Importing Libraries

```
import pandas as pd                  # for analys part and manuplation
import matplotlib.pyplot as plt      # used to create plots
import seaborn as sns                # for data visualization and
exploratory
import numpy as np                   # to perform mathematical operation
```

Importing the data

```
df = pd.read_csv('sales_data.csv')
```

Checking the shape (Total number of records & attributes in the dataset)

```
df.shape
```

```
(40000, 15)
```

In this dataset there are 40000 records and 15 attributes.

Duplication of values in the dataset

```
df.duplicated().sum()
```

```
55
```

There are 55 duplicate values in this dataset.

Drop Duplicate Values from the dataset

```
df=df.drop_duplicates()
```

Recheck Duplicate Values

```
df.duplicated().sum()
```

```
0
```

After using the command to drop duplicates values. Now the sum of duplicates values is 0.

Check Null values in the Dataset

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

```
18106
```

In this dataset there are 18106 null values.

```
df['marriage'].fillna(0,inplace=True)
df['house_owner'].fillna(0,inplace=True)
```

Here all dulplicate values are filled by 0.

```
df.head(5)
```

```
  flag gender         education  house_val     age online customer_psy  \
0    Y      M          4. Grad     756460    1_Unk      N            B

1    N      F          3. Bach     213171    7_>65      N            E

2    N      M   2. Some College     111147   2_<=25      Y            C

3    Y      M   2. Some College     354151   2_<=25      Y            B

4    Y      F   2. Some College     117087    1_Unk      Y            J


  marriage child     occupation mortgage house_owner     region  car_prob  \
0        0     U   Professional     1Low           0    Midwest         1
1        0     U   Professional     1Low       Owner  Northeast         3
2        0     Y   Professional     1Low       Owner    Midwest         1
3   Single     U  Sales/Service     1Low           0       West         2
4  Married     Y  Sales/Service     1Low           0      South         7

  fam_income
0          L
1          G
2          J
3          L
4          H
```

Head command is used to see the top 5 records of dataset.

#Sorting the null values in ascending Order

```
df.isnull().sum().sort_values(ascending=True)
```

```
flag                 0
gender               0
house_val            0
```

```
age               0
online            0
customer_psy      0
marriage          0
child             0
occupation        0
mortgage          0
house_owner       0
region            0
car_prob          0
fam_income        0
education       735
dtype: int64
```

Call top 10 records from the dataset

```
df.head(10)
```

```
  flag gender        education  house_val     age online customer_psy
\
0    Y      M          4. Grad     756460   1_Unk      N            B

1    N      F          3. Bach     213171   7_>65      N            E

2    N      M  2. Some College     111147  2_<=25      Y            C

3    Y      M  2. Some College     354151  2_<=25      Y            B

4    Y      F  2. Some College     117087   1_Unk      Y            J

5    Y      F          3. Bach     248694  6_<=65      Y            B

6    Y      M          3. Bach    2000000   1_Unk      Y            A

7    N      F          3. Bach     416925  5_<=55      Y            C

8    N      F            1. HS     207676  4_<=45      Y            G

9    Y      M            1. HS     241380   1_Unk      Y            C


  marriage child     occupation mortgage house_owner     region
car_prob  \
0        0     U   Professional     1Low           0    Midwest
1
1        0     U   Professional     1Low       Owner  Northeast
3
2        0     Y   Professional     1Low       Owner    Midwest
1
```

```
3   Single     U  Sales/Service     1Low          0       West
2
4  Married     Y  Sales/Service     1Low          0      South
7
5  Married     N   Professional     2Med      Owner       West
1
6  Married     U   Professional     1Low          0  Northeast
5
7  Married     Y   Professional     1Low      Owner      South
2
8        0     Y    Blue Collar     1Low     Renter       West
5
9  Married     U  Sales/Service     1Low          0  Northeast
6

  fam_income
0          L
1          G
2          J
3          L
4          H
5          G
6          C
7          I
8          D
9          G
```

Transposed of records & attributes

```
df.head().T

                         0             1                2
3  \
flag                     Y             N                N
Y
gender                   M             F                M
M
education          4. Grad       3. Bach  2. Some College  2. Some
College
house_val           756460        213171           111147
354151
age                  1_Unk         7_>65           2_<=25
2_<=25
online                   N             N                Y
Y
customer_psy             B             E                C
B
marriage                 0             0                0
Single
child                    U             U                Y
```

```
U
occupation    Professional  Professional     Professional
Sales/Service
mortgage              1Low          1Low             1Low
1Low
house_owner              0         Owner            Owner
0
region             Midwest     Northeast          Midwest
West
car_prob                 1             3                1
2
fam_income               L             G                J
L

                            4
flag                        Y
gender                      F
education    2. Some College
house_val              117087
age                     1_Unk
online                      Y
customer_psy                J
marriage              Married
child                       Y
occupation      Sales/Service
mortgage                 1Low
house_owner                 0
region                  South
car_prob                    7
fam_income                  H
```

This tranposed command used to transposed the records and attributes.

Call 10 bottom records from the dataset

```
df.tail(10)
```

```
      flag gender education  house_val     age online customer_psy
marriage  \
39990    N      F     1. HS     120630   1_Unk      Y            F
Single
39991    N      M    0. <HS      88682   1_Unk      N            G
Married
39992    N      F   4. Grad     256498  5_<=55      Y            E
Married
39993    N      M     1. HS          0   7_>65      Y            C
0
39994    Y      M   4. Grad     603554  5_<=55      Y            C
Married
39995    Y      F   3. Bach          0   7_>65      Y            C
```

```
0
39996    N      F      1. HS     213596  4_<=45      N            I
Married
39997    Y      M    0. <HS     134070  3_<=35      Y            F
Married
39998    N      M      1. HS     402210   7_>65      Y            E
0
39999    N      F   3. Bach     836030   7_>65      Y            B
Married

      child     occupation mortgage house_owner     region
car_prob  \
39990     U  Sales/Service     1Low       Owner    Midwest         7

39991     Y    Blue Collar     1Low      Renter       West         9

39992     N    Blue Collar     1Low       Owner      South         3

39993     Y   Professional     1Low       Owner  Northeast         2

39994     Y   Professional    3High       Owner       West         2

39995     U        Retired     1Low           0      South         3

39996     U    Blue Collar     1Low       Owner      South         1

39997     U  Sales/Service     1Low       Owner    Midwest         4

39998     Y  Sales/Service     1Low           0       West         2

39999     N        Retired     2Med       Owner  Northeast         1


      fam_income
39990          H
39991          D
39992          E
39993          G
39994          J
39995          F
39996          D
39997          E
39998          B
39999          J
```

```
#df.head().T.to_csv("head1.xls")
```

Summary of dataset

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 39945 entries, 0 to 39999
Data columns (total 15 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   flag          39945 non-null  object
 1   gender        39945 non-null  object
 2   education     39210 non-null  object
 3   house_val     39945 non-null  int64
 4   age           39945 non-null  object
 5   online        39945 non-null  object
 6   customer_psy  39945 non-null  object
 7   marriage      39945 non-null  object
 8   child         39945 non-null  object
 9   occupation    39945 non-null  object
 10  mortgage      39945 non-null  object
 11  house_owner   39945 non-null  object
 12  region        39945 non-null  object
 13  car_prob      39945 non-null  int64
 14  fam_income    39945 non-null  object
dtypes: int64(2), object(13)
memory usage: 4.9+ MB
```

In info command used to check number of attibutes there name and there data type.

Check there is any extreme values

```
df.describe()
```

```
          house_val      car_prob
count  3.994500e+04  39945.000000
mean   3.075966e+05      3.486719
std    4.223422e+05      2.573636
min    0.000000e+00      0.000000
25%    8.145500e+04      1.000000
50%    2.151330e+05      3.000000
75%    3.940670e+05      5.000000
max    9.999999e+06      9.000000
```

In this dataset there 2 attributes have integer values.So, describes command explain there five number summary.

