize = (15,8))
plt.scatter(supermarket_df['Product line'], supermarket_df['Total'], c=supermarket_df['Time'],s=supermarket_df['gross i

plt.xlabel('Product line')
plt.ylabel('Total')

plt.show()
```



shows the most total pay to the product in a different product lines

Algorithms section

Decision Tree Algorithm

In [61]:
```
pip install pydotplus
```

```
Requirement already satisfied: pydotplus in /Users/fedhaalmutairi/opt/anaconda3/lib/python3.9/site-packages (2.0.2)
Requirement already satisfied: pyparsing>=2.0.1 in /Users/fedhaalmutairi/opt/anaconda3/lib/python3.9/site-packages (f
rom pydotplus) (3.0.4)
Note: you may need to restart the kernel to use updated packages.
```

In [62]:
```
import pydotplus as pydotplus
from sklearn.tree import DecisionTreeClassifier
%matplotlib inline
from sklearn import tree
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn import metrics
seed =10
```

In [63]:
```
supermarket_df_n= supermarket_df.drop(['Invoice ID','Date'],axis=1)
supermarket_df_n
```

Out[63]:

	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Time	Payment	cogs	gross margin percentage	gross income	Rating	cluster
0	0	1	1	0	0	74.69	7	26.1415	548.9715	13	0	522.83	4.761905	26.1415	9.1	1
1	2	0	0	0	1	15.28	5	3.8200	80.2200	10	1	76.40	4.761905	3.8200	9.6	4
2	0	1	0	1	2	46.33	7	16.2155	340.5255	13	2	324.31	4.761905	16.2155	7.4	2
3	0	1	1	1	0	58.22	8	23.2880	489.0480	20	0	465.76	4.761905	23.2880	8.4	2
4	0	1	0	1	3	86.31	7	30.2085	634.3785	10	0	604.17	4.761905	30.2085	5.3	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	2	0	0	1	0	40.35	1	2.0175	42.3675	13	0	40.35	4.761905	2.0175	6.2	4
996	1	2	0	0	2	97.38	10	48.6900	1022.4900	17	0	973.80	4.761905	48.6900	4.4	3
997	0	1	1	1	4	31.84	1	1.5920	33.4320	13	1	31.84	4.761905	1.5920	7.7	4
998	0	1	0	1	2	65.82	1	3.2910	69.1110	15	1	65.82	4.761905	3.2910	4.1	4
999	0	1	1	0	5	88.34	7	30.9190	649.2990	13	1	618.38	4.761905	30.9190	6.6	1

In [64]:
```
features = ['Product line','Total','Rating','gross margin percentage' ,'Branch']

X = supermarket_df_n[features]
y = supermarket_df_n['Customer type']
print(X)
print(y)
```

```
      Product line      Total  Rating  gross margin percentage  Branch
0                0   548.9715     9.1                 4.761905       0
1                1    80.2200     9.6                 4.761905       2
2                2   340.5255     7.4                 4.761905       0
3                0   489.0480     8.4                 4.761905       0
4                3   634.3785     5.3                 4.761905       0
..             ...        ...     ...                      ...     ...
995              0    42.3675     6.2                 4.761905       2
996              2  1022.4900     4.4                 4.761905       1
997              4    33.4320     7.7                 4.761905       0
998              2    69.1110     4.1                 4.761905       0
999              5   649.2990     6.6                 4.761905       0

[1000 rows x 5 columns]
0      1
1      0
2      0
3      1
4      0
      ..
995    0
996    0
997    1
998    0
999    1
Name: Customer type, Length: 1000, dtype: int64
```

In [65]:
```
m=supermarket_df_n[ 'Customer type']
m.shape
```

Out[65]: (1000,)

In [66]:
```
supermarket_df_n['Customer type']
```

Out[66]:
```
0      1
1      0
2      0
```

In [67]:
```
df_test=supermarket_df_n.sample(n=10)
x_test=df_test.drop(['Customer type'],axis=1)
y_test=df_test['Customer type']
```

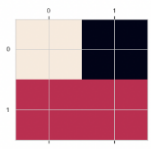
In [68]:
```
le = LabelEncoder()
le.fit(supermarket_df_n['Customer type'].values)
y = le.transform(supermarket_df_n['Customer type'].values)
X = supermarket_df_n.drop('Customer type', axis=1).values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    stratify=y, random_state=seed)
```

In [69]:
```
tree=DecisionTreeClassifier(criterion='entropy',min_samples_leaf=2,min_samples_split=2,max_depth=None,random_state=seed
tree.fit(X_train,y_train)
y_pred=tree.predict(X_test)
accuracy=accuracy_score(y_test, y_pred)
print('DecisionTreeClassifier accuracy score {}'.format(accuracy))
```

```
DecisionTreeClassifier accuracy score 0.545
```

In [70]:
```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt

print('Confusion Matrix is')
print(confusion_matrix(y_test, y_pred))
cm=confusion_matrix(y_test, y_pred)
plt.matshow(cm)
plt.show()
```

