asg2

June 29, 2020

[176]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from apyori import apriori
import numpy as np
import sklearn
from sklearn import cluster
from sklearn.cluster import KMeans
import networkx as nx

%matplotlib inline
```

[125]:
```
df = pd.read_csv("./barbershop.csv")
df.head()
```

[125]:
```
   cellphone    hairstyle  cost  monthly_rent  transport_cost  \
0  819107859         Trim    80        2500.0             800
1  818659596  Temple Fade    70        2800.0             760
2  817122486     Low Fade    70        3500.0             730
3  813777117       Mohawk    70        3800.0             410
4  817028800     Low Fade    60        2200.0             580

   clients_per_month  monthly_income form_of_transport working_hours  \
0                380           13000              taxi       6am-5pm
1                540           14000       private_car       6am-5pm
2                520           14000              taxi      6am-12pm
3                360           14000              taxi       6am-5pm
4                550           15000              taxi      6am-12pm

   clients_cut per trip shop_type promotion_platform_1 promotion_platform_2
0                   NaN  building                 none                 none
1                   NaN  building            instagram                 none
2                   NaN  building            instagram             whatsapp
3                   NaN  building                 none                 none
4                   NaN  building                 none                 none
```

What we are mining -which barbershop type makes more money(traditional or online based) and why

Data preprossing -pre-data mining

-cleaning

[22]:
```
#check if there are any null values
df.isnull().any()
```

[22]:
```
cellphone              False
hairstyle              False
cost                   False
monthly_rent            True
transport_cost         False
clients_per_month      False
monthly_income         False
form_of_transport      False
working_hours          False
clients_per_trip        True
shop_type              False
promotion_platform_1   False
promotion_platform_2   False
dtype: bool
```

Replace missing values depending on the data type,

[27]:
```
median = df['transport_cost'].median()
df.fillna({'monthly_rent':0, 'clients_per_trip': 0, 'transport_cost':median},␣
 ↪inplace=True)
# res = df.apply(lambda x: x.fillna(0) if x.dtype.kind in 'biufc' else x.
 ↪fillna('.'))
```

[29]:
```
#drop redundant working_hours column
df.drop(columns=['working_hours', 'cellphone'], inplace=True)
#check data types of our column,
#this guides as what type operations we can perform on them
#and the graphs that will be suitable to represent them
df.dtypes
```

[29]:
```
hairstyle                object
cost                      int64
monthly_rent            float64
transport_cost            int64
clients_per_month         int64
monthly_income            int64
form_of_transport        object
clients_per_trip        float64
shop_type                object
promotion_platform_1     object
promotion_platform_2     object
```