```
 round(df.describe().T,0)
```

```
             count      mean       std  min      25%       50%
75%  \
house_val  39945.0  307597.0  422342.0  0.0  81455.0  215133.0
```

```
394067.0
car_prob   39945.0       3.0       3.0  0.0      1.0       3.0
5.0

                 max
house_val  9999999.0
car_prob         9.0
```

Check Columns Name

```
df.columns
```

```
Index(['flag', 'gender', 'education', 'house_val', 'age', 'online',
       'customer_psy', 'marriage', 'child', 'occupation', 'mortgage',
       'house_owner', 'region', 'car_prob', 'fam_income'],
      dtype='object')
```

Renaming column name

```
df.rename(columns=
{'car_prob':'car_probability','customer_psy':'customer_psychology','ho
use_val': 'house_value','fam_income':'family_income'}, inplace= True)

df
```

```
      flag gender        education  house_value     age online  \
0        Y      M          4. Grad       756460   1_Unk      N
1        N      F          3. Bach       213171   7_>65      N
2        N      M  2. Some College       111147  2_<=25      Y
3        Y      M  2. Some College       354151  2_<=25      Y
4        Y      F  2. Some College       117087   1_Unk      Y
...    ...    ...              ...          ...     ...    ...
39995    Y      F          3. Bach            0   7_>65      Y
39996    N      F            1. HS       213596  4_<=45      N
39997    Y      M            0. <HS      134070  3_<=35      Y
39998    N      M            1. HS       402210   7_>65      Y
39999    N      F          3. Bach       836030   7_>65      Y

      customer_psychology marriage child     occupation mortgage
house_owner  \
0                       B        0     U   Professional     1Low
0
1                       E        0     U   Professional     1Low
Owner
2                       C        0     Y   Professional     1Low
Owner
3                       B   Single     U  Sales/Service     1Low
0
4                       J  Married     Y  Sales/Service     1Low
0
```

```
...                      ...       ...   ...            ...       ...
...
39995                      C         0     U        Retired      1Low
0
39996                      I   Married     U    Blue Collar      1Low
Owner
39997                      F   Married     U  Sales/Service      1Low
Owner
39998                      E         0     Y  Sales/Service      1Low
0
39999                      B   Married     N        Retired      2Med
Owner

          region  car_probability family_income
0        Midwest                1             L
1      Northeast                3             G
2        Midwest                1             J
3           West                2             L
4          South                7             H
...          ...              ...           ...
39995      South                3             F
39996      South                1             D
39997    Midwest                4             E
39998       West                2             B
39999  Northeast                1             J

[39945 rows x 15 columns]
```

```
df.nunique().sort_values(ascending=False)
```

```
house_value           19572
family_income            13
customer_psychology      11
car_probability          10
age                       7
occupation                6
education                 5
region                    5
child                     4
gender                    3
marriage                  3
mortgage                  3
house_owner               3
flag                      2
online                    2
dtype: int64
```

```
print ("car_probability:",df.car_probability.unique())
print("mortgage",df.mortgage.unique())
print("flag",df.flag.unique())
print("age",df.age.unique())
```

```
car_probability: [1 3 2 7 5 6 9 8 4 0]
mortgage ['1Low' '2Med' '3High']
flag ['Y' 'N']
age ['1_Unk' '7_>65' '2_<=25' '6_<=65' '5_<=55' '4_<=45' '3_<=35']
```

This command is used to check the unique values corresponding to each attribute.

Check data types

```
df.dtypes
```

```
flag                    object
gender                  object
education               object
house_value              int64
age                     object
online                  object
customer_psychology     object
marriage                object
child                   object
occupation              object
mortgage                object
house_owner             object
region                  object
car_probability          int64
family_income           object
dtype: object
```

Changing the data type from character to category

```
df.nunique()
```

```
flag                       2
gender                     3
education                  5
house_value            19572
age                        7
online                     2
customer_psychology       11
marriage                   3
child                      4
occupation                 6
mortgage                   3
house_owner                3
region                     5
car_probability           10
family_income             13
dtype: int64
```

```
df["flag"] = df["flag"].astype("category")
df["gender"] = df["gender"].astype("category")
df["education"] = df["education"].astype("category")
df["age"] = df["age"].astype("string")
df["online"] = df["online"].astype("category")
df["customer_psychology "] =
df["customer_psychology"].astype("category")
df["marriage"] = df["marriage"].astype("category")
df["child"] = df["child"].astype("category")
df["occupation"] = df["occupation"].astype("category")
df["mortgage"] = df["mortgage"].astype("category")
df["house_owner"] = df["house_owner"].astype("category")
df["child"] = df["child"].astype("category")
df['region']= df['region'].astype("category")
df["occupation"] = df["occupation"].astype("category")
df["family_income"] = df ["family_income"].astype("category")
```

```
df.dtypes
```

```
flag                        category
gender                      category
education                   category
house_value                    int64
age                           string
online                      category
customer_psychology           object
marriage                    category
child                       category
occupation                  category
mortgage                    category
house_owner                 category
region                      category
car_probability                int64
family_income               category
customer_psychology         category
dtype: object
```

Deleted Columns from Dataset

```
df.drop(['education','house_owner','child'],axis=1,inplace=True)
df
```

```
      flag gender  house_value     age online customer_psychology
marriage  \
0        Y      M       756460    1_Unk      N                   B
0
1        N      F       213171    7_>65      N                   E
0
2        N      M       111147   2_<=25      Y                   C
0
```

```
3        Y      M       354151  2_<=25      Y                   B
Single
4        Y      F       117087   1_Unk      Y                   J
Married
...    ...    ...          ...     ...    ...                 ...
...
39995    Y      F            0   7_>65      Y                   C
0
39996    N      F       213596  4_<=45      N                   I
Married
39997    Y      M       134070  3_<=35      Y                   F
Married
39998    N      M       402210   7_>65      Y                   E
0
39999    N      F       836030   7_>65      Y                   B
Married

          occupation mortgage     region  car_probability
family_income  \
0       Professional     1Low    Midwest                1
L
1       Professional     1Low  Northeast                3
G
2       Professional     1Low    Midwest                1
J
3      Sales/Service     1Low       West                2
L
4      Sales/Service     1Low      South                7
H
...              ...      ...        ...              ...             ..
.
39995        Retired     1Low      South                3
F
39996    Blue Collar     1Low      South                1
D
39997  Sales/Service     1Low    Midwest                4
E
39998  Sales/Service     1Low       West                2
B
39999        Retired     2Med  Northeast                1
J

      customer_psychology
0                       B
1                       E
2                       C
3                       B
4                       J
...                   ...
39995                   C
```

