In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [6]:

```
!pip install openpyxl
```

```
Collecting openpyxl
  Downloading openpyxl-3.0.7-py2.py3-none-any.whl (243 kB)
     |████████████████████████████████| 243 kB 282 kB/s eta 0:00:01
Collecting et-xmlfile
  Downloading et_xmlfile-1.1.0-py3-none-any.whl (4.7 kB)
Installing collected packages: et-xmlfile, openpyxl
Successfully installed et-xmlfile-1.1.0 openpyxl-3.0.7
WARNING: Running pip as root will break packages and permissions. You should
install packages reliably by using venv: https://pip.pypa.io/warnings/venv
 (https://pip.pypa.io/warnings/venv)
```

In [7]:

```
df=pd.read_excel('../input/ecommerce-platform-analysis-and-prediction/E-commerce.xlsx')
df.head()
```

Out[7]:

| ıder of ındent | 2 How old are you? | 3 Which city do you shop online from? | 4 What is the Pin Code of where you shop online from? | 5 Since How Long You are Shopping Online ? | 6 How many times you have made an online purchase in the past 1 year? | 7 How do you access the internet while shopping on-line? | 8 Which device do you use to access the online shopping? | 9 What is the screen size of your mobile device? \t\t\t\t\t\t | 10 What ope system (( your de |
|---|---|---|---|---|---|---|---|---|---|
| Male | 31-40 years | Delhi | 110009 | Above 4 years | 31-40 times | Dial-up | Desktop | Others | Window/wi l |
| 'emale | 21-30 years | Delhi | 110030 | Above 4 years | 41 times and above | Wi-Fi | Smartphone | 4.7 inches | IO |
| 'emale | 21-30 years | Greater Noida | 201308 | 3-4 years | 41 times and above | Mobile Internet | Smartphone | 5.5 inches | A |
| Male | 21-30 years | Karnal | 132001 | 3-4 years | Less than 10 times | Mobile Internet | Smartphone | 5.5 inches | IO |
| 'emale | 21-30 years | Bangalore | 530068 | 2-3 years | 11-20 times | Wi-Fi | Smartphone | 4.7 inches | IO |

71 columns

In [8]:

```
#Setting option to show max rows and max columns
pd.set_option("display.max_columns",None)
pd.set_option("display.max_rows", None)
```

## Pre-processing the columns names

In [9]:

```
from string import digits

#Removing tab spaces
df.columns = df.columns.str.replace('\t','')

#Removing digits
remove_digits = str.maketrans('', '', digits)
df.columns = df.columns.str.translate(remove_digits)

#Removing leading and trailling spaces
df.columns = df.columns.str.strip()
```

In [10]:

```
df.head()
```

Out[10]:

| | Gender of respondent | How old are you? | Which city do you shop online from? | What is the Pin Code of where you shop online from? | Since How Long You are Shopping Online ? | How many times you have made an online purchase in the past year? | How do you access the internet while shopping on-line? | Which device do you use to access the online shopping? | What is the screen size of your mobile device? |
|---|---|---|---|---|---|---|---|---|---|
| **0** | Male | 31-40 years | Delhi | 110009 | Above 4 years | 31-40 times | Dial-up | Desktop | Others |
| **1** | Female | 21-30 years | Delhi | 110030 | Above 4 years | 41 times and above | Wi-Fi | Smartphone | 4.7 inches |
| **2** | Female | 21-30 years | Greater Noida | 201308 | 3-4 years | 41 times and above | Mobile Internet | Smartphone | 5.5 inches |
| **3** | Male | 21-30 years | Karnal | 132001 | 3-4 years | Less than 10 times | Mobile Internet | Smartphone | 5.5 inches |
| **4** | Female | 21-30 years | Bangalore | 530068 | 2-3 years | 11-20 times | Wi-Fi | Smartphone | 4.7 inches |

In [11]:

```
df.shape
```

Out[11]:

```
(269, 71)
```

Dataset have 269 rows and 71 columns

In [12]:

```
df.dtypes
```

Out[12]:

```
Gender of respondent
object
How old are you?
object
Which city do you shop online from?
object
What is the Pin Code of where you shop online from?
int64
Since How Long You are Shopping Online ?
object
How many times you have made an online purchase in the past  year?
object
How do you access the internet while shopping on-line?
object
Which device do you use to access the online shopping?
object
What is the screen size of your mobile device?
object
```

All the columns are of object datatype except for pincode column which is of int type

In [13]:

```
df.isnull().sum().any()
```

Out[13]:

```
False
```

```
There are no null values is the dataset
```

In [14]:

```
df.nunique()
```

Out[14]:

```
Gender of respondent
2
How old are you?
5
Which city do you shop online from?
11
What is the Pin Code of where you shop online from?
39
Since How Long You are Shopping Online ?
5
How many times you have made an online purchase in the past  year?
6
How do you access the internet while shopping on-line?
4
Which device do you use to access the online shopping?
4
What is the screen size of your mobile device?
4
```

All the columns are of categorical types. There are no identifier or constant columns

# Univariate Analysis

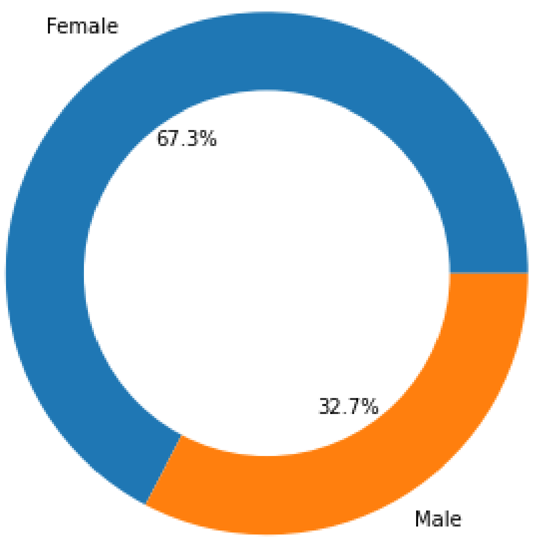
In [15]:

```
personal_info=['Gender of respondent','How old are you?','Which city do you shop onlin
               'What is the Pin Code of where you shop online from?','Since How Long Yo
                   'How many times you have made an online purchase in the past  year?
```

### Personal Info

In [16]:

```
for i in personal_info:
    if i!='What is the Pin Code of where you shop online from?':
        plt.figure(figsize=(8,6))
        df[i].value_counts().plot.pie(autopct='%1.1f%%')
        centre=plt.Circle((0,0),0.7,fc='white')
        fig=plt.gcf()
        fig.gca().add_artist(centre)
        plt.xlabel(i)
        plt.ylabel('')
        plt.figure()
```




-There is double the number of women than men who have taken this survey.
-Most of the people are in their 30's followed by 20's, teenagers and senior citizen are the least in number.
-Most of the people belong from delhi, noida and banglore, ambiguity can also be seen as noida has two categories (noida and grater noida) which need to be handled
-Most of the people shopping online have been shopping from a long time.
-Majority of people shop online 10 times a year, amiguity can also be seen for range 42 times and above which needs to be handled

# Analysis on the basis of Various following factors

## Intention of Repeat purchase:

In [17]:

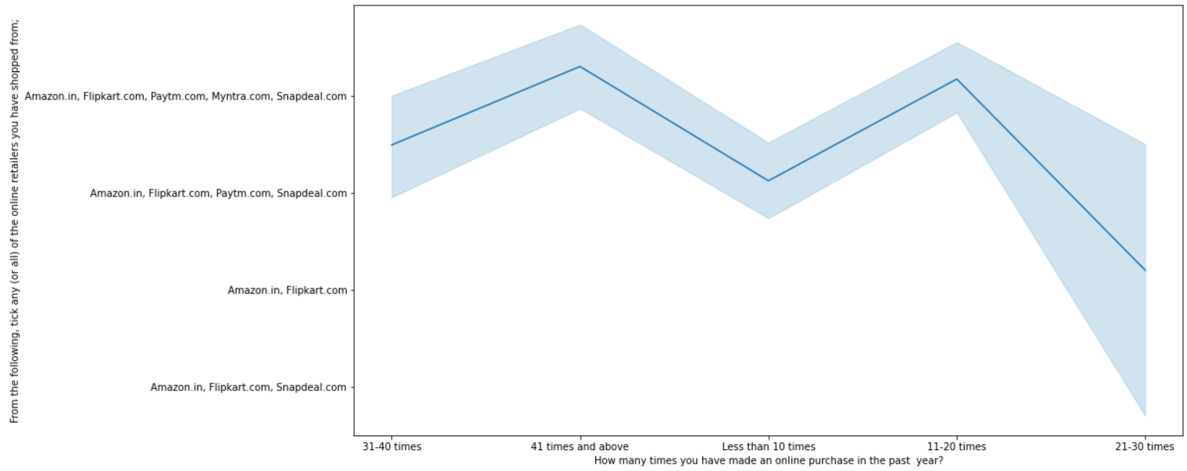
```
#Resolving ambiguity of column
#Changing 42 times and above to 41 times and above
df['How many times you have made an online purchase in the past  year?'].replace('42 ti
                                                                              inplace
```

In [18]:

```
plt.figure(figsize=(15,8))
sns.lineplot(df['How many times you have made an online purchase in the past  year?'],
              df['From the following, tick any (or all) of the online retailers you hav
```

Out[18]:

```
<AxesSubplot:xlabel='How many times you have made an online purchase in the
past  year?', ylabel='From the following, tick any (or all) of the online re
tailers you have shopped from;'>
```




Heavy shoppers who shop more than 41 times a year shop from all the online brands, some of the people who shop for 32-40 and less than 10 times a year seem to exclude myntra. People shop from Amazon and flipkart whatever be the case.

#### *Converting years to numbers for better analysis*

In [19]:

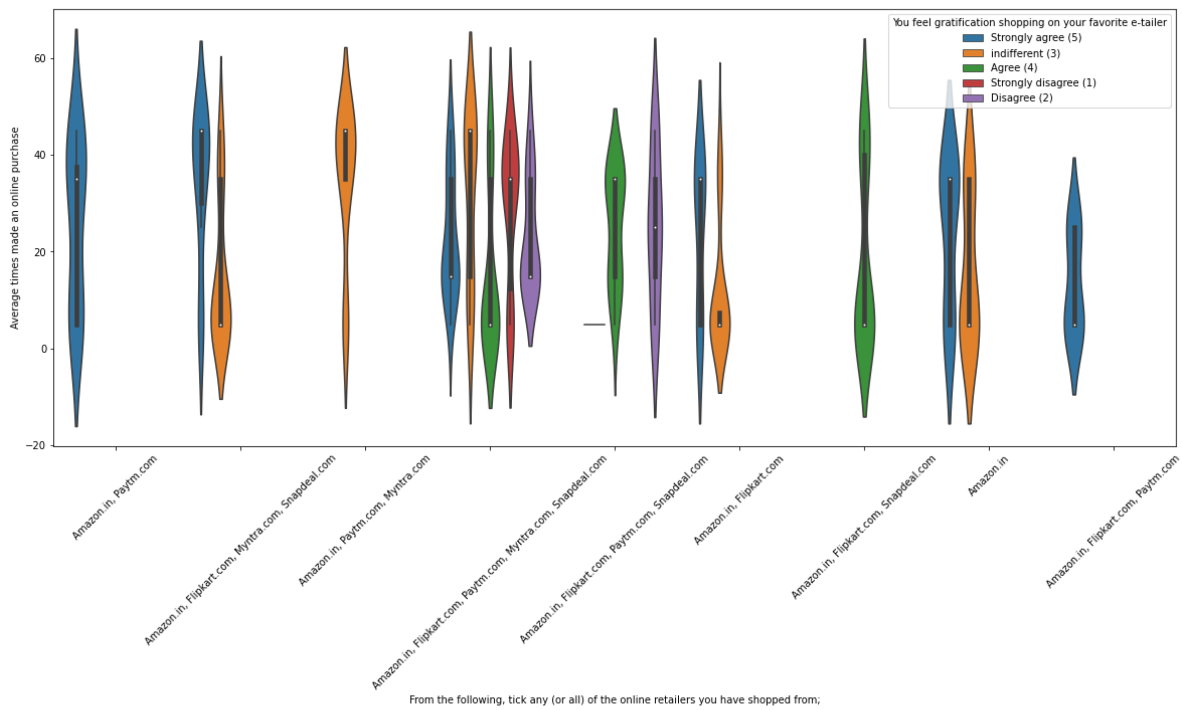
```
dict={'31-40 times':35,'41 times and above':45,'Less than 10 times':5,'11-20 times':15,
df['Average times made an online purchase']=df['How many times you have made an online
```

In [20]:

```
plt.figure(figsize=(20,8))
sns.violinplot(df['From the following, tick any (or all) of the online retailers you ha
                df['Average times made an online purchase'],hue=df['You feel gratificati
plt.xticks(rotation=45)
```

Out[20]:

```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8]),
 [Text(0, 0, 'Amazon.in, Paytm.com'),
  Text(1, 0, 'Amazon.in, Flipkart.com, Myntra.com, Snapdeal.com'),
  Text(2, 0, 'Amazon.in, Paytm.com, Myntra.com'),
  Text(3, 0, 'Amazon.in, Flipkart.com, Paytm.com, Myntra.com, Snapdeal.co
m'),
  Text(4, 0, 'Amazon.in, Flipkart.com, Paytm.com, Snapdeal.com'),
  Text(5, 0, 'Amazon.in, Flipkart.com'),
  Text(6, 0, 'Amazon.in, Flipkart.com, Snapdeal.com'),
  Text(7, 0, 'Amazon.in'),
  Text(8, 0, 'Amazon.in, Flipkart.com, Paytm.com')])
```