```
dtype: object
```

Calculate the avarage null values in the dataset This tells us that: overrall most of the columns are filled 50% of the records in the monthly rent column are null while only 49% in clients cut per trip are null

[5]:
```python
df.isnull().sum() / len(df)
```

[5]:
```
cellphone              0.00
hairstyle              0.00
cost                   0.00
monthly_rent           0.50
transport_cost         0.00
clients_per_month      0.00
monthly_income         0.00
form_of_transport      0.00
working_hours          0.00
clients_cut per trip   0.48
shop_type              0.00
promotion_platform_1   0.00
promotion_platform_2   0.00
dtype: float64
```

Format all string types to lower case

[ ]:
```python
df['hairstyle'].str.lower()
```

Fix column names that might cause issues down the road

[19]:
```python
df.rename(columns={'clients_cut per trip':'clients_per_trip'}, inplace=True)
```

1 1.Data Modeling

Data modeling refers to a group of processes in which multiple sets of data are combined and analyzed to uncover relationships or patterns. The goal of data modeling is to use past data to inform future efforts.

[30]:
```python
df.describe()
```

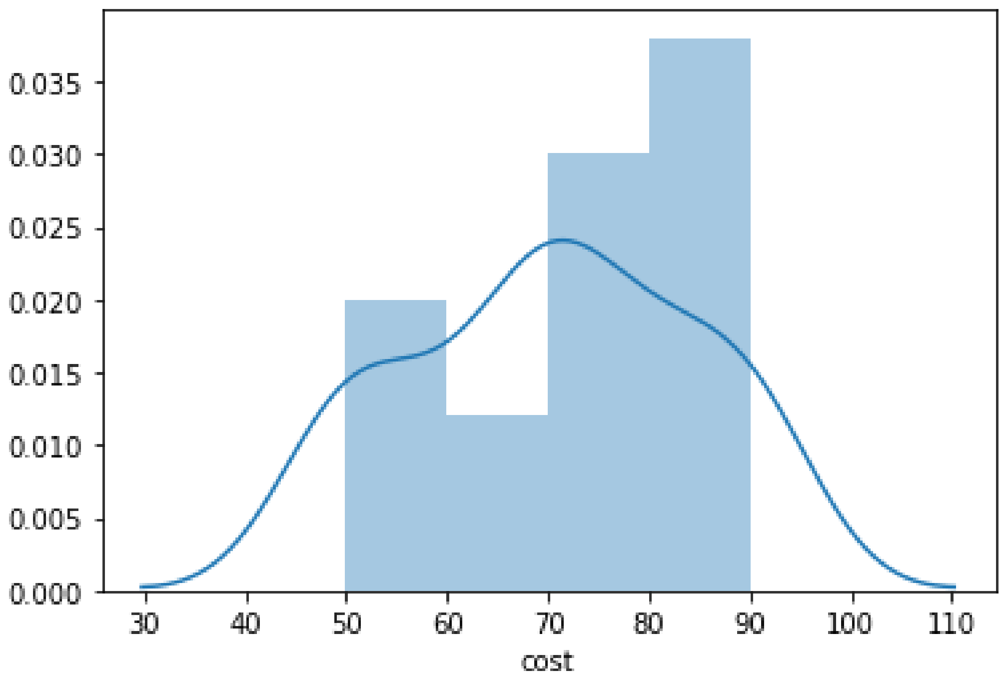
[30]:
```
            cost  monthly_rent  transport_cost  clients_per_month  \
count  50.000000      50.00000       50.000000          50.000000
mean   70.600000    1484.00000      590.600000         460.800000
std    13.910795    1548.45018      120.516067          69.188887
min    50.000000       0.00000      400.000000         340.000000
25%    60.000000       0.00000      485.000000         382.500000
50%    70.000000    1000.00000      615.000000         470.000000
75%    80.000000    3100.00000      697.500000         520.000000
```