```
Confusion Matrix is
[[59 41]
 [50 50]]
```



Using a decision tree to predict the potential value of membership

In [71]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred, labels=supermarket_df_n['Customer type'].unique()))
```

```
              precision    recall  f1-score   support

           1       0.55      0.50      0.52       100
           0       0.54      0.59      0.56       100

    accuracy                           0.55       200
   macro avg       0.55      0.54      0.54       200
weighted avg       0.55      0.55      0.54       200
```

In [72]:
```
accuracy= metrics.accuracy_score(y_test,y_pred)
print("Accuracy of this model = {:.3f}".format(accuracy))
```

```
Accuracy of this model = 0.545
```

KMeans Algorithm

In [34]:
```
from sklearn.cluster import KMeans
```

In [35]:
```
x = supermarket_df.iloc[:, [1,12,13,14,15,16]].values
supermarket_df
```

Out[35]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gross income	Rating
0	750-67-8428	0	1	1	0	0	74.69	7	26.1415	548.9715	1/5/2019	13	0	522.83	4.761905	26.1415	9.1
1	226-31-3081	2	0	0	0	1	15.28	5	3.8200	80.2200	3/8/2019	10	1	76.40	4.761905	3.8200	9.6
2	631-41-3108	0	1	0	1	2	46.33	7	16.2155	340.5255	3/3/2019	13	2	324.31	4.761905	16.2155	7.4
3	123-19-1176	0	1	1	1	0	58.22	8	23.2880	489.0480	1/27/2019	20	0	465.76	4.761905	23.2880	8.4
4	373-73-7910	0	1	0	1	3	86.31	7	30.2085	634.3785	2/8/2019	10	0	604.17	4.761905	30.2085	5.3

In [36]:
```
Kmeans5 = KMeans(n_clusters = 5)
y_kmeans5= Kmeans5.fit_predict(x)
print(y_kmeans5)
Kmeans5.cluster_centers_
```

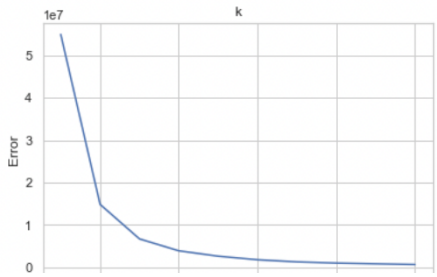
```
[1 4 2 2 1 1 2 3 4 0 4 4 0 2 3 1 1 2 0 4 2 0 4 0 0 2 4 0 2 0 1 3 1 0 2 4 2
 1 2 0 4 4 2 3 4 2 1 0 4 3 1 0 4 4 4 3 4 3 3 0 4 2 1 0 2 0 4 1 3 4 3 1 4 2
 1 2 2 0 3 4 1 0 0 2 0 1 2 2 4 2 0 2 0 1 4 4 3 4 0 1 4 3 0 2 0 3 0 2 2 3 4
 0 1 1 3 4 4 4 4 4 3 2 3 1 2 1 0 0 1 3 0 1 4 1 1 0 4 0 1 1 3 3 3 4 2 0 2 0
 1 0 2 0 3 4 1 2 4 0 3 1 2 4 2 0 2 2 3 3 2 1 2 2 0 2 0 2 0 0 2 0 2 2 0 0 4
 4 3 0 4 0 0 4 3 4 0 4 4 4 0 2 4 0 2 0 4 0 1 0 4 3 0 3 2 4 0 4 4 0 3 0 0 2
 4 4 4 0 1 4 1 1 0 4 3 0 1 0 4 0 4 4 0 4 0 0 3 2 0 4 2 1 2 2 4 4 1 4 4 0 4
 0 1 4 0 0 4 3 4 3 0 2 2 4 0 4 1 2 0 2 3 2 1 4 4 1 0 2 4 0 2 1 2 0 4 4 0 0
 4 1 0 4 0 4 4 0 4 1 0 3 4 0 1 4 4 4 3 0 4 2 0 0 0 4 4 2 4 2 1 2 0 4 0 4 2
 4 4 4 3 2 4 2 4 2 1 1 4 3 4 3 0 4 3 2 2 0 4 2 2 3 4 4 1 3 0 2 0 2 2 0 4 0
 0 4 2 0 2 1 2 1 2 4 2 4 1 2 0 4 0 2 1 4 0 2 4 1 4 2 0 4 0 0 0 0 4 0 3 0 4
 0 0 4 2 4 4 4 2 4 0 4 4 0 4 0 3 0 4 2 2 4 2 3 4 4 4 1 4 3 2 0 4 2 4 3 3 4
 4 4 0 4 4 4 2 4 2 4 4 4 3 3 2 4 0 3 0 4 0 1 0 0 4 4 1 0 2 1 2 0 0 2 2 4 1
 2 0 1 2 0 2 4 4 3 4 0 0 4 0 1 4 1 4 2 4 4 4 0 4 0 4 1 2 4 1 0 2 3 2 4 0 0
 4 2 4 1 0 4 1 2 0 4 1 3 0 3 0 4 4 4 4 4 0 1 4 4 4 4 4 0 0 1 0 2 0 1 1 4 4
 4 4 3 0 0 0 3 2 2 0 3 1 1 1 1 2 0 4 4 2 1 0 4 0 4 4 2 0 4 0 0 0 0 1 4 0 0
 4 0 0 4 0 2 0 4 4 4 3 2 2 4 2 0 4 4 4 3 0 4 1 2 2 2 1 0 0 4 1 3 4 0 4 3 0
 4 1 0 0 2 0 1 0 4 4 0 0 3 4 1 4 0 2 4 4 0 2 2 3 2 0 4 4 0 4 4 4 4 2 3 0 0
 0 4 0 4 1 0 0 0 2 0 2 0 1 0 4 4 0 4 4 4 4 1 4 0 1 2 1 1 1 2 4 0 1 3 1 4 0
 3 3 1 0 4 4 4 1 2 3 4 3 0 1 4 4 4 0 3 4 0 4 0 0 3 2 2 4 0 4 1 4 3 1 1 3 3
 2 4 4 4 0 4 1 4 4 1 4 4 0 2 0 3 0 1 0 0 3 3 1 4 0 3 0 4 2 4 2 1 1 4 4 4 2
 4 4 3 0 2 4 2 4 1 1 4 0 2 4 0 3 1 4 4 4 0 4 2 4 0 1 2 3 2 0 0 4 1 4 2 2 4
 2 2 2 0 2 0 0 4 4 0 2 4 1 4 1 3 4 0 4 4 0 4 0 0 1 0 0 4 4 4 4 0 4 4 1 1 4
 0 2 1 1 2 0 4 2 2 4 4 4 2 0 4 3 4 4 0 0 4 0 1 4 1 4 4 3 4 2 4 2 0 0 2 4 3
 4 3 2 2 2 4 2 3 2 0 1 2 2 0 4 4 4 2 3 2 0 1 4 0 3 2 2 0 4 2 0 4 2 4 4 1 0
 4 4 2 3 4 1 0 3 2 2 0 1 2 0 2 0 3 2 0 4 1 0 4 0 0 4 4 0 4 2 2 2 0 0 3 4 4
 4 0 4 4 2 0 4 0 3 0 1 0 0 4 2 0 4 4 0 0 3 3 1 2 4 1 3 1 0 3 4 0 4 4 3 4 4
 1]
```