```
39996                   I
39997                   F
39998                   E
39999                   B

[39945 rows x 13 columns]
```

```
df["region"].unique()
```

```
['Midwest', 'Northeast', 'West', 'South', 'Rest']
Categories (5, object): ['Midwest', 'Northeast', 'Rest', 'South',
'West']
```

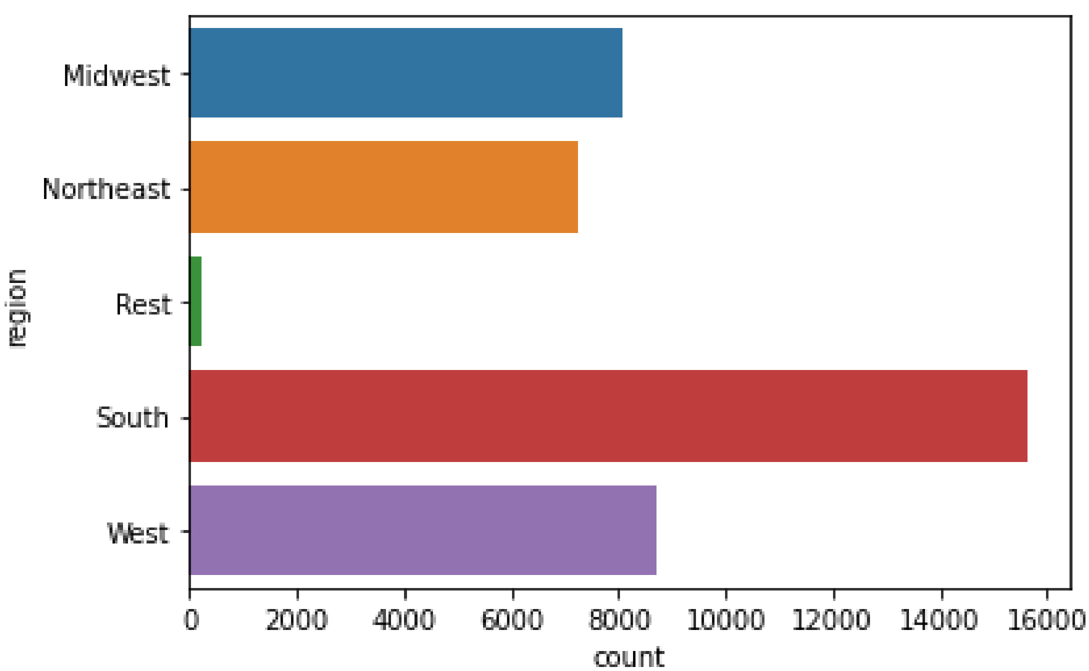
```
df["region"].value_counts()
```

```
South        15652
West          8717
Midwest       8097
Northeast     7234
Rest           245
Name: region, dtype: int64
```

Data Visualization

```
# Customer with region Visualization
sns.countplot(y='region',data=df)
```

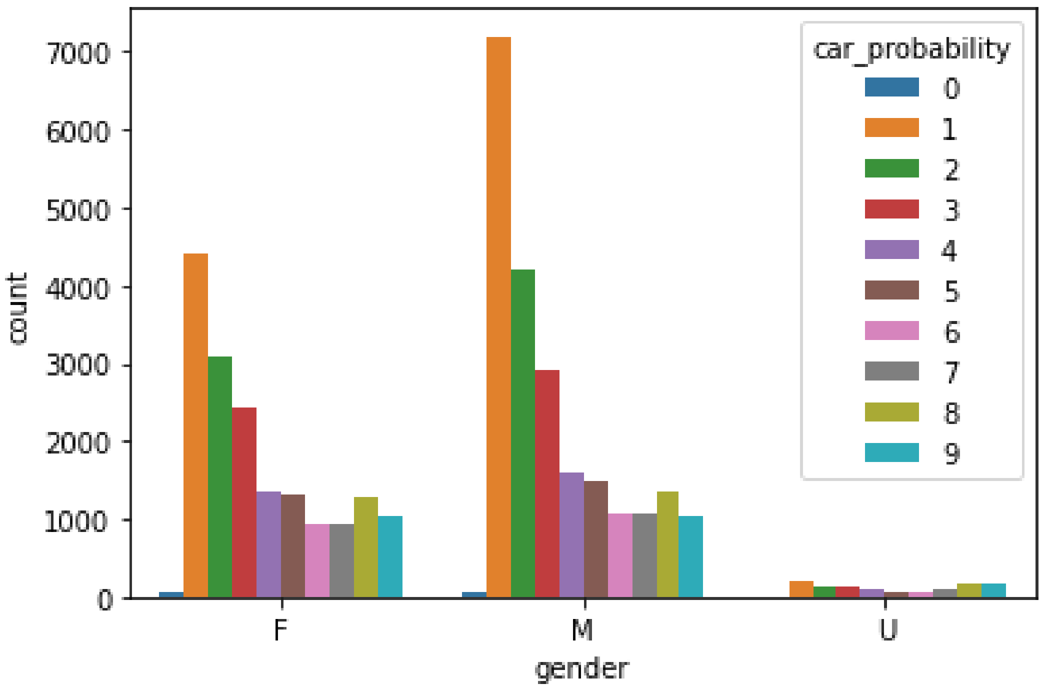
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2078743690>
```



This bar graph represent the information cross ponding to customers count from different region. The highest count of customer belongs to south region that is 15676. The count of customers from midwest and west almost same with small difference that is 8107 from midwest and 8725 from west. The count of rest of region is 245.

```
sns.countplot(x='gender', hue = 'car_probability', data = df)
#plt.savefig('Car_prob Vs gender.jpg')
```

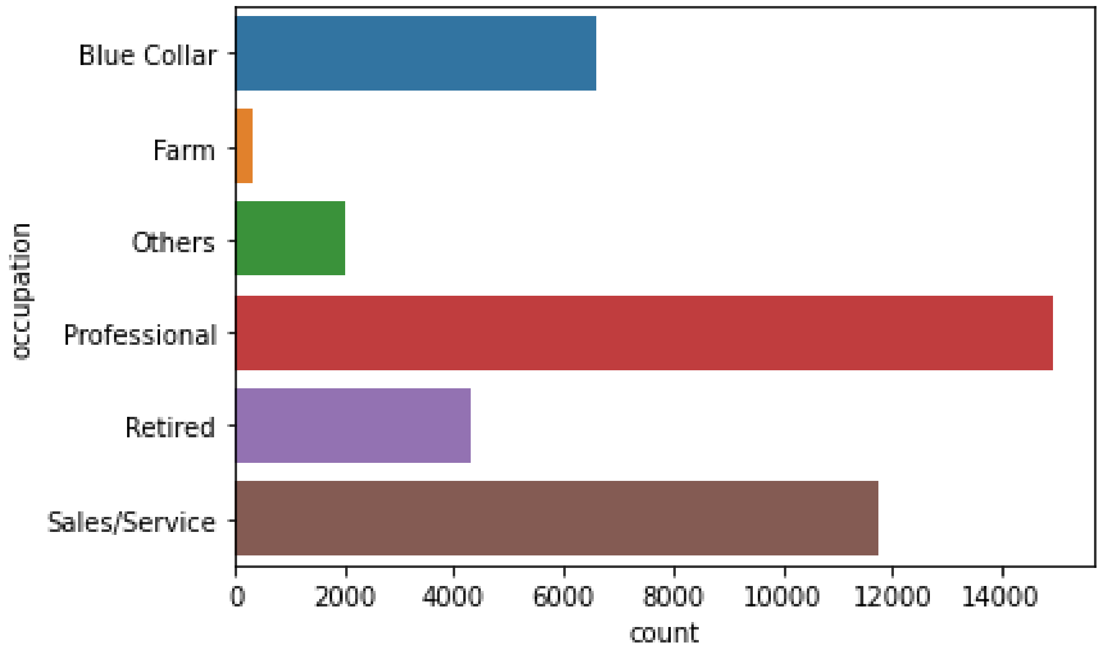
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f20784c5a90>
```



This bar graph shows the information gender cross ponding car probability. This graph shows the count of female,male cross ponding to the count of car probability. As the car probability of 1 in both female,male is highest.And the probability of 4 to 9 cars in female,male is almost same.There are some unknown values in gender attribute the count of car probability is low as compared to male and female.