```
max     90.000000     3800.00000       800.000000          550.000000

       monthly_income  clients_per_trip
count        50.00000         50.000000
mean      21460.00000          1.260000
std        5639.54695          1.723902
min       12000.00000          0.000000
25%       16250.00000          0.000000
50%       22000.00000          1.000000
75%       26000.00000          2.000000
max       34000.00000          7.000000
```

Data Distribution

[36]:
```
filter_data = df.dropna(subset=['cost'])
sns.distplot(filter_data['cost'])
```

[36]: <matplotlib.axes._subplots.AxesSubplot at 0x7fc1346f35d0>



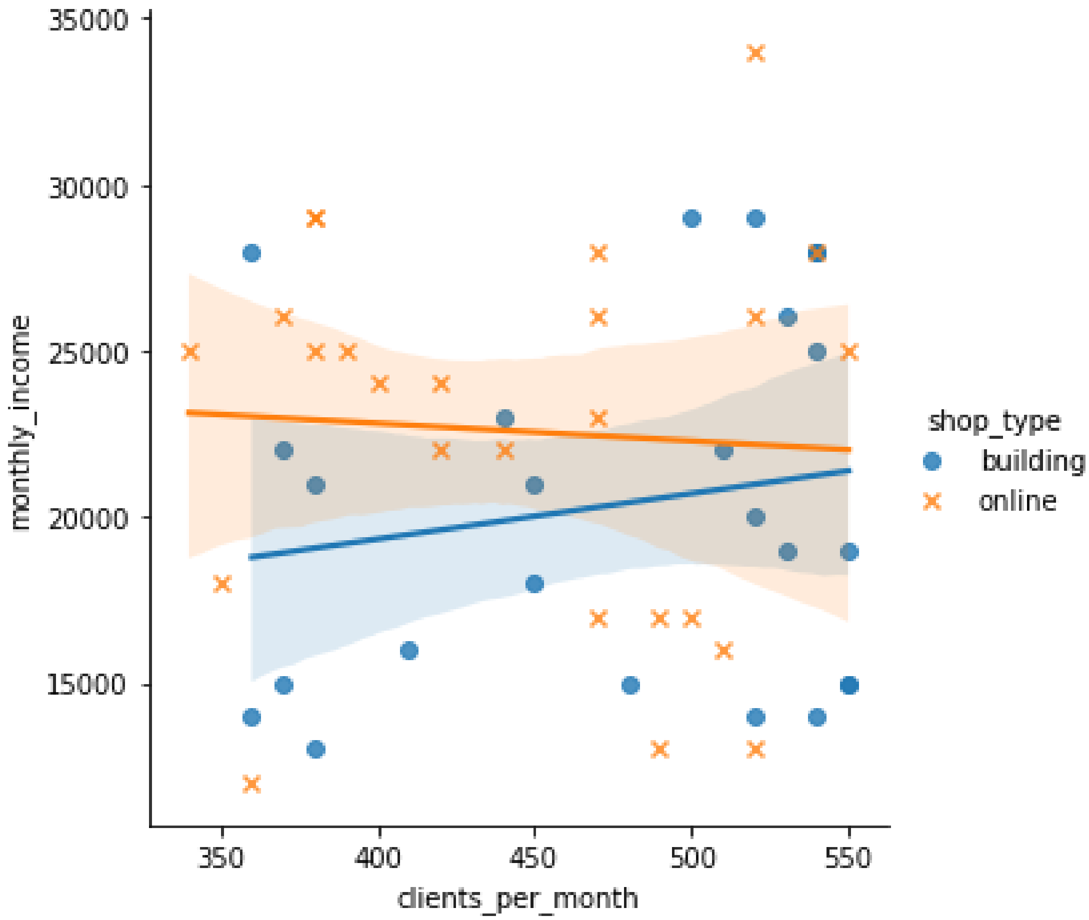
Statistical Regression

[5]:
```
#linear regression
import statsmodels.api as sm
from statsmodels.formula.api import ols
```

[6]:
```python
m = ols('monthly_income ~ clients_per_month + transport_cost', df).fit()
print(m.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:         monthly_income   R-squared:                       0.004
Model:                            OLS   Adj. R-squared:                 -0.038
Method:                 Least Squares   F-statistic:                   0.09293
Date:                Sun, 28 Jun 2020   Prob (F-statistic):              0.911
Time:                        03:39:50   Log-Likelihood:                -502.22
No. Observations:                  50   AIC:                             1010.
Df Residuals:                      47   BIC:                             1016.
Df Model:                           2
Covariance Type:            nonrobust
================================================================================
=====
                        coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-----
Intercept           1.93e+04   6686.463      2.886      0.006    5846.373
3.27e+04
clients_per_month     1.0446     11.880      0.088      0.930     -22.855
24.944
transport_cost        2.8460      6.820      0.417      0.678     -10.875
16.567
==============================================================================
Omnibus:                        7.759   Durbin-Watson:                   0.218
Prob(Omnibus):                  0.021   Jarque-Bera (JB):                2.502
Skew:                          -0.030   Prob(JB):                        0.286
Kurtosis:                       1.906   Cond. No.                     6.22e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 6.22e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[8]:
```python
sns.lmplot(x="clients_per_month", y="monthly_income", hue="shop_type", data=df,
 ↪markers=["o", "x"])
plt.show()
```



2 2.Identifying Patterns

[177]:
```
df2 = df[['monthly_income','cost']]

records = []
for i in range(0, df2.shape[0]):
    records.append([str(df2.values[i,j]) for j in range(0, len(df2.columns))])

association_rules = apriori(records, min_support=0.0045, min_confidence=0.2,␣
↪min_lift=3, min_length=2, use_colnames = True)
association_results = list(association_rules)
```