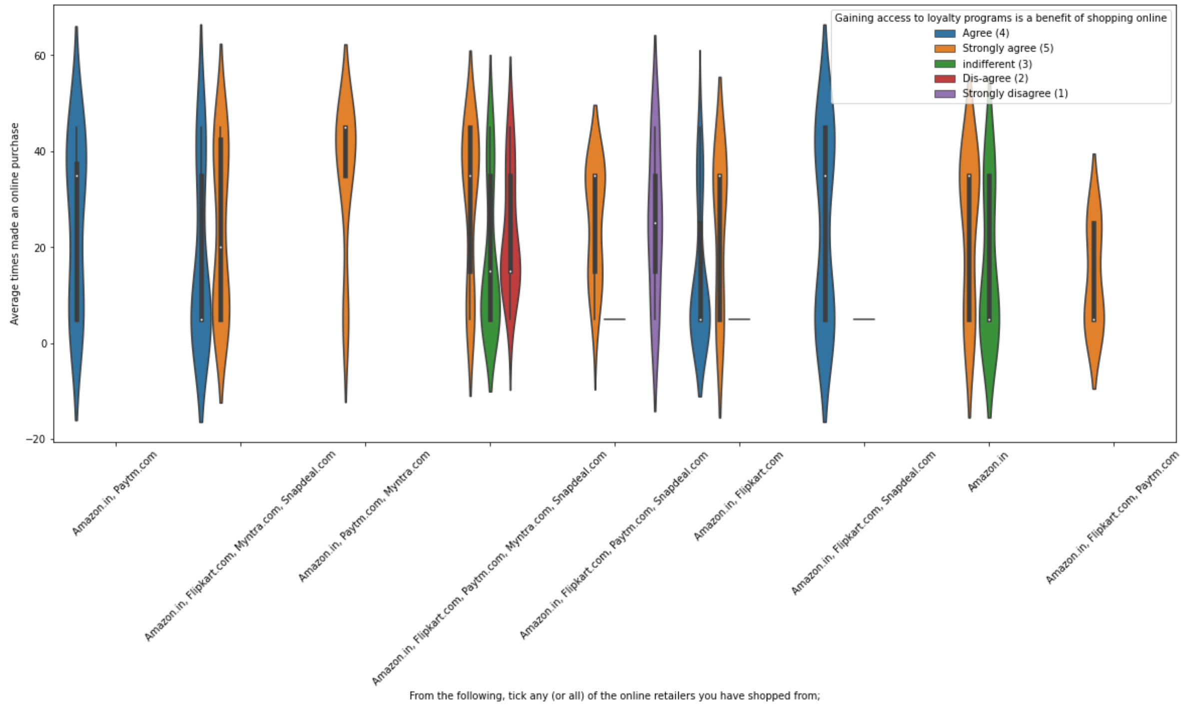

Almost all the people who have shopped from amazon, flipkart and paytm are satisfied. People who shop from a more number of online brands dosent seem to be satisfied.

In [21]:

```
plt.figure(figsize=(20,8))
sns.violinplot(df['From the following, tick any (or all) of the online retailers you ha
               df['Average times made an online purchase'],hue=df['Gaining access to lo
plt.xticks(rotation=45)
```

Out[21]:

```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8]),
 [Text(0, 0, 'Amazon.in, Paytm.com'),
  Text(1, 0, 'Amazon.in, Flipkart.com, Myntra.com, Snapdeal.com'),
  Text(2, 0, 'Amazon.in, Paytm.com, Myntra.com'),
  Text(3, 0, 'Amazon.in, Flipkart.com, Paytm.com, Myntra.com, Snapdeal.co
m'),
  Text(4, 0, 'Amazon.in, Flipkart.com, Paytm.com, Snapdeal.com'),
  Text(5, 0, 'Amazon.in, Flipkart.com'),
  Text(6, 0, 'Amazon.in, Flipkart.com, Snapdeal.com'),
  Text(7, 0, 'Amazon.in'),
  Text(8, 0, 'Amazon.in, Flipkart.com, Paytm.com')])
```




People shopping from amazon and paytm are getting benefits from the loyalty points, flipkart and sanpdeal also seem to give such benefits but people who shop from almost everywhere disagree with this statement too

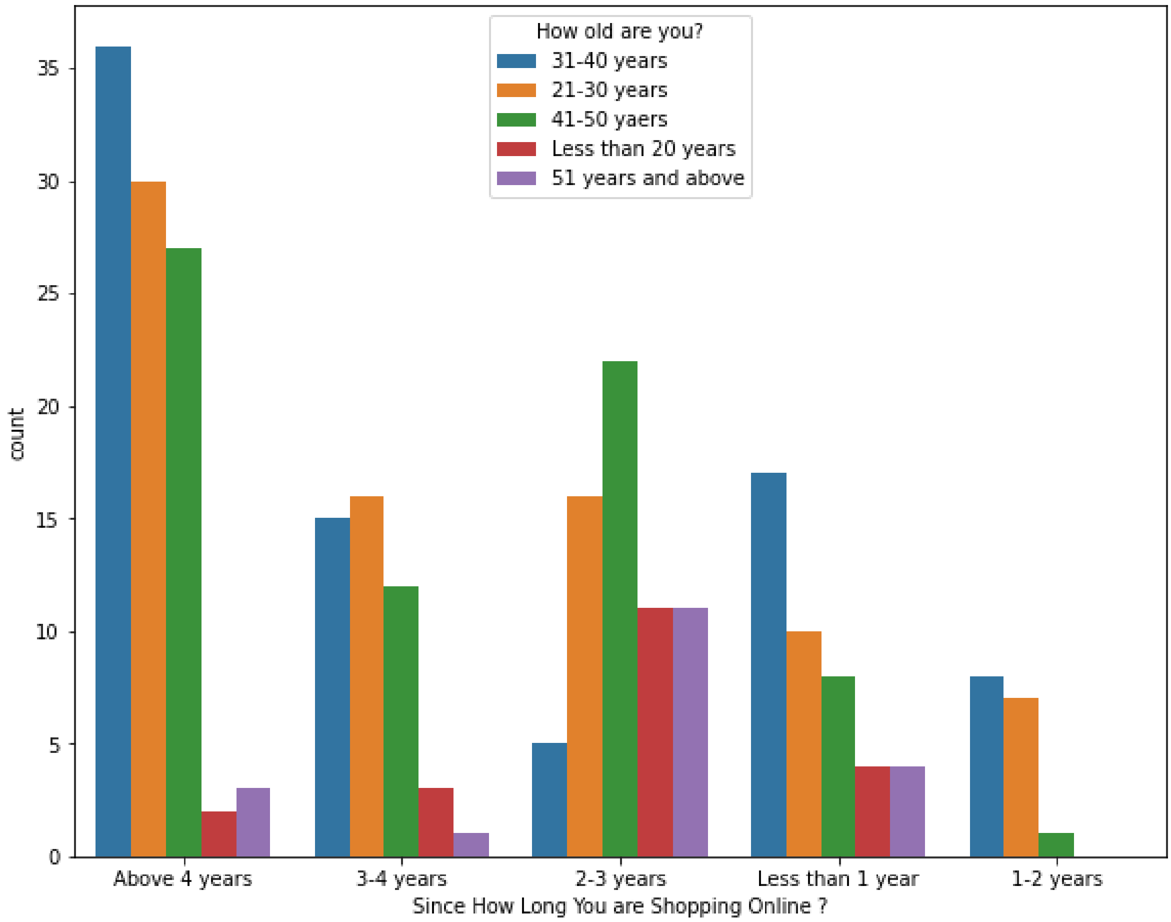
## Online Retailing:

In [22]:

```
plt.figure(figsize=(10,8))
sns.countplot(df['Since How Long You are Shopping Online ?'],hue=df['How old are you?']
```

Out[22]:

```
<AxesSubplot:xlabel='Since How Long You are Shopping Online ?', ylabel='coun
t'>
```




Highest number of people have been shopping online for above 4 years except for the age group below 20 years and above 50 years. People who are shopping online for 1-2 years does not include teenagers and elder people.

```
##### Converting Years to numbers for better analysis
```

In [23]:

```
df['Since How Long You are Shopping Online ?'].unique()
```

Out[23]:

```
array(['Above 4 years', '3-4 years', '2-3 years', 'Less than 1 year',
       '1-2 years'], dtype=object)
```

In [24]:

```
dict={'Above 4 years':4.5,'3-4 years':3.5,'2-3 years':2.5,'1-2 years':1.5,'Less than 1
df['Average years of shopping online']=df['Since How Long You are Shopping Online ?'].
```

In [25]:

```
df['Which city do you shop online from?'].unique()
```

Out[25]:

```
array(['Delhi', 'Greater Noida', 'Karnal ', 'Bangalore ', 'Noida',
       'Solan', 'Moradabad', 'Gurgaon ', 'Merrut', 'Ghaziabad',
       'Bulandshahr'], dtype=object)
```

In [26]:

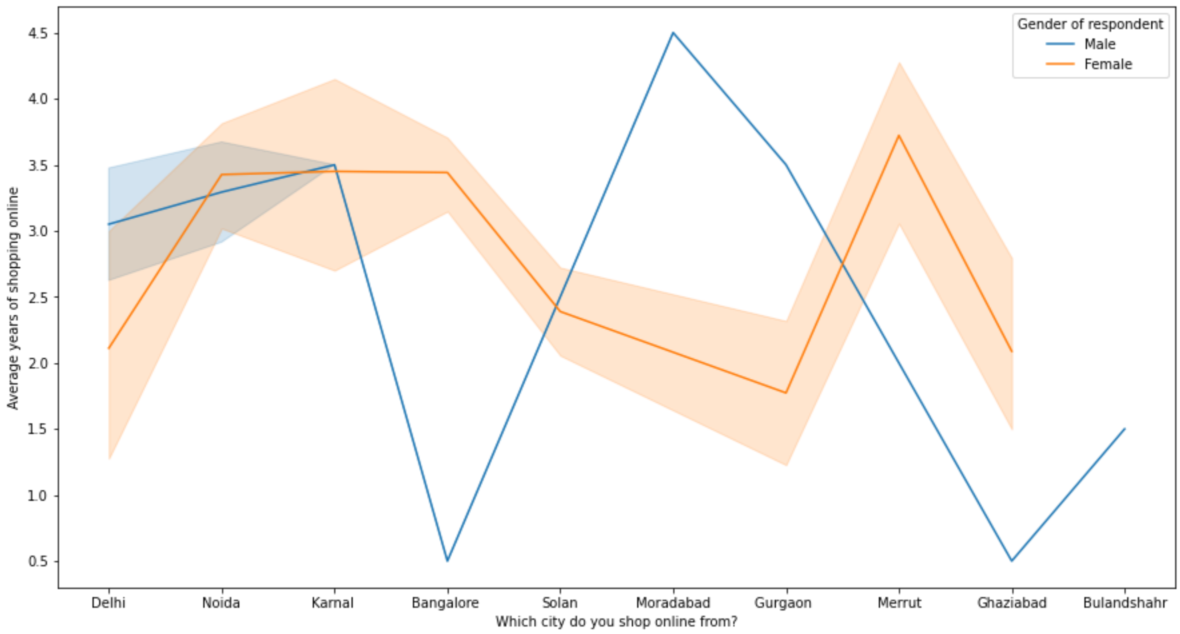
```
#Changing Greater noida to noida
df['Which city do you shop online from?'].replace({'Greater Noida':'Noida'},inplace=Tru
```

In [27]:

```
plt.figure(figsize=(15,8))
sns.lineplot(df['Which city do you shop online from?'],df['Average years of shopping or
```

Out[27]:

```
<AxesSubplot:xlabel='Which city do you shop online from?', ylabel='Average y
ears of shopping online'>
```



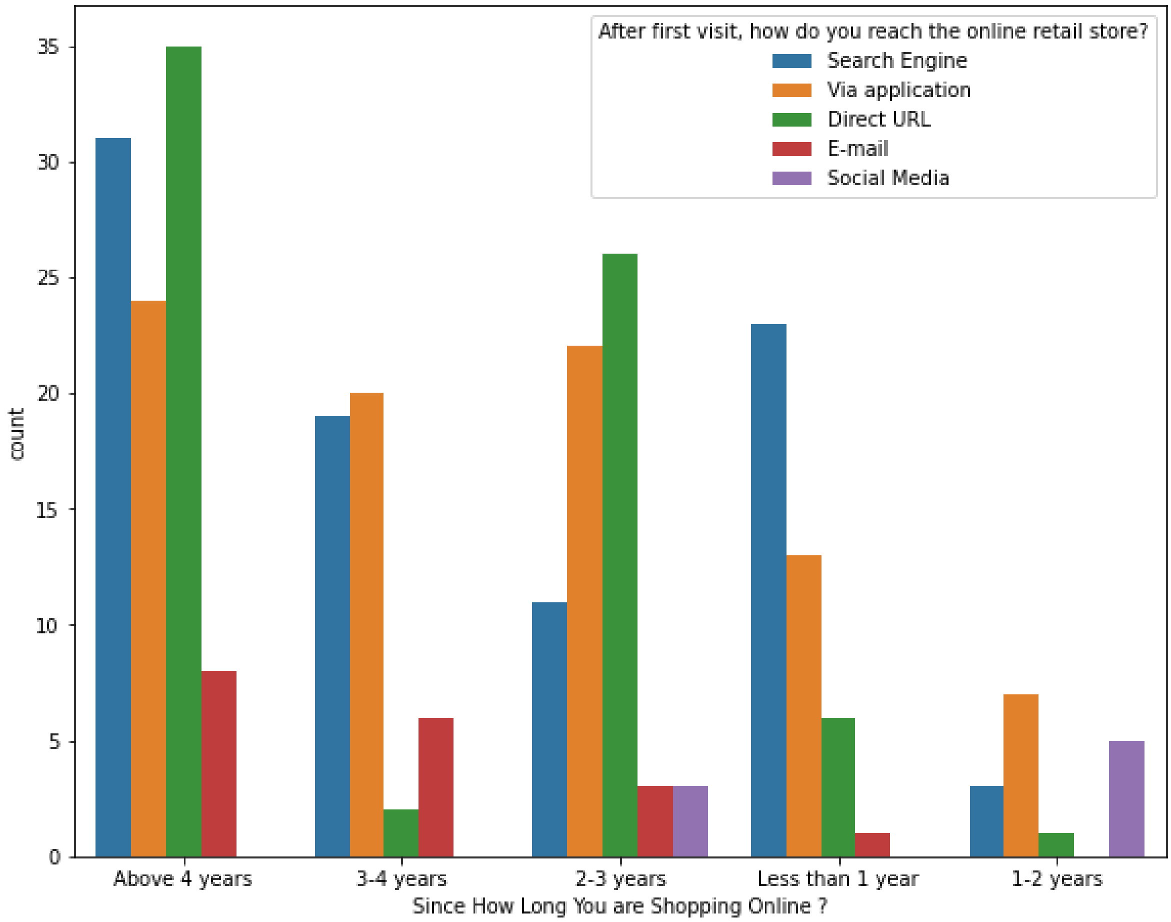
In lines, we can see that density of female customers is more than male. Men living in banglore and ghaziabad shop have shopped online for less than 1 year. Highest number of men shopping online belong from delhi and noida, while men from moradabad have been shopping online for the longest. Women from meerut and noida have shopped the longest.

In [28]:

```
plt.figure(figsize=(10,8))
sns.countplot(df['Since How Long You are Shopping Online ?'],
              hue=df['After first visit, how do you reach the online retail store?'])
```

Out[28]:

<AxesSubplot:xlabel='Since How Long You are Shopping Online ?', ylabel='count'>




Even though people who are shopping online for more than 3 years donot use the application rather use search engine and direct url's in large number which indicates that online brands should update all their platforms rather than just application.