```
# Customer with occupation Visualization
sns.countplot(y='occupation',data=df)
```

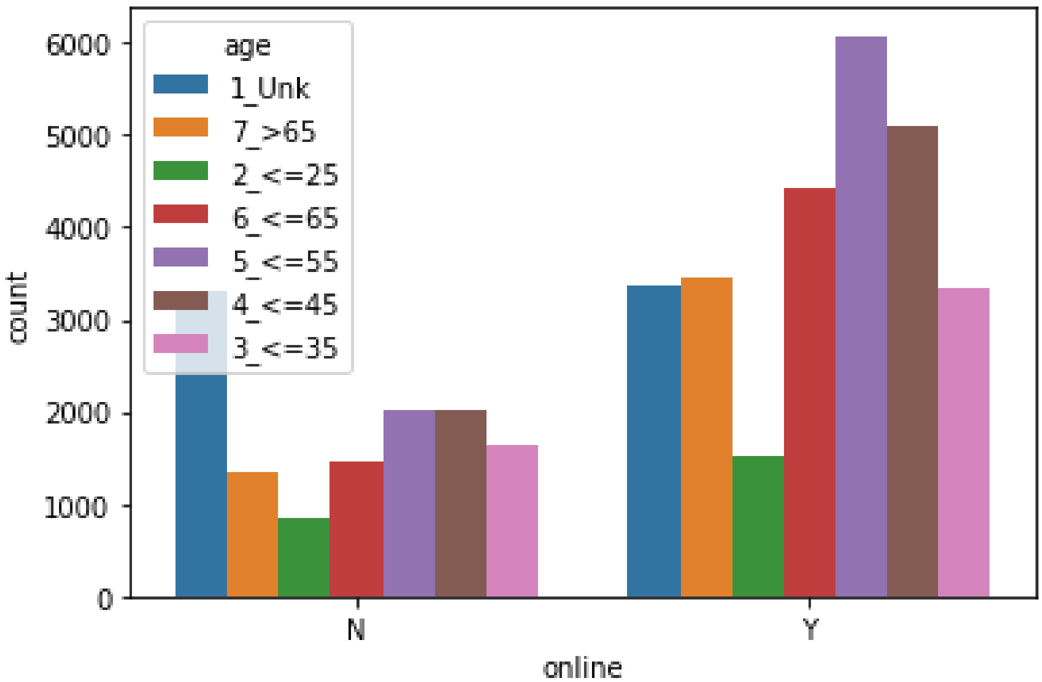
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2077f7c850>
```



This graph represent the information of customers occupation in different fields. As the highest count of customers count lies in the professional category that is 15000. the sales/service category is second highest category with 11500 count of customers. In blue collar the count of customer is 6300 and the count of customers in in farm and others is lowest as compared to other four categories.

```
sns.countplot(x='online', hue = 'age', data = df)
#plt.savefig('online Vs age.jpg')
```

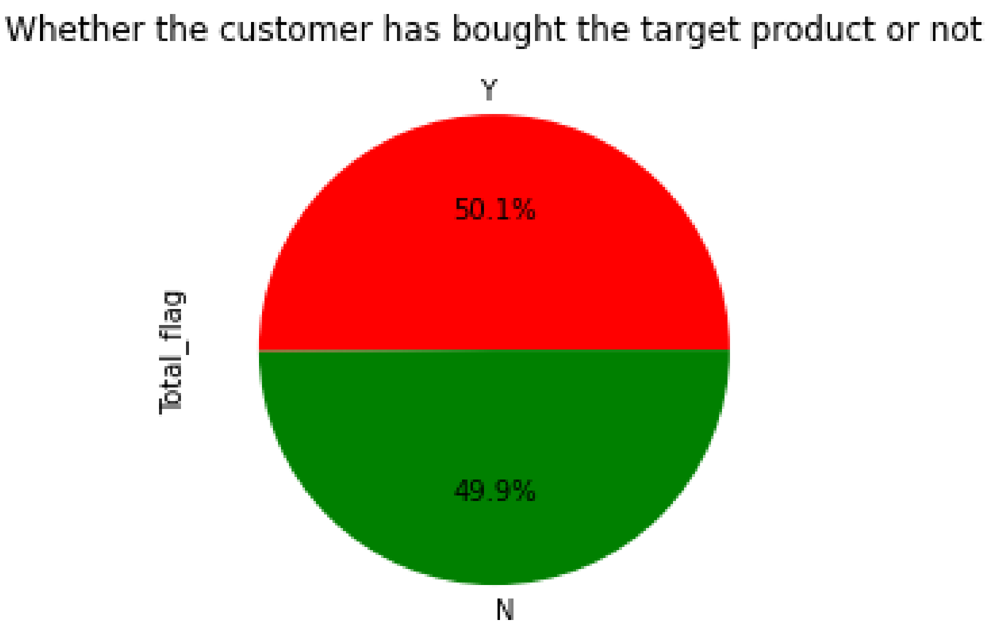
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2077ef0d10>
```



This bar graph showes the information about how many customers have online shopping experience and how many do not have online shopping experience according to their age group. It is also seen that more customers have online shopping experience.

1. Find out how many customer has purchased the target product and how many do not buy their target product?

```
color=['red','green']
df= pd.DataFrame({'Total_flag': [19644,19566]},
                 index=['Y', 'N'])

df

   Total_flag
Y       19644
N       19566

df['Total_flag'].plot.pie(colors=color,autopct='%1.1f%%')
plt.title('Whether the customer has bought the target product or not')
#plt.savefig('Whether the customer has bought the target product or
not.jpg')
plt.show()
```



In this pie chart Y shows that the percentage of customer who bought their target product and N shows that the percentage of customer who do not purchase their target products. It is clear from the pie chart that the percentage of both type customer who purchase their order product or who do not buy their products are almost same with small difference. As 50.1% customer received their order products and 49.9% do not bought their target products. So, from this percentage corresponding to both type customer its is predicted that company does not have more profit. As half of the count of total customer do not satisfy from the service of company because they do not buy their target product which they want to buy.

2. Find out how many consumers have online shopping experience and how many do not have online experience?

```
color=['skyblue','navy']
df= pd.DataFrame({'Total_online': [18139,17839]},
                 index=['Y', 'N'])

df

   Total_online
Y         18139
N         17839

df['Total_online'].plot.pie(colors=color,autopct='%1.1f%%')
plt.title('Whether the customer has online shopping experience or not')
#plt.savefig('Whether the customer has online shopping experience or not.jpg')
plt.show()
```

In this dataset we find the consumers who have online shopping experience which are denoted by Y and who do not have online experience those are denoted by N by plotting the pie chart. With the help of pie chart, it is found that 50.4% customers have online shopping experience and customers with the percentage of 49.6% do not have online shopping experience in this dataset. So, the number of counts of both customers is almost equal.

3. Find out total number of male and female in the dataset by descriptive analysis?

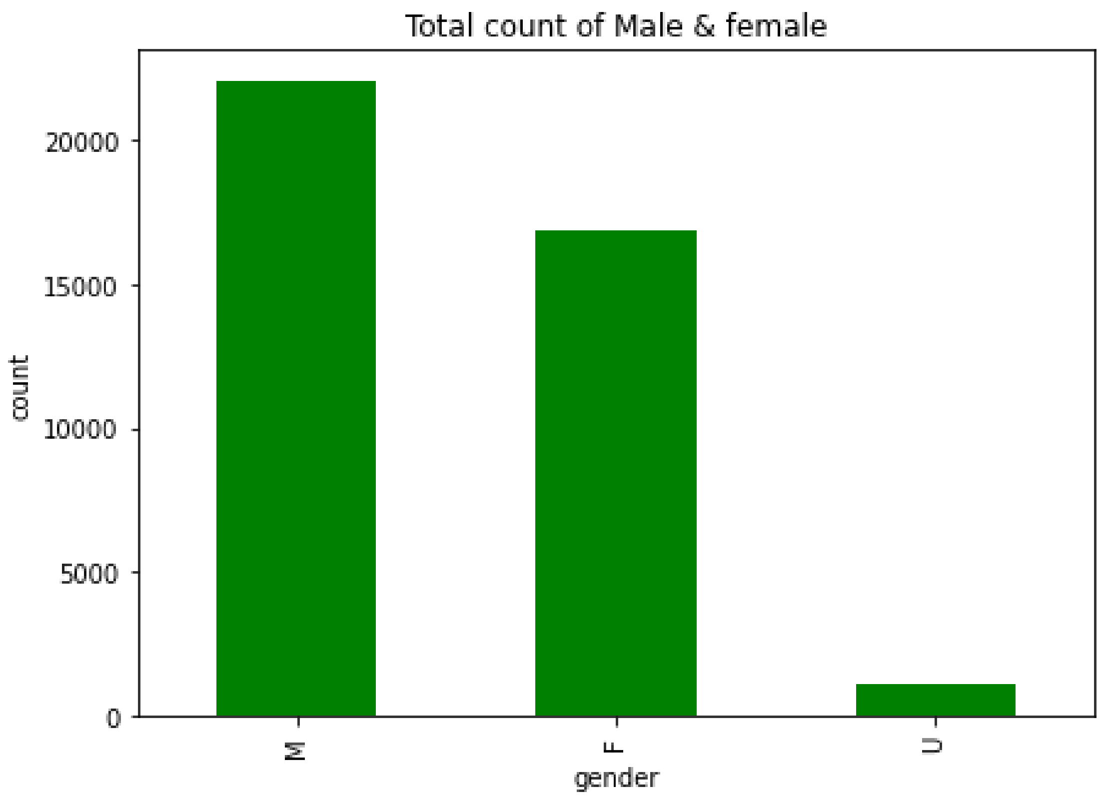
```
Gender= df['gender']
Gender.value_counts()
```