Out[36]:
```
array([[  0.91954023,   0.95402299, 216.75003831,   4.76190476,
         10.83750192,   7.09042146],
       [  1.        ,   0.96183206, 561.89618321,   4.76190476,
         28.09480916,   6.89007634],
       [  0.9673913 ,   0.92391304, 378.38809783,   4.76190476,
         18.91940489,   7.01521739],
       [  1.09803922,   1.01960784, 795.89784314,   4.76190476,
         39.79489216,   6.80784314],
       [  1.01552795,   0.98447205,  82.61562112,   4.76190476,
          4.13078106,   6.93881988]])
```

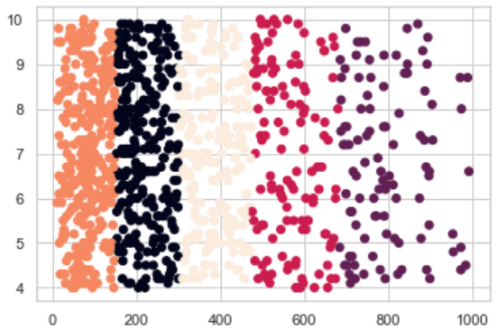
```
supermarket_df['cluster']= y_kmeans5
supermarket_df.head(25)
```

Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gross income	Rating	cluster
750-67-8428	0	1	1	0	0	74.69	7	26.1415	548.9715	1/5/2019	13	0	522.83	4.761905	26.1415	9.1	2
226-31-3081	2	0	0	0	1	15.28	5	3.8200	80.2200	3/8/2019	10	1	76.40	4.761905	3.8200	9.6	3
631-41-3108	0	1	0	1	2	46.33	7	16.2155	340.5255	3/3/2019	13	2	324.31	4.761905	16.2155	7.4	4
123-19-1176	0	1	1	1	0	58.22	8	23.2880	489.0480	1/27/2019	20	0	465.76	4.761905	23.2880	8.4	4
373-73-7910	0	1	0	1	3	86.31	7	30.2085	634.3785	2/8/2019	10	0	604.17	4.761905	30.2085	5.3	2
699-14-3026	2	0	0	1	1	85.39	7	29.8865	627.6165	3/25/2019	18	0	597.73	4.761905	29.8865	4.1	2
355-53-5943	0	1	1	0	1	68.84	6	20.6520	433.6920	2/25/2019	14	0	413.04	4.761905	20.6520	5.8	4
315-22-5665	2	0	0	0	2	73.56	10	36.7800	772.3800	2/24/2019	11	0	735.60	4.761905	36.7800	8.0	1
665-32-9167	0	1	1	0	0	36.26	2	3.6260	76.1460	1/10/2019	17	2	72.52	4.761905	3.6260	7.2	3
692-92-5582	1	2	1	0	4	54.84	3	8.2260	172.7460	2/20/2019	13	2	164.52	4.761905	8.2260	5.9	0