[178]:
```
for item in association_results:

    # first index of the inner list
    # Contains base item and add item
```

```
    pair = item[0]
    items = [x for x in pair]
    print("Rule: " + items[0] + " -> " + items[1])

    #second index of the inner list
    print("Support: " + str(item[1]))

    #third index of the list located at 0th
    #of the third index of the inner list

    print("Confidence: " + str(item[2][0][2]))
    print("Lift: " + str(item[2][0][3]))
    print("=====================================")
```

```
Rule: 90 -> 12000
Support: 0.02
Confidence: 1.0
Lift: 5.0
=====================================
Rule: 14000 -> 70
Support: 0.06
Confidence: 1.0
Lift: 3.3333333333333335
=====================================
Rule: 17000 -> 50
Support: 0.04
Confidence: 0.6666666666666667
Lift: 3.3333333333333335
=====================================
Rule: 80 -> 18000
Support: 0.04
Confidence: 1.0
Lift: 5.555555555555555
=====================================
Rule: 70 -> 20000
Support: 0.02
Confidence: 1.0
Lift: 3.3333333333333335
=====================================
Rule: 60 -> 23000
Support: 0.02
Confidence: 0.5
Lift: 4.166666666666667
=====================================
Rule: 60 -> 25000
Support: 0.04
Confidence: 0.39999999999999997
```

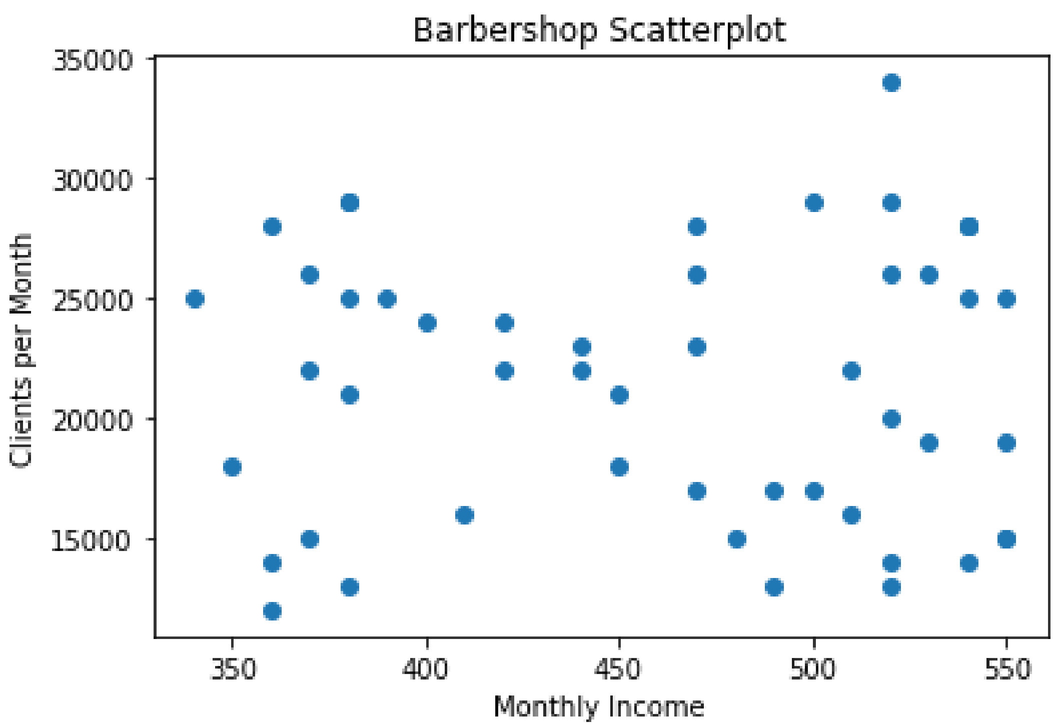
```
Lift: 3.333333333333333
=====================================
Rule: 60 -> 28000
Support: 0.04
Confidence: 0.39999999999999997
Lift: 3.333333333333333
=====================================
Rule: 80 -> 34000
Support: 0.02
Confidence: 1.0
Lift: 5.555555555555555
=====================================
```