```
M    22019
F    16830
U     1151
Name: gender, dtype: int64
```

```
df['gender'].value_counts().max()
```

```
22019
```

```
x=['M','F','U']
plt.xlabel("gender")
plt.ylabel("count")
plt.title('Total count of Male & female')
Gender.value_counts().plot(kind='bar',color='green', figsize=(7,5))
plt.savefig('Total count of Male & female.jpg')
plt.show()
```

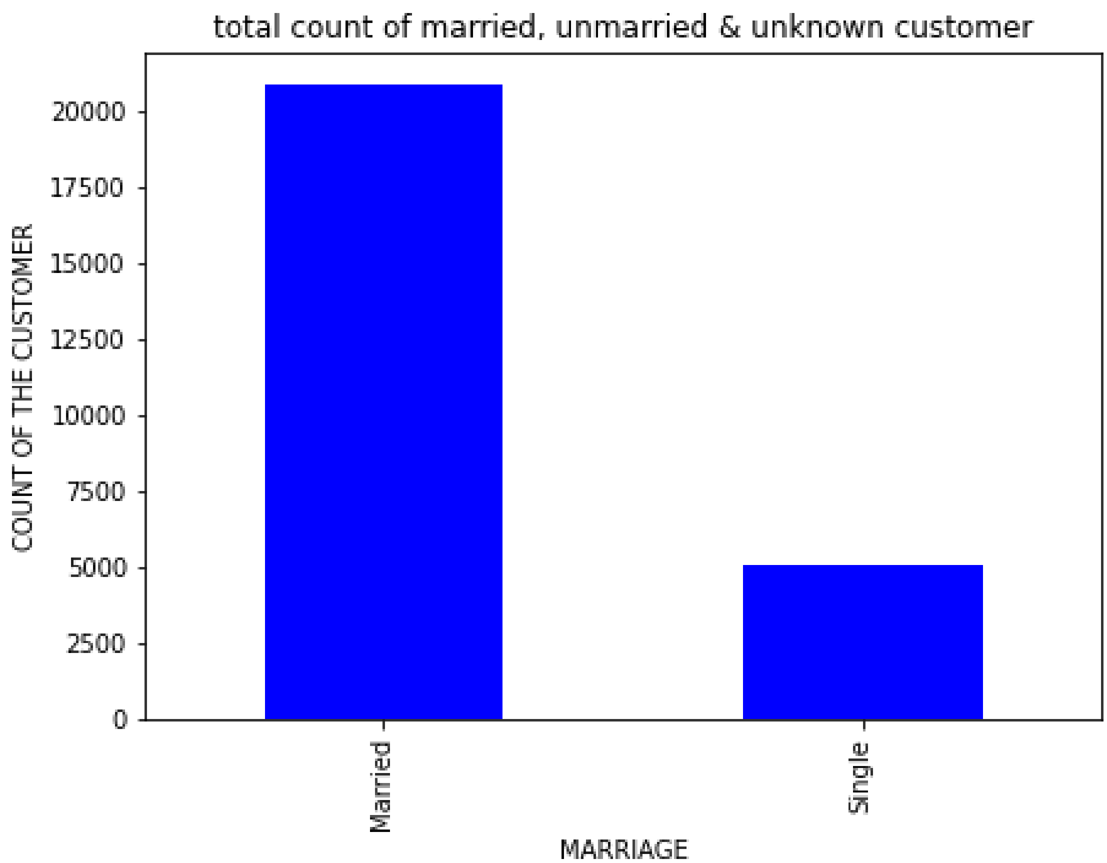


In this bar graph, M represent male, F female and U unknowns. It is found that more male count included in sale of products as the count of male is highest (22500) as compared to other two categories female and unknown. Females have approximately (17000) count in this dataset who bought the products which are on second position. Whereas, there are some unknown values in gender attributes the count of unknown is approximately 150. Least count of gender is denoted in the category of unknown.

4. By predictive analysis find out which type of consumers increased the sales of products married and unmarried in the dataset?

```
MARRIAGE= df['marriage']
MARRIAGE.value_counts()

Married    20891
Single      5082
Name: marriage, dtype: int64

df['marriage'].value_counts().max()

20891

x = ['married', 'single']
plt.xlabel("MARRIAGE")
plt.ylabel("COUNT OF THE CUSTOMER")
plt.title('total count of married, unmarried & unknown customer')
```

```
MARRIAGE.value_counts().plot(kind='bar',  color = 'blue',
figsize=(7,5))
#plt.savefig('total count of married, unmarried & unknown
customer.jpg')
plt.show()
```



In married and unmarried bar graph it is found that more count of customers is married in this dataset that is approximately 21000 who purchased more products from the company and increased the sale of company. The count of unmarried customers in this dataset is 500 which is less as compared to married category.

5. Find out how many customers have highest and lowest family income?

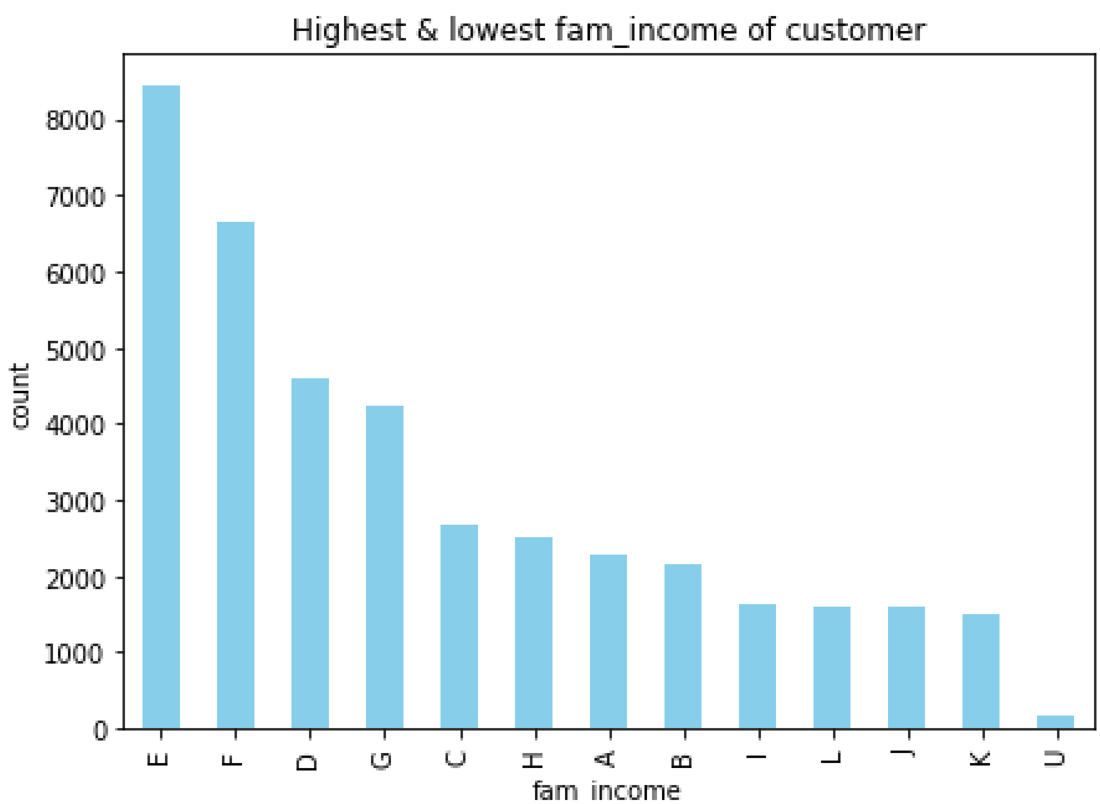
```
fam_income1= df['fam_income']
fam_income1.value_counts()
```