In [38]:
```
Error = []
for i in range(1,11):
    kmeans = KMeans(n_clusters = i).fit(x)
    kmeans.fit(x)
    Error.append(kmeans.inertia_)

plt.plot(range(1,11),Error)
plt.title('k')
plt.xlabel('No of clusters')
plt.ylabel('Error')
plt.show()
```



In [222]:
```
plt.scatter(x[:,2],x[:,5],c=y_kmeans5)
plt.show()
```



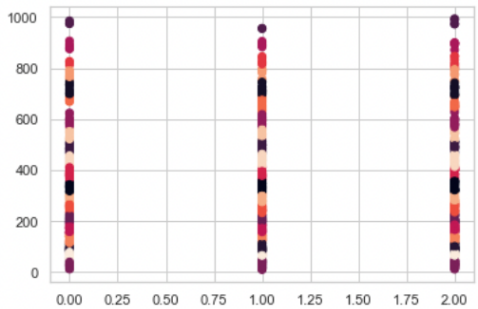
Cluster the data and to sit based on their similarity

```
In [40]: Kmeans20 = KMeans(n_clusters = 20)
         y_kmeans20= Kmeans20.fit_predict(x)
         print(y_kmeans20)
         Kmeans5.cluster_centers_
```

```
[17 19  0  3  7  7 18  1 19  9 19  2 12 18  1 17  3 18  9 19 18 12 19  9
 12 18  6  9 18  5  3  1 13  5 10 19  0 17 18 12  2  2 18 15  6 10  3  9
  2 11 13  5 19  6  6 13  2  1  1  9 19  0  3  9 16  9 19  7 15  6  8  7
 14 18 13 18 18  9 15  2  7 16  9  0 12  7 10  0  6 18  5 18  9  7  2  2
  8 19  9  3  6  1  5  0  5  8  9 10 16 11  6 16  3 17  1 19  2 19  2  2
 15 10  8  3 18 13  9 16 17 11 16 17 14 17  3 16 14  5 17  7  8  8 13  6
  3 16 16 12  7 12 10 12 15  2 13 18 19 12  8 17  0 19  0  9 10  0  4  4
 16  3  0 10  9  0  9 16  9  9 18 12 18 16  9 16  2 19 15  9 19 16 12 14
 15  2  9 19  2  6  9  0 14 12 18 12  2  5  7 16  2  8 12 11 18 14  5  6
 14 16 15  9 16 18 19  6  2  9  7 14 17  3  9  6 15 16 17  9  2  5 19  2
  5 14  9 12 15 10 12  2 16  3 18  0  6  2  3 14 19  9 14 12 13 19  9  5
 19 13 14  1 16  0  0  6  5  6  7  0  5 10  1 18 17  6  6  7  5  0 19  5
  0 13 10  5 19 19  9 12 14  3 12  2  5  6  6  5 14  3 16  1 19  9  3 14
 19 14 15  5 19 10  5  9  5  6  2 16 14 10  3 10  5 14  5  2 10  6 19 14
 13  0 14 10 14 10 17  3 14  1 19  1  9 14  4 10 10 12  6  0 10  4  2 19
```

```
Out[40]: array([[  0.91954023,   0.95402299, 216.75003831,   4.76190476,
          10.83750192,   7.09042146],
        [  1.        ,   0.96183206, 561.89618321,   4.76190476,
          28.09480916,   6.89007634],
        [  0.9673913 ,   0.92391304, 378.38809783,   4.76190476,
          18.91940489,   7.01521739],
        [  1.09803922,   1.01960784, 795.89784314,   4.76190476,
          39.79489216,   6.80784314],
        [  1.01552795,   0.98447205,  82.61562112,   4.76190476,
           4.13078106,   6.93881988]])
```

```
In [41]: plt.scatter(x[:,1],x[:,2],c=y_kmeans20)
         plt.show()
```



Naive Bayes

In [121]:
```
from sklearn.naive_bayes import GaussianNB, BernoulliNB
from sklearn.preprocessing import LabelEncoder
```

In [122]:
```
dataframe= [('Mandalay','Female','Home and lifestyle','Normal'),
            ('Naypyitaw', ' Female ','Electronic accessories','Normal'),
            ('Mandalay','Male','Sports and travel','Member'),
            ('Mandalay','Male','Sports and travel','Normal'),
            ('Naypyitaw', ' Female ','Home and lifestyle','Normal'),
            ('Yangon','Female','Electronic and accessories' ,'Member'),
            ('Mandalay','Female','Food and beverages','Normal'),
           ('Mandalay','Female','Sports and travel','Member'),
           ('Yangon','Female','Health and beauty' ,'Member'),
            ('Naypyitaw', 'Male ','Electronic accessories','Normal'),
           ('Yangon','Male','Home and lifestyle' ,'Normal'),
            ('Yangon','Male','Health and beauty' ,'Member'),
            ('Yangon','Male','Sports and travel' ,'Normal'),
          ('Naypyitaw', 'Male','Electronic accessories','Member')]

superMarketdf=pd.DataFrame(dataframe, columns=['city','Gender', 'Product line','Customer type'])
superMarketdf
```