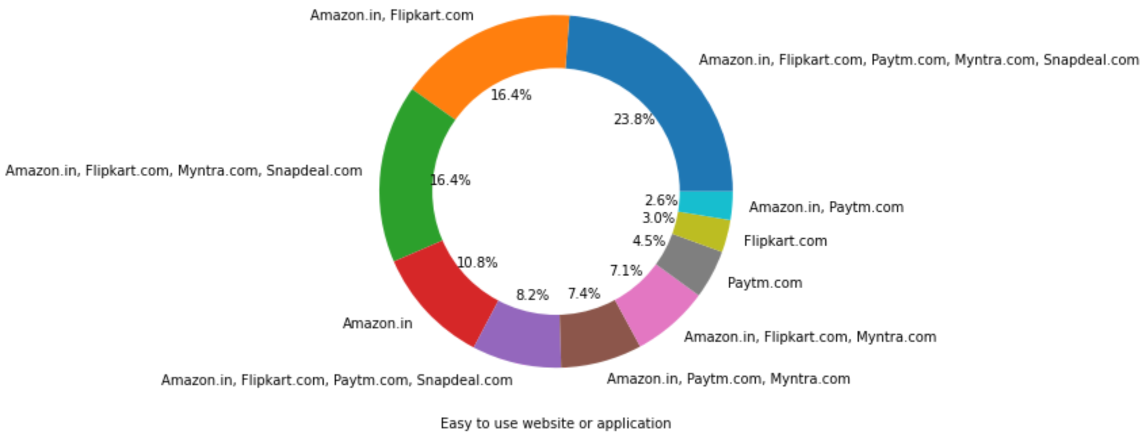
### Brand image

In [29]:

```
performance=['Easy to use website or application',
        'Visual appealing web-page layout', 'Wild variety of product on offer',
        'Complete, relevant description information of products',
        'Fast loading website speed of website and application',
        'Reliability of the website or application',
        'Quickness to complete purchase',
        'Availability of several payment options', 'Speedy order delivery',
        'Privacy of customers’ information',
        'Security of customer financial information',
        'Perceived Trustworthiness',
        'Presence of online assistance through multi-channel']
```

In [30]:

```
for i in performance:
        plt.figure(figsize=(8,6))
        df[i].value_counts().plot.pie(autopct='%1.1f%%')
        centre=plt.Circle((0,0),0.7,fc='white')
        fig=plt.gcf()
        fig.gca().add_artist(centre)
        plt.xlabel(i)
        plt.ylabel('')
        plt.figure()
```




```
<Figure size 432x288 with 0 Axes>
```

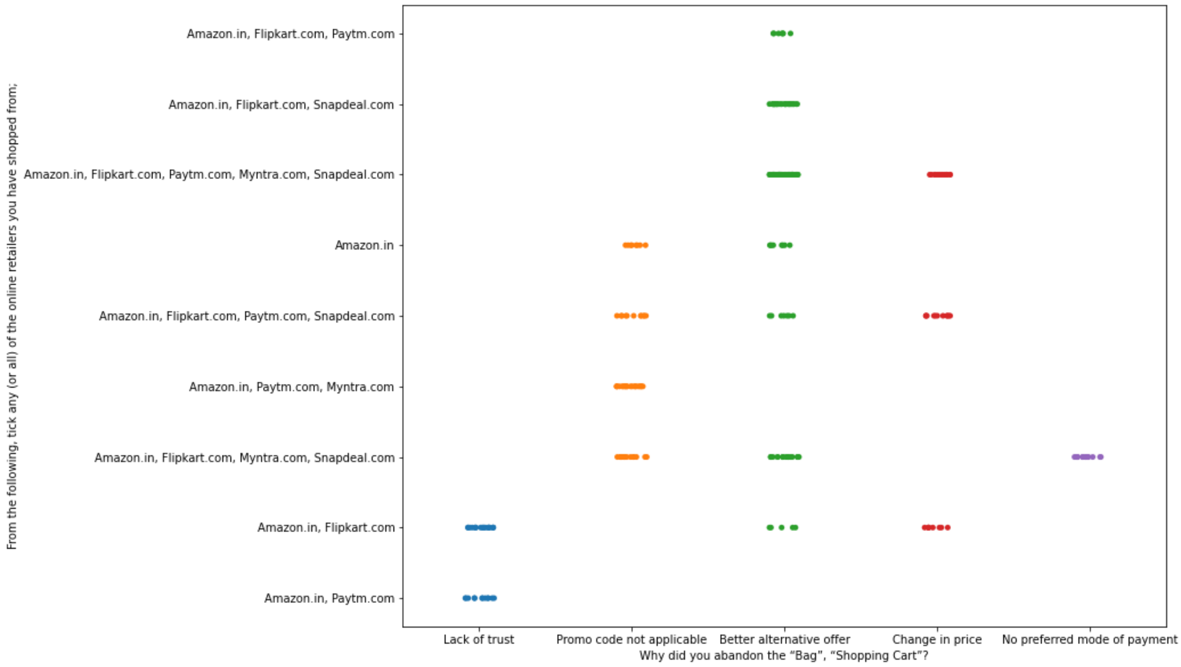

Amazon, Flipkart have been had the highest votes for having all the positive points and have maintained a very good brand image followed by paytm and the myntra.

In [31]:

```
plt.figure(figsize=(12,10))
sns.stripplot(df['Why did you abandon the “Bag”, “Shopping Cart”?'],
              df['From the following, tick any (or all) of the online retailers you hav
```

Out[31]:

```
<AxesSubplot:xlabel='Why did you abandon the “Bag”, “Shopping Cart”?', ylabe
l='From the following, tick any (or all) of the online retailers you have sh
opped from;'>
```




We can clearly see that most of the time people abandon the bag is beacuse they get a better alternative offer or promo code not applicable. There is also lack of trust seen in amazon, flipkart and paytm by some people.

### Loyalty

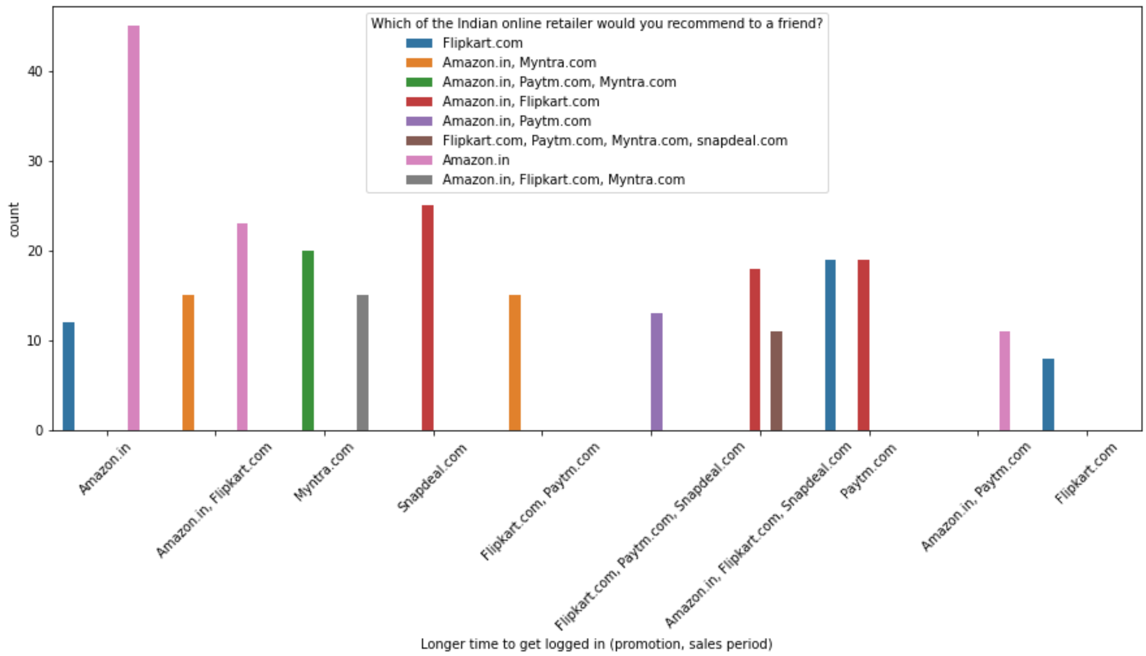
Loyal customers are those who keep using the same brand even if it is not good as other brands

In [32]:

```
#Collecting all the negative remarks about a brand
bad=['Longer time to get logged in (promotion, sales period)',
       'Longer time in displaying graphics and photos (promotion, sales period)',
       'Late declaration of price (promotion, sales period)',
       'Longer page loading time (promotion, sales period)',
       'Limited mode of payment on most products (promotion, sales period)',
       'Longer delivery period', 'Change in website/Application design',
       'Frequent disruption when moving from one page to another']
```