```
E    8432
F    6641
D    4582
G    4224
C    2687
H    2498
```

```
A    2274
B    2169
I    1622
L    1617
J    1614
K    1487
U     153
Name: fam_income, dtype: int64
```

```
df['fam_income'].value_counts().max()
```

```
plt.xlabel("fam_income")
plt.ylabel("count")
plt.title('Highest & lowest fam_income of customer')
fam_income1.value_counts().plot(kind='bar',color='skyblue',
figsize=(7,5))
#plt.savefig('Highest & lowest fam_income of customer.jpg')
plt.show()
```



From this bar graph it clear that in this dataset there are 13 levels of family income in which customers are divided. The customer who has highest income the found in level E and the count of those customers are 8400. Whereas, the customer who have least family income level they are found in the level U. The count of customer who have least family income is 200.

```
df["flag"].unique()
```

```
df["flag"].value_counts()
```

```
Y    20000
N    20000
Name: flag, dtype: int64
```

```
df["gender"].unique()
```

```
array(['M', 'F', 'U'], dtype=object)
```

```
df["gender"].value_counts()
```

```
M    22019
F    16830
U     1151
Name: gender, dtype: int64
```

```
df["online"].unique()
```

```
array(['N', 'Y'], dtype=object)
```

```
df["online"].value_counts()
```

```
Y    27319
N    12681
Name: online, dtype: int64
```

```
a = df.online.value_counts()
#a.to_csv('a.csv')
```

```
df['flag'] = df['flag'].cat.rename_categories({'Y': 1, 'N':
2}).astype(int)
df['gender'] =df['gender'].cat.rename_categories({'M': 1, 'F':
2,'U':3}).astype(int)
df['online'] =df['online'].cat.rename_categories({'N':2 , 'Y':
1}).astype(int)
df['marriage'] =df['marriage'].cat.rename_categories({'Married':1 ,
'Single': 2}).astype(int)
df['occupation']
=df['occupation'].cat.rename_categories({'Professional':1,'Sales/Servi
ce':2,'Blue Collar':3,'Others':4,'Retired':5,'Farm':6}).astype(int)
df['region']
=df['region'].cat.rename_categories({'Midwest':1,'Northeast':2,'West':
3,'South':4,'Rest':5}).astype(int)
df['family_income']
=df['family_income'].cat.rename_categories({'E':1,'F':2,'D':3,'G':4,'C
':5,'H':6,'A':7,'B':8,'I':9,'J':10,'K':11,'U':12,'L':13}).astype(int)
```

Recheck the Data type

```
df.dtypes
```

```
flag                         int64
gender                       int64
education                 category
house_value                  int64
age                         string
online                       int64
customer_psychology         object
marriage                     int64
child                     category
occupation                   int64
mortgage                  category
house_owner               category
region                       int64
car_probability              int64
family_income                int64
customer_psychology       category
dtype: object
```

Checking the target varible is balance or imbalance

```
df["online"].value_counts() # checking amount of values of target
variable
```

```
1    27296
2    12649
Name: online, dtype: int64
```

```
df.dtypes
```

```
flag                         int64
gender                       int64
education                 category
house_value                  int64
age                         string
online                       int64
customer_psychology         object
marriage                     int64
child                     category
occupation                   int64
mortgage                  category
house_owner               category
region                       int64
car_probability              int64
family_income                int64
customer_psychology       category
dtype: object
```

```
df['Age'] = df['age'].str[0].astype(int)
df['Mortgage'] = df['mortgage'].str[0].astype(int)
```

```
df
```

```
       flag  gender        education  house_value     age  online  \
0         1       1          4. Grad       756460   1_Unk       2
1         2       2          3. Bach       213171   7_>65       2
2         2       1  2. Some College       111147  2_<=25       1
3         1       1  2. Some College       354151  2_<=25       1
4         1       2  2. Some College       117087   1_Unk       1
...     ...     ...              ...          ...     ...     ...
39995     1       2          3. Bach            0   7_>65       1
39996     2       2            1. HS       213596  4_<=45       2
39997     1       1            0. <HS      134070  3_<=35       1
39998     2       1            1. HS       402210   7_>65       1
39999     2       2          3. Bach       836030   7_>65       1

      customer_psychology  marriage child  occupation mortgage
house_owner  \
0                       B         0     U           1     1Low
0
1                       E         0     U           1     1Low
Owner
2                       C         0     Y           1     1Low
Owner
3                       B         2     U           2     1Low
0
4                       J         1     Y           2     1Low
0
...                   ...       ...   ...         ...      ...
...
39995                   C         0     U           5     1Low
0
39996                   I         1     U           3     1Low
Owner
39997                   F         1     U           2     1Low
Owner
39998                   E         0     Y           2     1Low
0
39999                   B         1     N           5     2Med
Owner

       region  car_probability  family_income customer_psychology
Age  \
0           1                1             13                   B
1
1           2                3              4                   E
7
2           1                1             10                   C
2
3           3                2             13                   B
2
4           4                7              6                   J
1
```

```
...       ...                ...              ...                    ...  ..
.
39995       4                  3                2                      C
7
39996       4                  1                3                      I
4
39997       1                  4                1                      F
3
39998       3                  2                8                      E
7
39999       2                  1               10                      B
7

       Mortgage
0             1
1             1
2             1
3             1
4             1
...         ...
39995         1
39996         1
39997         1
39998         1
39999         2

[39945 rows x 18 columns]
```

Rearrange the Attributes

```
df=df.reindex(columns= ['flag', 'gender', 'house_value', 'Age',
        'marriage', 'occupation', 'Mortgage',
     'region', 'car_probability', 'family_income','online'])#
rearrange columns for easy to progress
print(df)

       flag  gender  house_value  Age  marriage  occupation  Mortgage
region  \
0         1       1       756460    1         0           1         1
1
1         2       2       213171    7         0           1         1
2
2         2       1       111147    2         0           1         1
1
3         1       1       354151    2         2           2         1
3
4         1       2       117087    1         1           2         1
4
...     ...     ...          ...  ...       ...         ...       ...
...
```