Out[122]:

	city	Gender	Product line	Customer type
0	Mandalay	Female	Home and lifestyle	Normal
1	Naypyitaw	Female	Electronic accessories	Normal
2	Mandalay	Male	Sports and travel	Member
3	Mandalay	Male	Sports and travel	Normal
4	Naypyitaw	Female	Home and lifestyle	Normal
5	Yangon	Female	Electronic and accessories	Member
6	Mandalay	Female	Food and beverages	Normal
7	Mandalay	Female	Sports and travel	Member

In [45]:
```
data={'City':['Mandalay','Naypyitaw','Yangon','Naypyitaw','Mandalay'],
      'ProductLine':['Home and lifestyle','Electronic accessories','Electronic and accessories','Home and lifestyle',
      'Customer':['Normal','Normal','Member','Normal','Member']}

data=pd.DataFrame(data)
data
```

Out[45]:

	City	ProductLine	Customer
0	Mandalay	Home and lifestyle	Normal
1	Naypyitaw	Electronic accessories	Normal
2	Yangon	Electronic and accessories	Member
3	Naypyitaw	Home and lifestyle	Normal
4	Mandalay	Sports and travel	Member

In [124]:
```
le = LabelEncoder()
Gender_encoded=le.fit_transform(superMarketdf['Gender'])
Custypy_encoded=le.fit_transform(superMarketdf['Customer type'])
City_encoded=le.fit_transform(data['City'])
print("City encoded:",City_encoded)
print("Gender encoded:",Gender_encoded)
print("Customer type encoded:",Custypy_encoded)
```

```
City encoded: [0 1 2 1 0]
Gender encoded: [1 0 2 2 0 1 1 1 1 3 2 2 2 2]
Customer type encoded: [1 1 0 1 1 0 1 0 0 1 1 0 1 0]
```

In [125]:
```
city_encoded=le.fit_transform(superMarketdf['city'])
print("city encoded:",city_encoded)
```

```
city encoded: [0 1 0 0 1 2 0 0 2 1 2 2 2 1]
```

In [126]:
```
features=zip(city_encoded,Gender_encoded)
features_Is=list(features)
print(features_Is)
```

```
[(0, 1), (1, 0), (0, 2), (0, 2), (1, 0), (2, 1), (0, 1), (0, 1), (2, 1),
(1, 3), (2, 2), (2, 2), (2, 2), (1, 2)]
```

In [127]:
```
City_encoded=le.fit_transform(data['City'])
product_encoded=le.fit_transform(data['ProductLine'])
cust_encoded=le.fit_transform(data['Customer'])

print("city encoded:",city_encoded)
print("product line encoded:",product_encoded)
print("customer type encoded:",cust_encoded)
```

```
city encoded: [0 1 0 0 1 2 0 0 2 1 2 2 2 1]
product line encoded: [2 0 1 2 3]
customer type encoded: [1 1 0 1 0]
```

PCA

In [12

In [179]:
```
from sklearn.preprocessing import StandardScaler , MinMaxScaler
feature = ['Branch','Customer type','Total','Payment']
a = supermarket_df.loc[:,feature].values
s = supermarket_df.loc[:,'Product line'].values
a = StandardScaler().fit_transform(a)
print(a)
```

In [12

```
[[-1.20897001  0.998002    0.91960685 -1.19373575]
 [ 1.2383377  -1.002002   -0.98772956  0.04201554]
 [-1.20897001 -1.002002    0.07144605  1.27776684]
 ...
 [-1.20897001  0.998002   -1.17810859  0.04201554]
 [-1.20897001 -1.002002   -1.03293176  0.04201554]
 [-1.20897001  0.998002    1.32783657  0.04201554]]
```

In [13

In [186]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components =2)
principalComponents = pca.fit_transform(a)
pricipalDF= pd.DataFrame(data = principlaComponents, columns=['PC1','PC2']
```

```
In [188]: pricipalDF
```

Out[188]:

	PC1	PC2
0	-0.257853	0.074790
1	-0.420903	0.391724
2	-0.342486	-0.803170
3	-0.347511	-0.078420
4	-1.326835	1.003739
...	...	...
995	-1.189543	0.822741
996	-0.234612	2.662370
997	-0.317198	-1.771123
998	-1.460583	-0.969317
999	0.604263	-0.196493

1000 rows × 2 columns