3 3.Data Visualization

Scatter Plot

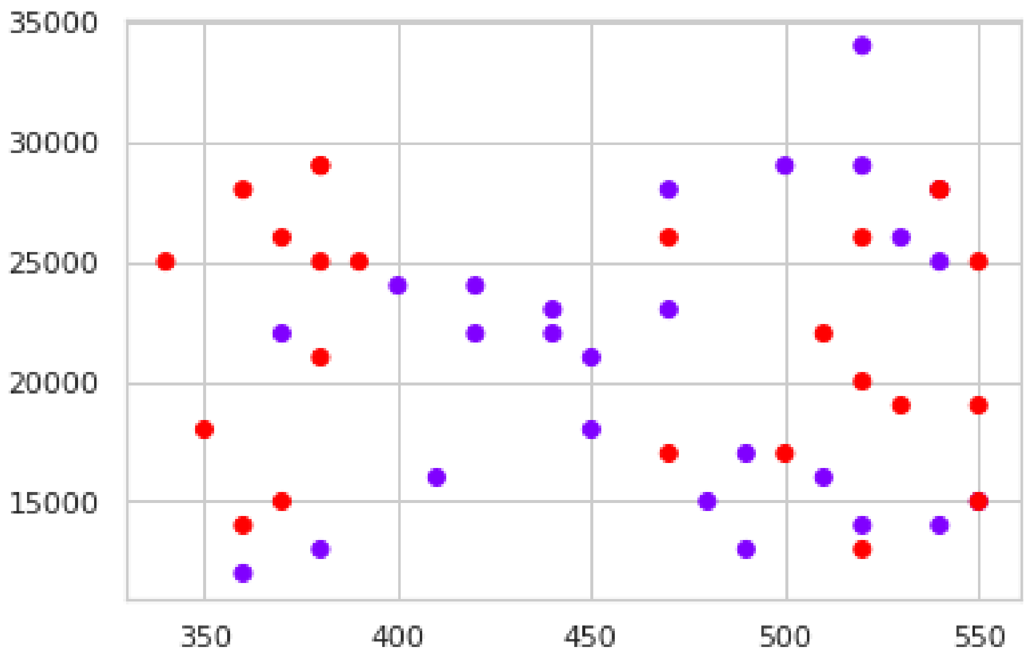
[19]:
```
plt.scatter(df.clients_per_month, df.monthly_income)
plt.title('Barbershop Scatterplot')
plt.xlabel('Monthly Income')
plt.ylabel('Clients per Month')
```

[19]:
```
Text(0, 0.5, 'Clients per Month')
```



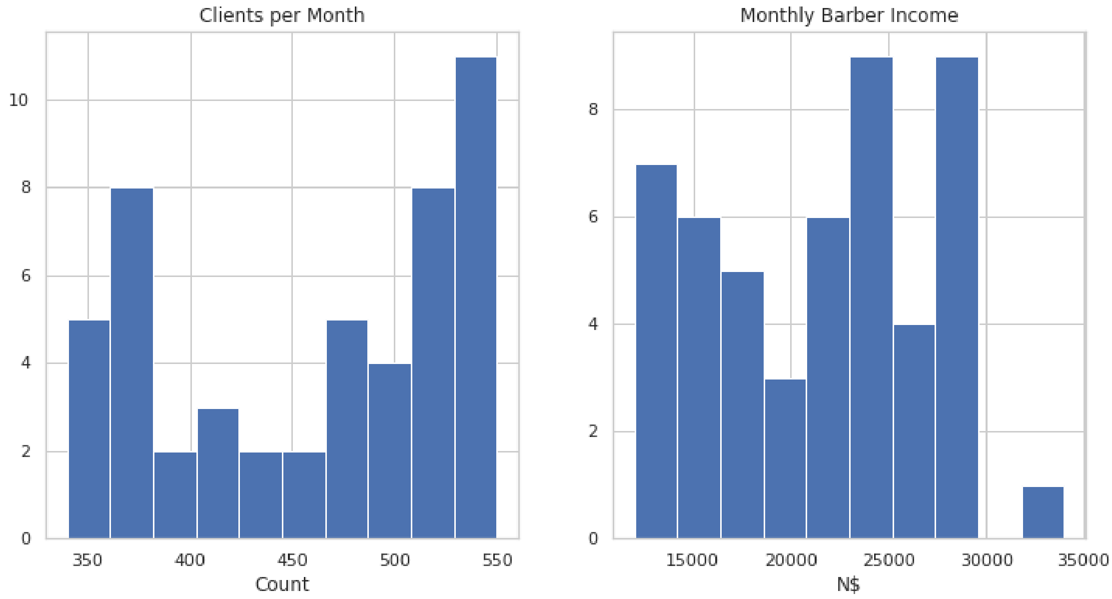
Data Cluster

[132]:
```
df2 = df.dropna(axis='columns')
kmeans = KMeans(n_clusters=2).fit(df2._get_numeric_data())
centroids = kmeans.cluster_centers_

plt.scatter(df['clients_per_month'], df['monthly_income'], c=kmeans.labels_,␣
 ↪cmap='rainbow')
plt.show()
```