```
39995     1       2           0    7         0           5         1
4
39996     2       2      213596    4         1           3         1
4
39997     1       1      134070    3         1           2         1
1
39998     2       1      402210    7         0           2         1
3
39999     2       2      836030    7         1           5         2
2

       car_probability  family_income  online
0                    1             13       2
1                    3              4       2
2                    1             10       1
3                    2             13       1
4                    7              6       1
...                ...            ...     ...
39995                3              2       1
39996                1              3       2
39997                4              1       1
39998                2              8       1
39999                1             10       1

[39945 rows x 11 columns]
```

Separating the independent Variables from the Dependent Variables

```
x = df.iloc[:,:-1] # independent variables
y = df.iloc[:,-1]  # dependent variable
```

Show only independent Variables

```
x.head(4) # print first 4 rows of independence
```

```
   flag  gender  house_value  Age  marriage  occupation  Mortgage
region  \
0     1       1       756460    1         0           1         1
1
1     2       2       213171    7         0           1         1
2
2     2       1       111147    2         0           1         1
1
3     1       1       354151    2         2           2         1
3

   car_probability  family_income
0                1             13
1                3              4
```

```
2                 1             10
3                 2             13
```

It show ten records corresponding response varibles

```
y.head(10) #print first 10 rows of dependence variable
```

```
0    2
1    2
2    1
3    1
4    1
5    1
6    1
7    1
8    1
9    1
Name: online, dtype: int64
```

train-test-split method

```
from sklearn.model_selection import train_test_split # train test
split package
```

With random state

```
x_train, x_test, y_train, y_test =
train_test_split(x,y,test_size=0.30,random_state=4) # train
independent , y target, response
```

```
x
```

```
       flag  gender  house_value  Age  marriage  occupation  Mortgage
region  \
0         1       1       756460    1         0           1         1
1
1         2       2       213171    7         0           1         1
2
2         2       1       111147    2         0           1         1
1
3         1       1       354151    2         2           2         1
3
4         1       2       117087    1         1           2         1
4
...     ...     ...          ...  ...       ...         ...       ...
...
39995     1       2            0    7         0           5         1
4
39996     2       2       213596    4         1           3         1
4
```

```
39997     1       1       134070    3         1           2         1
1
39998     2       1       402210    7         0           2         1
3
39999     2       2       836030    7         1           5         2
2

       car_probability  family_income
0                    1             13
1                    3              4
2                    1             10
3                    2             13
4                    7              6
...                ...            ...
39995                3              2
39996                1              3
39997                4              1
39998                2              8
39999                1             10

[39945 rows x 10 columns]
```

```
x_train.head(3)
```

```
       flag  gender  house_value  Age  marriage  occupation  Mortgage
region  \
18980     2       1       144425    3         0           3         2
1
13589     1       2       261327    5         0           1         1
4
5487      2       2       172132    3         0           2         1
4

       car_probability  family_income
18980                6              2
13589                1              2
5487                 3              7
```

```
y_train.head(5)
```

```
18980    1
13589    1
5487     1
2040     2
21531    1
Name: online, dtype: int64
```

Summarize Class Distribution

```
print("Before undersampling:")
y_train.value_counts() # show target variable imbalance
```

```
Before undersampling:

1    19103
2     8858
Name: online, dtype: int64
```

Define under sampling strategy

```
!pip install imblearn #libraray
```

```
Requirement already satisfied: imblearn in
/usr/local/lib/python3.7/dist-packages (0.0)
Requirement already satisfied: imbalanced-learn in
/usr/local/lib/python3.7/dist-packages (from imblearn) (0.8.1)
Requirement already satisfied: scikit-learn>=0.24 in
/usr/local/lib/python3.7/dist-packages (from imbalanced-learn-
>imblearn) (1.0.2)
Requirement already satisfied: scipy>=0.19.1 in
/usr/local/lib/python3.7/dist-packages (from imbalanced-learn-
>imblearn) (1.4.1)
Requirement already satisfied: numpy>=1.13.3 in
/usr/local/lib/python3.7/dist-packages (from imbalanced-learn-
>imblearn) (1.21.6)
Requirement already satisfied: joblib>=0.11 in
/usr/local/lib/python3.7/dist-packages (from imbalanced-learn-
>imblearn) (1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from scikit-learn>=0.24-
>imbalanced-learn->imblearn) (3.1.0)
```

```
from imblearn.under_sampling import RandomUnderSampler
```

```
undersample =
RandomUnderSampler(sampling_strategy='majority',random_state=4)
```

Fit and Apply the Transform, Random State Three

```
x_train_under, y_train_under =
undersample.fit_resample(x_train,y_train)
x_train_under.head(3)
```

```
   flag  gender  house_value  Age  marriage  occupation  Mortgage
region  \
0     2       1       313625    7         1           5         2
4
1     2       1            0    1         1           2         1
```

```
2
2     1       1            0    3         0           1         1
1

   car_probability  family_income
0                2              4
1                7              1
2                7              9
```

Recheck the target variable is balance or imbalance

```
print("after undersampling:")
y_train_under.value_counts()
```

```
after undersampling:

1    8858
2    8858
Name: online, dtype: int64
```

```
print(y)
```

```
0        2
1        2
2        1
3        1
4        1
        ..
39995    1
39996    2
39997    1
39998    1
39999    1
Name: online, Length: 39945, dtype: int64
```

```
y_train = y_train.astype('float')
```

```
y_train
```

```
24525    1.0
9083     2.0
7349     2.0
18792    1.0
1889     2.0
        ...
23774    1.0
12061    1.0
27576    1.0
8506     2.0
17845    1.0
Name: online, Length: 27447, dtype: float64
```

Modelling

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import
classification_report,confusion_matrix,accuracy_score
from sklearn import metrics
```

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

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(random_state=0)
model.fit(x_train_under,y_train_under)
```

```
LogisticRegression(random_state=0)
```

```
predict_logistic=model.predict(x_test)
predict_logistic
```

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

```
y_test
```

```
37592    1
7713     1
10464    1
35632    1
23664    1
        ..
34354    1
15053    1
9408     1
19429    1
21402    1
Name: online, Length: 11984, dtype: int64
```

```
cf_matrix_logistic = confusion_matrix(y_test,predict_logistic)
logistic_acc = accuracy_score(y_test,predict_logistic)*100
print("accuracy of Logistic Regression:",logistic_acc)
```

```
accuracy of Logistic Regression: 67.06441922563417
```

```
from sklearn.ensemble import RandomForestClassifier
RFC = RandomForestClassifier()
RFC.fit(x_train_under,y_train_under)
pred_RFC=RFC.predict(x_test)
RFC_acc = accuracy_score(y_test,pred_RFC)*100
print("accuracy of RFC:",RFC_acc)
```

```
accuracy of RFC: 65.57910547396529
```

```
from sklearn.naive_bayes import GaussianNB
GNB=GaussianNB()
GNB.fit(x_train_under,y_train_under)
pred_GNB=GNB.predict(x_test)
GNB_acc = accuracy_score(y_test,pred_GNB)*100
print("accuracy of GNB :", GNB_acc)
```

```
accuracy of GNB : 42.523364485981304
```

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
knn.fit(x_train_under,y_train_under)
pred_k=knn.predict(x_test)
knn_acc = accuracy_score(y_test,pred_k)*100
print("accuracy of KNN :", knn_acc)
```

```
accuracy of KNN : 56.78404539385847
```

```
labels = ["Logistic Regression","KNN","Naive Bayes","random forest"]
x = [logistic_acc,knn_acc,GNB_acc,RFC_acc]
eval_frame = pd.DataFrame()
eval_frame['Model'] = labels
eval_frame['train_test_split'] = x
eval_frame
```

	Model	train_test_split
0	Logistic Regression	67.064419
1	KNN	56.784045
2	Naive Bayes	42.523364
3	random forest	65.579105