```
In [189]: finalDF =pd.concat([pricipalDF, supermarket_df[['Product line']]],axis=1)
In [190]: finalDF
```

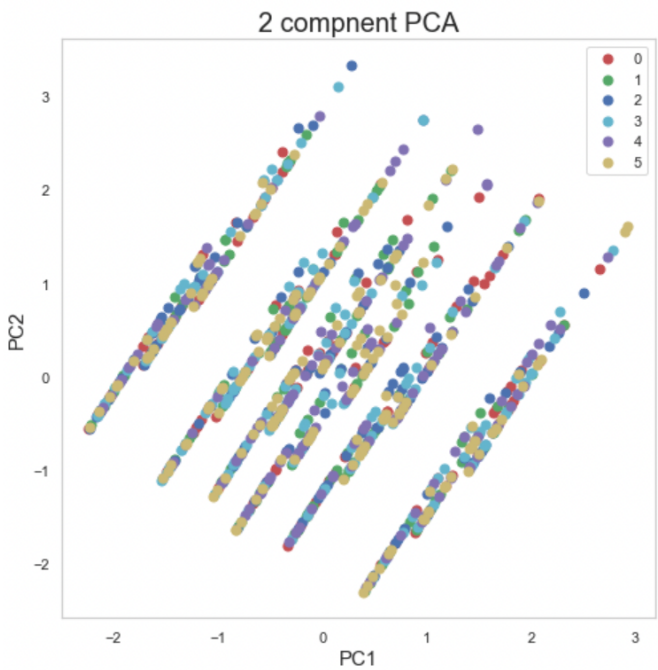
Out[190]:

	PC1	PC2	Product line
0	-0.257853	0.074790	0
1	-0.420903	0.391724	1
2	-0.342486	-0.803170	2
3	-0.347511	-0.078420	0
4	-1.326835	1.003739	3
...	...	...	...
995	-1.189543	0.822741	0
996	-0.234612	2.662370	2
997	-0.317198	-1.771123	4
998	-1.460583	-0.969317	2
999	0.604263	-0.196493	5

1000 rows × 3 columns

Using PCA to summarize the large data based on product line to make it easily visualized

```
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('PC1',fontsize =15)
ax.set_ylabel('PC2',fontsize =15)
ax.set_title('2 compnent PCA',fontsize =20)

targets = [0,1,2,3,4,5]
colors = ['r','g','b','c','m','y']

for target, color in zip(targets,colors):
    indicesToKeep = finalDF['Product line'] == target
    ax.scatter(finalDF.loc[indicesToKeep, 'PC1']
               , finalDF.loc[indicesToKeep, 'PC2']
               , c =color
               , s=50)

ax.legend(targets)
ax.grid()
```



Apriori Algorithm

```
supermarket_df.Gender.unique()
```

```
array([0, 1])
```

```
supermarket_dff =  pd.read_csv("./supermarket-sales/supermarket_sales - Sheet1.csv")
df = supermarket_dff.iloc[:, [1,3,5,4,12]].values
```

```
df
```

```
array([['A', 'Member', 'Health and beauty', 'Female', 'Ewallet'],
       ['C', 'Normal', 'Electronic accessories', 'Female', 'Cash'],
       ['A', 'Normal', 'Home and lifestyle', 'Male', 'Credit card'],
       ...,
       ['A', 'Member', 'Food and beverages', 'Male', 'Cash'],
       ['A', 'Normal', 'Home and lifestyle', 'Male', 'Cash'],
       ['A', 'Member', 'Fashion accessories', 'Female', 'Cash']],
      dtype=object)
```

```
from mlxtend.preprocessing import TransactionEncoder
TranEncod=TransactionEncoder()
te_ary=TranEncod.fit(df).transform(df)
dff=pd.DataFrame(te_ary, columns=TranEncod.columns_)
```

```
dff
```

	A	B	C	Cash	Credit card	Electronic accessories	Ewallet	Fashion accessories	Female	Food and beverages	Health and beauty	Home and lifestyle	Male	Member	Normal	Sports and travel
0	True	False	False	False	False	False	True	False	True	False	True	False	False	True	False	False
1	False	False	True	True	False	True	False	False	True	False	False	False	False	False	True	False
2	True	False	False	False	True	False	False	False	False	False	False	True	True	False	True	False

```
from mlxtend.frequent_patterns import apriori
apriori(dff, min_support=0.2,use_colnames=True)
```

	support	itemsets
0	0.340	(A)
1	0.332	(B)
2	0.328	(C)
3	0.344	(Cash)
4	0.311	(Credit card)
5	0.345	(Ewallet)
6	0.501	(Female)
7	0.499	(Male)
8	0.501	(Member)
9	0.499	(Normal)
10	0.261	(Member, Female)
11	0.240	(Normal, Female)
12	0.240	(Male, Member)
13	0.259	(Male, Normal)

```
In [75]: from sklearn.linear_model import LinearRegression
classifier = LinearRegression()
model = classifier.fit(Slic_df[['Total']],Slic_df[['Quantity']])
```