In [33]:

```
for i in bad:
        plt.figure(figsize=(15,6))
        sns.countplot(df[i],hue=df['Which of the Indian online retailer would you recom
        plt.xticks(rotation=45)
        plt.figure()
```




Customers seem to be more loyal to amazon, flipkart and paytm as even though many of them have given negative remarks about them still they would recommend these platforms to their friend

## Processing the dataframe

### *Separating the label from rest of the features*

In [34]:

```
x=df.copy()
x.drop('Which of the Indian online retailer would you recommend to a friend?',axis=1,ir
y=df['Which of the Indian online retailer would you recommend to a friend?']
```

### *Encoding Categorical Features*

In [35]:

```
cat=[i for i in x.columns if x[i].dtypes=='O']
```

In [36]:

```
from sklearn.preprocessing import OrdinalEncoder,LabelEncoder
encode=OrdinalEncoder()
labe=LabelEncoder()
```

In [37]:

```
#using ordinal encoder for independent features
for i in cat:
    x[i]=encode.fit_transform(x[i].values.reshape(-1,1))

#Using label encoder for Label Column
y=labe.fit_transform(y)
```

*Scaling*

In [38]:

```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
```

In [39]:

```
xd=scaler.fit_transform(x)
x=pd.DataFrame(xd,columns=x.columns)
```

### Using various feature selection method to see which feature affects the most

*Using Feature importance of random forrest*

In [40]:

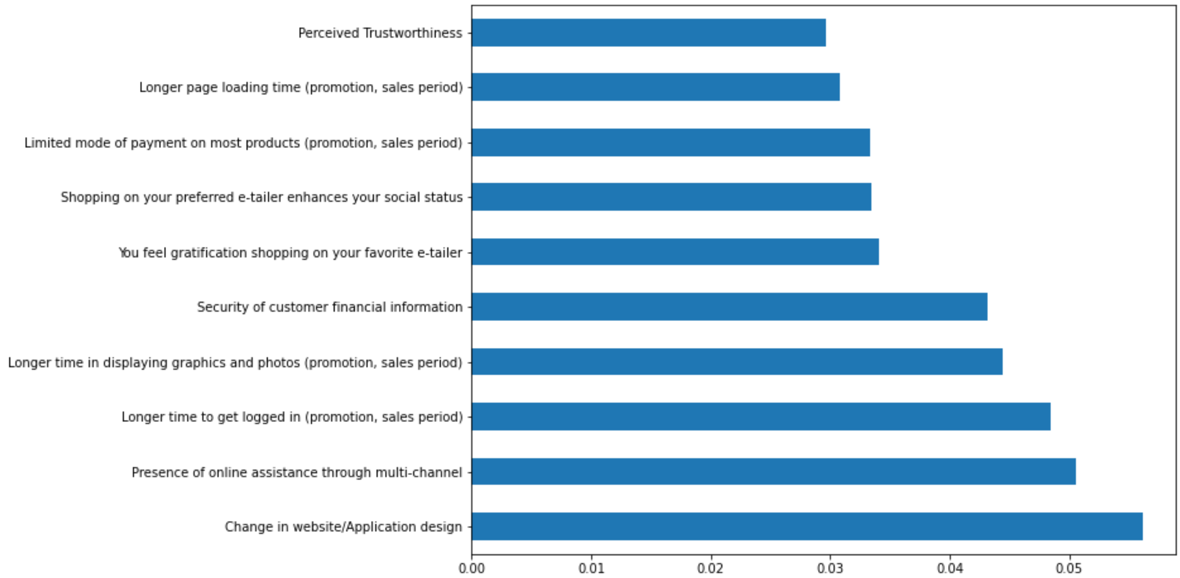
```
from sklearn.ensemble import RandomForestClassifier
m=RandomForestClassifier()
m.fit(x,y)
```

Out[40]:

```
RandomForestClassifier()
```

In [41]:

```
#plot graph of feature importances for better visualization
feat_importances = pd.Series(m.feature_importances_, index=x.columns)
plt.figure(figsize=(10,8))
feat_importances.nlargest(10).plot(kind='barh')
plt.show()
```




In the above chart we can see that above features are of most importance in determining whhich platform will a ciustomer recommend to his friend.

***Using chi2 test***

In [42]:

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
```

In [43]:

```
selection = SelectKBest(score_func=chi2)
fit = selection.fit(x,y)
```

In [44]:

```
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(x.columns)
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Features','Score']  #naming the dataframe columns
```

In [45]:

```
print(featureScores.nlargest(10,'Score'))  #print10 best features
feat=list(featureScores.nlargest(10,'Score')['Features'])
```

```
                                         Features      Score
16    Why did you abandon the “Bag”, “Shopping Cart”?  75.754028
22                        Loading and processing speed  59.810983
42  Shopping on the website gives you the sense of...  59.253569
10  What browser do you run on your device to acce...  57.171099
67             Change in website/Application design  55.301526
49                 Visual appealing web-page layout  54.245760
65  Limited mode of payment on most products (prom...  53.269266
61  Longer time to get logged in (promotion, sales...  48.222655
62  Longer time in displaying graphics and photos ...  48.130643
50                 Wild variety of product on offer  47.605973
```

# PCA

In [46]:

```
from sklearn.decomposition import PCA
pca = PCA().fit(x)
```

In [47]:

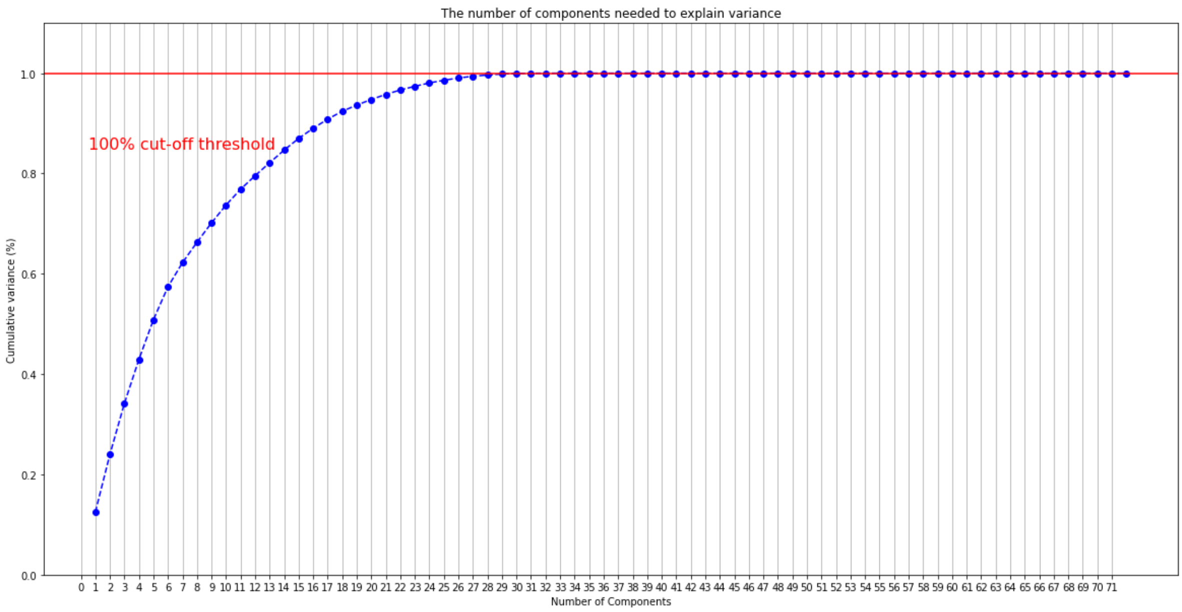
```
fig, ax = plt.subplots(figsize=(20,10))
xi = np.arange(1, 73, step=1)
yi = np.cumsum(pca.explained_variance_ratio_)

plt.ylim(0.0,1.1)
plt.plot(xi, yi, marker='o', linestyle='--', color='b')

plt.xlabel('Number of Components')
plt.xticks(np.arange(0, 72, step=1)) #change from 0-based array index to 1-based human-
plt.ylabel('Cumulative variance (%)')
plt.title('The number of components needed to explain variance')

plt.axhline(y=1, color='r', linestyle='-')
plt.text(0.5, 0.85, '100% cut-off threshold', color = 'red', fontsize=16)

ax.grid(axis='x')
plt.show()
```




```
We can clearly see that with 29 features all the information can be retained
```

In [48]:

```
pca=PCA(n_components=29)
x=pca.fit_transform(x)
x=pd.DataFrame(x)
x.head()
```

Out[48]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 2.065419 | -0.577759 | -1.030081 | -1.109784 | 0.652387 | -1.137025 | 0.699876 | -0.023177 | -0.960 |
| **1** | 0.048667 | -1.490547 | 1.081348 | 0.641617 | 0.066388 | -0.820495 | 0.072214 | -0.644870 | 0.087 |
| **2** | 1.671684 | -0.120022 | 0.775570 | -1.481374 | 0.128287 | 0.836151 | -0.793600 | 0.102789 | 0.448 |
| **3** | -0.009522 | 2.146296 | 0.753236 | -0.363176 | -1.348954 | -0.176575 | 0.567430 | -0.548924 | -0.142 |
| **4** | 0.051352 | -0.187387 | 2.386865 | 0.914150 | 0.273219 | -0.992250 | -0.511792 | 0.701105 | -0.225 |

# Modelling Phase

In [49]:

```
from sklearn.model_selection import train_test_split,cross_val_score

from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier

from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,roc_a
```

In [50]:

```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.3,random_state=7)
```

### Random Forest

In [51]:

```
model=RandomForestClassifier()
model.fit(xtrain,ytrain)
p=model.predict(xtest)
s=cross_val_score(model,x,y,cv=10)
```

In [52]:

```
print('Accuracy',np.round(accuracy_score(p,ytest),4))
print('-----------------------------------------------------------')
print('Mean of Cross Validation Score',np.round(s.mean(),4))
print('-----------------------------------------------------------')
print('Confusion Matrix')
print(confusion_matrix(p,ytest))
print('-----------------------------------------------------------')
print('Classification Report')
print(classification_report(p,ytest))
```

```
Accuracy 1.0
-----------------------------------------------------------
Mean of Cross Validation Score 0.9926
-----------------------------------------------------------
Confusion Matrix
[[26  0  0  0  0  0  0  0]
 [ 0 22  0  0  0  0  0  0]
 [ 0  0  4  0  0  0  0  0]
 [ 0  0  0  4  0  0  0  0]
 [ 0  0  0  0  5  0  0  0]
 [ 0  0  0  0  0  7  0  0]
 [ 0  0  0  0  0  0 11  0]
 [ 0  0  0  0  0  0  0  2]]
-----------------------------------------------------------
Classification Report
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        26
           1       1.00      1.00      1.00        22
           2       1.00      1.00      1.00         4
           3       1.00      1.00      1.00         4
           4       1.00      1.00      1.00         5
           5       1.00      1.00      1.00         7
           6       1.00      1.00      1.00        11
           7       1.00      1.00      1.00         2

    accuracy                           1.00        81
   macro avg       1.00      1.00      1.00        81
weighted avg       1.00      1.00      1.00        81
```

## Xgboost

In [53]:

```
model=XGBClassifier(verbosity=0)
model.fit(xtrain,ytrain)
p=model.predict(xtest)
s=cross_val_score(model,x,y,cv=10)
```

In [54]:

```
print('Accuracy',np.round(accuracy_score(p,ytest),4))
print('-----------------------------------------------------------')
print('Mean of Cross Validation Score',np.round(s.mean(),4))
print('-----------------------------------------------------------')
print('Confusion Matrix')
print(confusion_matrix(p,ytest))
print('-----------------------------------------------------------')
print('Classification Report')
print(classification_report(p,ytest))
```

```
Accuracy 1.0
-----------------------------------------------------------
Mean of Cross Validation Score 0.9926
-----------------------------------------------------------
Confusion Matrix
[[26  0  0  0  0  0  0  0]
 [ 0 22  0  0  0  0  0  0]
 [ 0  0  4  0  0  0  0  0]
 [ 0  0  0  4  0  0  0  0]
 [ 0  0  0  0  5  0  0  0]
 [ 0  0  0  0  0  7  0  0]
 [ 0  0  0  0  0  0 11  0]
 [ 0  0  0  0  0  0  0  2]]
-----------------------------------------------------------
Classification Report
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        26
           1       1.00      1.00      1.00        22
           2       1.00      1.00      1.00         4
           3       1.00      1.00      1.00         4
           4       1.00      1.00      1.00         5
           5       1.00      1.00      1.00         7
           6       1.00      1.00      1.00        11
           7       1.00      1.00      1.00         2

    accuracy                           1.00        81
   macro avg       1.00      1.00      1.00        81
weighted avg       1.00      1.00      1.00        81
```

# Hyperparameter Tuning

In [55]:

```
from sklearn.model_selection import RandomizedSearchCV
```

### Random Forest

In [56]:

```
params={'n_estimators':[100, 300, 500, 700],
        'min_samples_split':[1,2,3,4],
        'min_samples_leaf':[1,2,3,4],
        'max_depth':[None,1,2,3,4,5,6,7,8,9,10,15,20,25,30,35,40]}
```

In [57]:

```
g=RandomizedSearchCV(RandomForestClassifier(),params,cv=10)
```

In [58]:

```
g.fit(xtrain,ytrain)
```

Out[58]:

```
RandomizedSearchCV(cv=10, estimator=RandomForestClassifier(),
                   param_distributions={'max_depth': [None, 1, 2, 3, 4, 5,
6, 7,
                                                      8, 9, 10, 15, 20, 25,
30,
                                                      35, 40],
                                        'min_samples_leaf': [1, 2, 3, 4],
                                        'min_samples_split': [1, 2, 3, 4],
                                        'n_estimators': [100, 300, 500, 70
0]})
```

In [59]:

```
print(g.best_estimator_)
print(g.best_params_)
print(g.best_score_)
```

```
RandomForestClassifier(max_depth=20, min_samples_leaf=4, min_samples_split=
4,
                       n_estimators=700)
{'n_estimators': 700, 'min_samples_split': 4, 'min_samples_leaf': 4, 'max_de
pth': 20}
0.9947368421052631
```

In [65]:

```
m=RandomForestClassifier(max_depth=20, min_samples_leaf=4, min_samples_split=4,n_estima
m.fit(xtrain,ytrain)
p=m.predict(xtest)
score=cross_val_score(m,x,y,cv=10)
```

In [66]:

```
print('Accuracy',np.round(accuracy_score(p,ytest),4))
print('-----------------------------------------------------------')
print('Mean of Cross Validation Score',np.round(s.mean(),4))
print('-----------------------------------------------------------')
print('Confusion Matrix')
print(confusion_matrix(p,ytest))
print('-----------------------------------------------------------')
print('Classification Report')
print(classification_report(p,ytest))
```