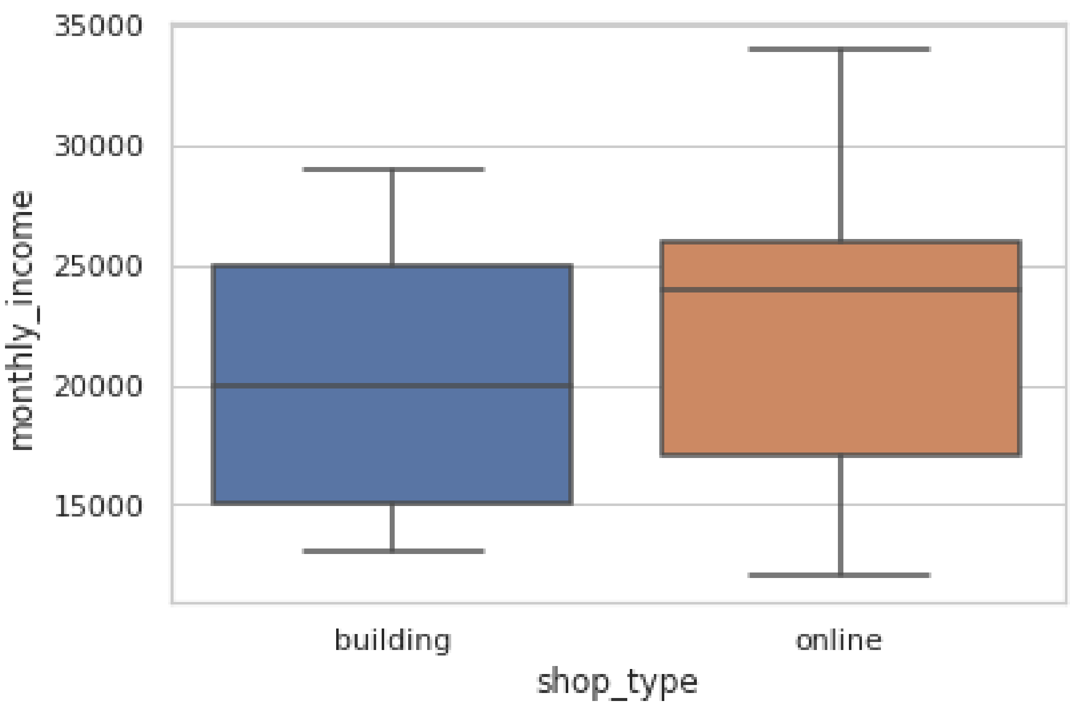
Histogram

[118]:
```
fig = plt.figure(figsize=(12, 6))
clients = fig.add_subplot(121)
income = fig.add_subplot(122)

clients.hist(df.clients_per_month)
clients.set_xlabel('Count')
clients.set_title("Clients per Month")

income.hist(df.monthly_income)
income.set_xlabel('N$')
income.set_title("Monthly Barber Income")

plt.show()
```