```
frequent_itemsets=apriori(dff, min_support=0.1, use_colnames=True)
frequent_itemsets['length']=frequent_itemsets['itemsets'].apply(lambda x: len(x))
frequent_itemsets
```

	support	itemsets	length
0	0.340	(A)	1
1	0.332	(B)	1
2	0.328	(C)	1
3	0.344	(Cash)	1
4	0.311	(Credit card)	1
5	0.170	(Electronic accessories)	1
6	0.345	(Ewallet)	1
7	0.178	(Fashion accessories)	1
8	0.501	(Female)	1
9	0.174	(Food and beverages)	1
10	0.152	(Health and beauty)	1
11	0.160	(Home and lifestyle)	1
12	0.499	(Male)	1
13	0.501	(Member)	1
14	0.499	(Normal)	1
15	0.166	(Sports and travel)	1
16	0.110	(Cash, A)	2
17	0.104	(Credit card, A)	2
18	0.126	(Ewallet, A)	2

46	0.161	(Ewallet, Member)	2
47	0.184	(Ewallet, Normal)	2
48	0.261	(Member, Female)	2
49	0.240	(Normal, Female)	2
50	0.240	(Male, Member)	2
51	0.259	(Male, Normal)	2
52	0.105	(Ewallet, Male, Normal)	3

is used to give an understanding of the association rule between different itemset:

the result shows that Male customers, without membership, paid by an E-wallet

```
frequent_itemsets [ (frequent_itemsets['length'] == 3) &(frequent_itemsets['support'] >= 0.1) ]
```

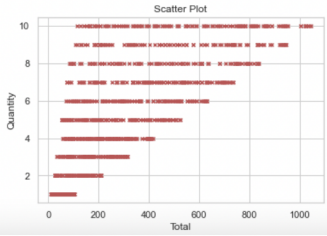
	support	itemsets	length
52	0.105	(Ewallet, Male, Normal)	3

LinearRegression

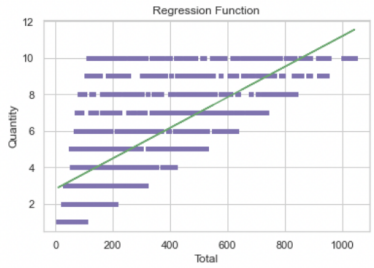
In [73]:
```
X=supermarket_df['Total']
Y=supermarket_df['Quantity']
Slic_df=pd.DataFrame({'Total':X,'Quantity':Y})
print(Slic_df)
```

```
         Total  Quantity
0     548.9715         7
1      80.2200         5
2     340.5255         7
3     489.0480         8
4     634.3785         7
..         ...       ...
995    42.3675         1
996  1022.4900        10
997    33.4320         1
998    69.1110         1
999   649.2990         7

[1000 rows x 2 columns]
```

In [74]:
```
plt.scatter(Slic_df[['Total']], Slic_df[['Quantity']],color = "r", marker = "x", s = 15)
plt.xlabel('Total')
plt.ylabel('Quantity')
plt.title('Scatter Plot')
plt.show()
```



In [77]:
```
plt.scatter(Slic_df[['Total']],Slic_df[['Quantity']], color= 'm',
            marker='s',s=10)
plt.plot(Slic_df['Total'],y_pred, color='g')
plt.xlabel('Total')
plt.ylabel('Quantity')
plt.title('Regression Function')
plt.show()
print('Coefficient: \n', classifier.coef_)
print('Intercept: \n', classifier.intercept_)
```




```
Coefficient:
 [[0.0083881]]
Intercept:
 [2.80092355]
```

In [78]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error

print('Mean squared error:')
print(mean_squared_error(Slic_df['Total'],y_pred))
```

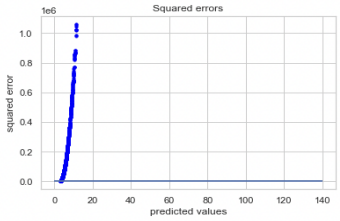
```
Mean squared error:
160168.90792514067
```

In [75]:
```
from sklearn.linear_model import LinearRegression
classifier = LinearRegression()
model = classifier.fit(Slic_df[['Total']],Slic_df[['Quantity']])
```

In [76]:
```
y_pred = classifier.predict(Slic_df[['Total']])
print(y_pred)
print('Coefficient: \n', classifier.coef_)
print('Intercept: \n', classifier.intercept_)
```

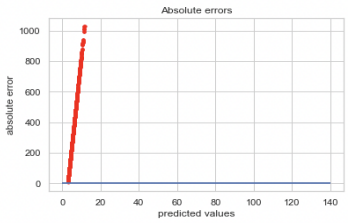
```
[ 7.37554006]
[ 4.44158502]
[ 3.5676166 ]
[ 4.74826223]
[ 3.38309943]
[ 3.5908684 ]
[ 4.77785544]
[ 3.85377234]
[ 4.44264192]
[ 5.00808354]
[ 9.41500521]
[ 6.15411569]
[ 4.95171553]
[ 3.59095648]
[ 5.53899976]
[ 7.3052562 ]
[ 6.49082649]
[ 5.90028349]
[ 3.05440346]
```