```
Accuracy 1.0
-----------------------------------------------------------
Mean of Cross Validation Score 0.9926
-----------------------------------------------------------
Confusion Matrix
[[26  0  0  0  0  0  0  0]
 [ 0 22  0  0  0  0  0  0]
 [ 0  0  4  0  0  0  0  0]
 [ 0  0  0  4  0  0  0  0]
 [ 0  0  0  0  5  0  0  0]
 [ 0  0  0  0  0  7  0  0]
 [ 0  0  0  0  0  0 11  0]
 [ 0  0  0  0  0  0  0  2]]
-----------------------------------------------------------
Classification Report
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        26
           1       1.00      1.00      1.00        22
           2       1.00      1.00      1.00         4
           3       1.00      1.00      1.00         4
           4       1.00      1.00      1.00         5
           5       1.00      1.00      1.00         7
           6       1.00      1.00      1.00        11
           7       1.00      1.00      1.00         2

    accuracy                           1.00        81
   macro avg       1.00      1.00      1.00        81
weighted avg       1.00      1.00      1.00        81
```

## Xgboost

In [67]:

```
params={'n_estimators':[100,200,300,400,500],
      'learning_rate':[0.001,0.01,0.10,],
      'subsample':[0.5,1],
      'max_depth':[1,2,3,4,5,6,7,8,9,10]}
```

In [68]:

```
g=RandomizedSearchCV(XGBClassifier(),params,cv=10)
```

In [69]:

```
g.fit(xtrain,ytrain)
```

Out[69]:

```
RandomizedSearchCV(cv=10,
                   estimator=XGBClassifier(base_score=None, booster=None,
                                           colsample_bylevel=None,
                                           colsample_bynode=None,
                                           colsample_bytree=None, gamma=Non
e,
                                           gpu_id=None, importance_type='gai
n',
                                           interaction_constraints=None,
                                           learning_rate=None,
                                           max_delta_step=None, max_depth=No
ne,
                                           min_child_weight=None, missing=na
n,
                                           monotone_constraints=None,
                                           n_estimators=100, n_jobs=None,
                                           num_parallel_tree=None,
                                           random_state=None, reg_alpha=Non
e,
                                           reg_lambda=None,
                                           scale_pos_weight=None,
                                           subsample=None, tree_method=None,
                                           validate_parameters=None,
                                           verbosity=None),
                   param_distributions={'learning_rate': [0.001, 0.01, 0.1],
                                        'max_depth': [1, 2, 3, 4, 5, 6, 7,
8, 9,
                                                      10],
                                        'n_estimators': [100, 200, 300, 400,
                                                         500],
                                        'subsample': [0.5, 1]})
```

In [70]:

```
print(g.best_estimator_)
print(g.best_params_)
print(g.best_score_)
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.1, max_delta_step=0, max_depth=10,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=500, n_jobs=4, num_parallel_tree=1,
              objective='multi:softprob', random_state=0, reg_alpha=0,
              reg_lambda=1, scale_pos_weight=None, subsample=0.5,
              tree_method='exact', validate_parameters=1, verbosity=None)
{'subsample': 0.5, 'n_estimators': 500, 'max_depth': 10, 'learning_rate': 0.
1}
0.9894736842105264
```

In [71]:

```
m=XGBClassifier(max_depth=10,learning_rate=0.1,n_estimators=500,subsample=0.5)
m.fit(xtrain,ytrain)
p=m.predict(xtest)
score=cross_val_score(m,x,y,cv=10)
```

In [72]:

```
print('Accuracy',np.round(accuracy_score(p,ytest),4))
print('----------------------------------------------------------')
print('Mean of Cross Validation Score',np.round(s.mean(),4))
print('----------------------------------------------------------')
print('Confusion Matrix')
print(confusion_matrix(p,ytest))
print('----------------------------------------------------------')
print('Classification Report')
print(classification_report(p,ytest))
```

```
Accuracy 1.0
----------------------------------------------------------
Mean of Cross Validation Score 0.9926
----------------------------------------------------------
Confusion Matrix
[[26  0  0  0  0  0  0  0]
 [ 0 22  0  0  0  0  0  0]
 [ 0  0  4  0  0  0  0  0]
 [ 0  0  0  4  0  0  0  0]
 [ 0  0  0  0  5  0  0  0]
 [ 0  0  0  0  0  7  0  0]
 [ 0  0  0  0  0  0 11  0]
 [ 0  0  0  0  0  0  0  2]]
----------------------------------------------------------
Classification Report
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        26
           1       1.00      1.00      1.00        22
           2       1.00      1.00      1.00         4
           3       1.00      1.00      1.00         4
           4       1.00      1.00      1.00         5
           5       1.00      1.00      1.00         7
           6       1.00      1.00      1.00        11
           7       1.00      1.00      1.00         2

    accuracy                           1.00        81
   macro avg       1.00      1.00      1.00        81
weighted avg       1.00      1.00      1.00        81
```

***Conclusion***

Both the models give accurate and equal results so we choose xgboost as or final model because of its quick speed.

# Finalizing the best Model

In [73]:

```
model=XGBClassifier(max_depth=2,learning_rate=0.01,n_estimators=500,subsample=1)
model.fit(xtrain,ytrain)
p=model.predict(xtest)
score=cross_val_score(model,x,y,cv=10)
```

# Evaluation Metrics

In [74]:

```
print('Accuracy',np.round(accuracy_score(p,ytest),4))
print('----------------------------------------------------------')
print('Mean of Cross Validation Score',np.round(s.mean(),4))
print('----------------------------------------------------------')
print('Confusion Matrix')
print(confusion_matrix(p,ytest))
print('----------------------------------------------------------')
print('Classification Report')
print(classification_report(p,ytest))
```

```
Accuracy 1.0
----------------------------------------------------------
Mean of Cross Validation Score 0.9926
----------------------------------------------------------
Confusion Matrix
[[26  0  0  0  0  0  0  0]
 [ 0 22  0  0  0  0  0  0]
 [ 0  0  4  0  0  0  0  0]
 [ 0  0  0  4  0  0  0  0]
 [ 0  0  0  0  5  0  0  0]
 [ 0  0  0  0  0  7  0  0]
 [ 0  0  0  0  0  0 11  0]
 [ 0  0  0  0  0  0  0  2]]
----------------------------------------------------------
Classification Report
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        26
           1       1.00      1.00      1.00        22
           2       1.00      1.00      1.00         4
           3       1.00      1.00      1.00         4
           4       1.00      1.00      1.00         5
           5       1.00      1.00      1.00         7
           6       1.00      1.00      1.00        11
           7       1.00      1.00      1.00         2

    accuracy                           1.00        81
   macro avg       1.00      1.00      1.00        81
weighted avg       1.00      1.00      1.00        81
```

# Saving the Model

In [75]:

```
import joblib
joblib.dump(model,'Retention.obj')
```

Out[75]:

```
['Retention.obj']
```

# Conclusion

The results of this study suggest following outputs which might be useful for E-commerce websites to extend
their business

1. The cost of the product, the reliability of the E-commerce company and the return policies all play an equally important role in deciding the buying behaviour of online customers. The cost is an important factor as it was the basic criteria used by online retailers to attract customers. The reliability of the E-commerce company is also important, as it is even required in offline retail. It is important because customers are paying online, so they need to be sure of security of the online transaction. The return policies are important because in online retail customer does not get to feel the product. Thus, he wants to be sure that it will be possible to return the product if he does not like it in real. Whereas, the logistics factor, which included Cash on delivery option, One day delivery and the quality of packaging plays a secondary role in this process though these are Must-be-quality. This is so because these all does not interfere with the real product and people believe that this is the basic value that E-commerce websites provide.


2. All the websites were not equally preferred by online customers. Amazon was the most preferred followed by Flipkart. This can be explained easily by previous result that we got. These two companies are most trusted in the industry and hence, have a huge reliability. Also, the sellers listed on these websites are generally from Tier 1 cities as compared to Snapdeal and PayTM which have more sellers from tier 2 and 3 cities. Also, these websites have the most lenient return policies as compared to others and also the time required to process a return is low for these.

In [ ]:

```

```