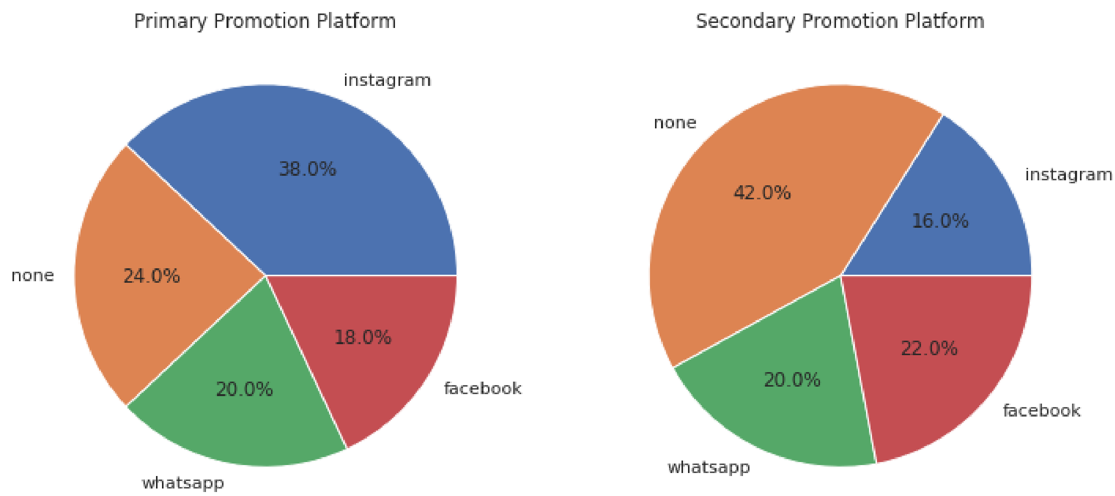
Box Plot

[119]:
```
ax = sns.boxplot(x='shop_type', y='monthly_income', data=df, orient="v")
```



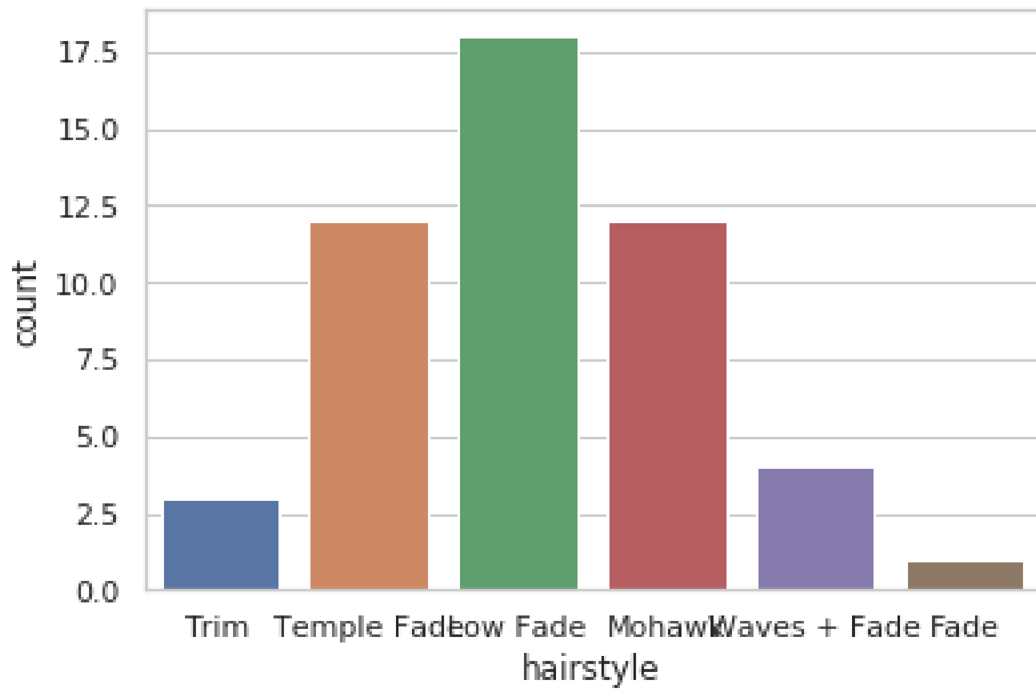
Pie Chart

[112]:
```
fig = plt.figure()
fig.set_figheight(12)
fig.set_figwidth(12)

one = fig.add_subplot(121)
two = fig.add_subplot(122)

one.set_title("Primary Promotion Platform")
two.set_title("Secondary Promotion Platform")
type_counts = df['promotion_platform_1'].value_counts()
df2 = pd.DataFrame({'prtype': type_counts},
                   index = ['instagram', 'none', 'whatsapp', 'facebook'])

one.pie(df2['prtype'],labels=['instagram', 'none', 'whatsapp', 'facebook'],␣
↪autopct='%1.1f%%')
type_counts = df['promotion_platform_2'].value_counts()
df2 = pd.DataFrame({'prtype': type_counts},
                   index = ['instagram', 'none', 'whatsapp', 'facebook'])

two.pie(df2['prtype'],labels=['instagram', 'none', 'whatsapp', 'facebook'],␣
↪autopct='%1.1f%%')

plt.show(block=False)
```



Bar Graph

[40]:
```
ax = sns.countplot(x='hairstyle', data=df)
```



4 4.Prototype

(Transitioning modelling - show using a graph and little explanation)

[ ]:

```

```