In [79]:
```
plt.scatter(y_pred, (Slic_df[['Total']] - y_pred)**2,color='blue',s=10)
plt.title('Squared errors')
plt.hlines(y=0,xmin=0 , xmax=140, linewidth=2)
plt.xlabel('predicted values')
plt.ylabel('squared error')
plt.show()
```



In [80]:
```
print('Mean absolute error: ')
print(mean_absolute_error(Slic_df['Total'],y_pred))

plt.scatter(y_pred, (Slic_df[['Total']] - y_pred),color='red',s=10)
plt.title('Absolute errors')
plt.hlines(y=0,xmin=0 , xmax=140, linewidth=2)
plt.xlabel('predicted values')
plt.ylabel('absolute error')
plt.show()
```

```
Mean absolute error:
317.456749
```



LogisticRegression

```
from sklearn.linear_model import LogisticRegression
print(supermarket_df.columns.tolist())
z= supermarket_df[['Total']]
v= supermarket_df['City']
print('z:',z.head(),'\n')
print('v:',v.head(),'\n')

classifier3= LogisticRegression(solver = 'liblinear',multi_class='ovr')
classifier3.fit(z,v)
pred= classifier3.predict(z)

print('score:',classifier3.score(z,v))
print('Coefficient: ',classifier3.coef_)
print('inercept:',classifier3.intercept_)
```

```
['Invoice ID', 'Branch', 'City', 'Customer type', 'Gender', 'Product line', 'Unit price', 'Quantity', 'Tax 5%', 'Tota
l', 'Date', 'Time', 'Payment', 'cogs', 'gross margin percentage', 'gross income', 'Rating', 'cluster']
z:     Total
0  548.9715
1   80.2200
2  340.5255
3  489.0480
4  634.3785

v: 0    1
1    0
2    1
3    1
4    1
Name: City, dtype: int64

score: 0.35
Coefficient:  [[ 3.02948948e-04]
 [-2.83905226e-04]
 [-9.81259340e-05]]
inercept: [-0.80827346 -0.56962489 -0.66359708]
```

```
print(supermarket_df.columns.tolist(),'\n')
z= supermarket_df.iloc[:,0:11]
v= supermarket_df['City']
print('z:',z.head(),'\n')
print('v:',v.head(),'\n')
from sklearn.linear_model import LogisticRegression
classifier4 = LogisticRegression()
classifier4= LogisticRegression(solver = 'liblinear',multi_class='ovr')
classifier4.fit(z,v)
pred= classifier4.predict(z)

print('score:',classifier4.score(z,v))
print('Coefficient: ',classifier4.coef_)
print('inercept:',classifier4.intercept_)
```

```
['Invoice ID', 'Branch', 'City', 'Customer type', 'Gender', 'Product line', 'Unit price', 'Quantity', 'Tax 5%', 'Tota
l', 'Date', 'Time', 'Payment', 'cogs', 'gross margin percentage', 'gross income', 'Rating', 'cluster']

z:     Invoice ID  Branch  City  Customer type  Gender  Product line  Unit price  \
0  750-67-8428       0     1              1       0             0       74.69
1  226-31-3081       2     0              0       0             1       15.28
2  631-41-3108       0     1              0       1             2       46.33
3  123-19-1176       0     1              1       1             0       58.22
4  373-73-7910       0     1              0       1             3       86.31

   Quantity   Tax 5%     Total       Date
0         7  26.1415  548.9715   1/5/2019
1         5   3.8200   80.2200   3/8/2019
2         7  16.2155  340.5255   3/3/2019
3         8  23.2880  489.0480  1/27/2019
4         7  30.2085  634.3785   2/8/2019

v: 0    1
1    0
2    1
3    1
4    1
Name: City, dtype: int64
```

Conclusion:

In conclusion, by analyzing data on sales, we discovered that subscribing to membership will reduce the general budget of customers by our loyalty card. with this strategy, the revenue and transactions will continue to rise, and we .believe we can develop the supermarket using algorithms and other methods

And we accurately reached an assessment satisfaction the client rate 7 out of 10 of our services. and 54.5% of customers have a membership, which is a suitable proportion that we targeted from the retail community.

References:

[1]"Supermarket sales", Kaggle.com, 2022. [Online]. Available: https://www.kaggle.com/datasets/aungpyaeap/supermarket-sales.
[2]"seaborn: statistical data visualization — seaborn 0.11.2 documentation", Seaborn.pydata.org, 2022. [Online]. Available: https://seaborn.pydata.org/.
[3]"Matplotlib — Visualization with Python", Matplotlib.org, 2022. [Online]. Available: https://matplotlib.org/.
[4]"OS Module in Python with Examples - GeeksforGeeks", GeeksforGeeks, 2022. [Online]. Available: https://www.geeksforgeeks.org/os-module-python-examples/.
[5]"Python map() function - GeeksforGeeks", GeeksforGeeks, 2022. [Online]. Available: https://www.geeksforgeeks.org/python-map-function/.