K-FOLDS CROSS VALIDATION

```
from sklearn.model_selection import KFold

kfold = KFold(n_splits = 5)

#Modeling step test differents algorithms

classifiers1 = []

classifiers1.append(KNeighborsClassifier())
classifiers1.append(LogisticRegression())
classifiers1.append(GaussianNB())
classifiers1.append(RandomForestClassifier())

from sklearn.model_selection import cross_val_score

accuracy_results1 = []
for a in classifiers1:
  accuracy_results1.append(cross_val_score(a, x_train_under,
```

```
y_train_under, scoring = "accuracy", cv = kfold))#Here a is 1st model
knn
```

```
#folds corresponding to models
accuracy_results1
```

```
[array([0.38233634, 0.38893593, 0.55151002, 0.33107536, 0.34575219]),
 array([0.        , 0.09060119, 0.55066328, 0.00282247, 0.00366921]),
 array([0.09057562, 0.09596387, 0.54784081, 0.        , 0.        ]),
 array([0.48250564, 0.49082698, 0.64436918, 0.46683601, 0.45667513])]
```

```
accuracy_means1 = []
for e in accuracy_results1:
  accuracy_means1.append(e.mean()*100)
```

```
accuracy_means1
```

```
[39.99219680303068, 12.955122777307365, 14.687606121248841,
50.82425907059927]
```

```
eval_frame['kfolds_5'] = accuracy_means1
eval_frame
```

	Model	train_test_split	kfolds_5
0	Logistic Regression	67.064419	39.992197
1	KNN	56.784045	12.955123
2	Naive Bayes	42.523364	14.687606
3	random forest	65.579105	50.824259

STRATIFIED K FOLD

```
from sklearn.model_selection import StratifiedKFold
```

```
Stratifiedkfold = StratifiedKFold(n_splits = 5)
```

```
# Modeling step Test differents algorithms
classifiers_4 = []
classifiers_4.append(KNeighborsClassifier())
classifiers_4.append(LogisticRegression())
classifiers_4.append(GaussianNB())
classifiers_4.append(RandomForestClassifier())
accuracy_results_4 = []
for classifier in classifiers_4 :
    accuracy_results_4.append(cross_val_score(classifier,
x_train_under,y_train_under, scoring = "accuracy", cv =
Stratifiedkfold))
accuracy_means_4 = []
for accuracy_result in accuracy_results_4:
    accuracy_means_4.append(accuracy_result.mean()*100)
accuracy_means_4
eval_frame['Stratifiedkfold_5']=accuracy_means_4
eval_frame
```

```
                 Model  train_test_split   kfolds_5  Stratifiedkfold_5
0  Logistic Regression         67.064419  39.992197          56.463094
1                  KNN         56.784045  12.955123          56.259879
2          Naive Bayes         42.523364  14.687606          54.713180
3        random forest         65.579105  50.824259          64.862292
```

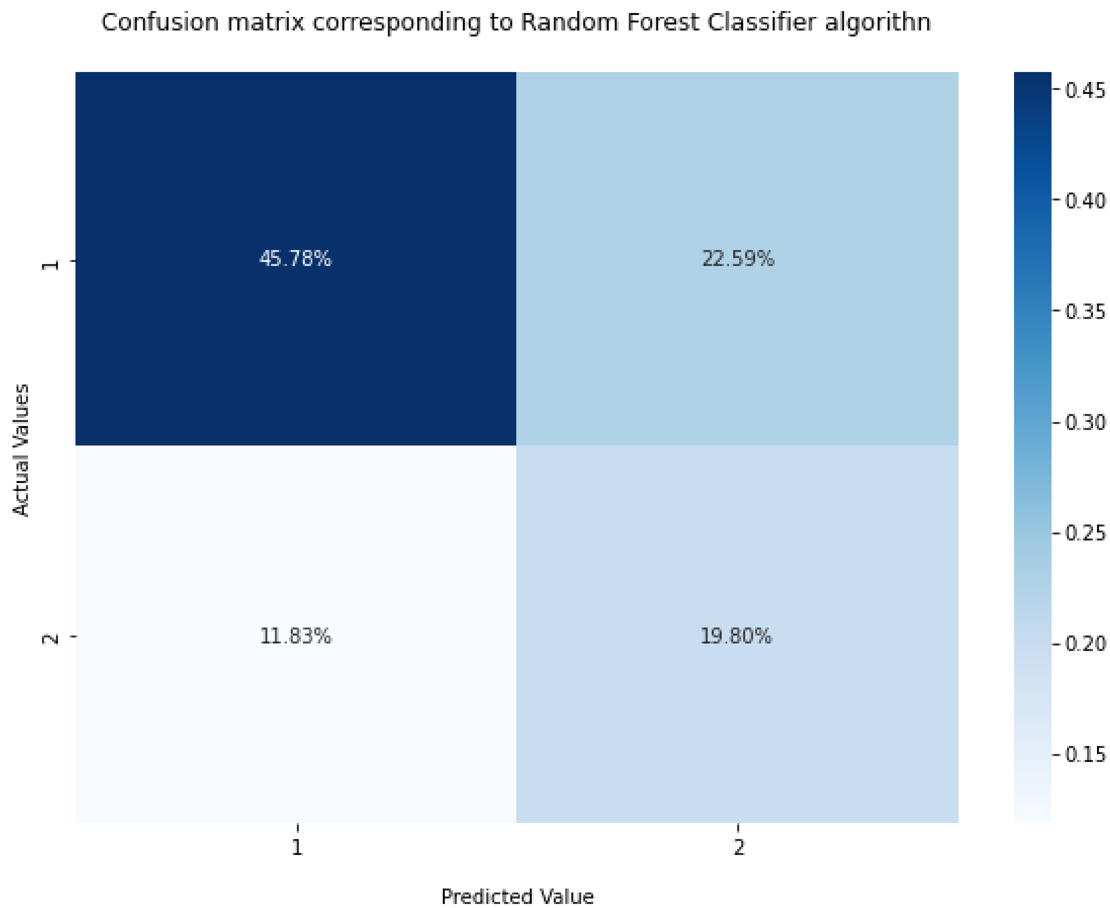
Repeated Random Train-Test Splits

```
from sklearn.model_selection import ShuffleSplit

kfold = ShuffleSplit(n_splits=5,test_size=0.3)
# Modeling step Test differents algorithms
classifiers_2 = []
classifiers_2.append(KNeighborsClassifier())
classifiers_2.append(LogisticRegression())
classifiers_2.append(GaussianNB())
classifiers_2.append(RandomForestClassifier())
accuracy_results_2 = []
for classifier in classifiers_2 :
    accuracy_results_2.append(cross_val_score(classifier, 
x_train_under,y_train_under, scoring = "accuracy", cv = kfold))
accuracy_means_2 = []
for accuracy_result in accuracy_results_2:
    accuracy_means_2.append(accuracy_result.mean()*100)
accuracy_means_2
eval_frame['RRTestTrainSplits_5']=accuracy_means_2
eval_frame.round(2)
```

```
                 Model  train_test_split  kfolds_5  Stratifiedkfold_5 
\
0  Logistic Regression             67.06     39.99              56.46   

1                  KNN             56.78     12.96              56.26   

2          Naive Bayes             42.52     14.69              54.71   

3        random forest             65.58     50.82              64.86   


   RRTestTrainSplits_5  
0                56.76  
1                56.31  
2                54.97  
3                64.81  
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,pred_RFC)
plt.figure(figsize=(10,7))
ax=sns.heatmap(cm/np.sum(cm),annot=True,fmt='.2%',cmap="Blues")
ax.set_title('Confusion matrix corresponding to Random Forest 
Classifier algorithn\n')
```

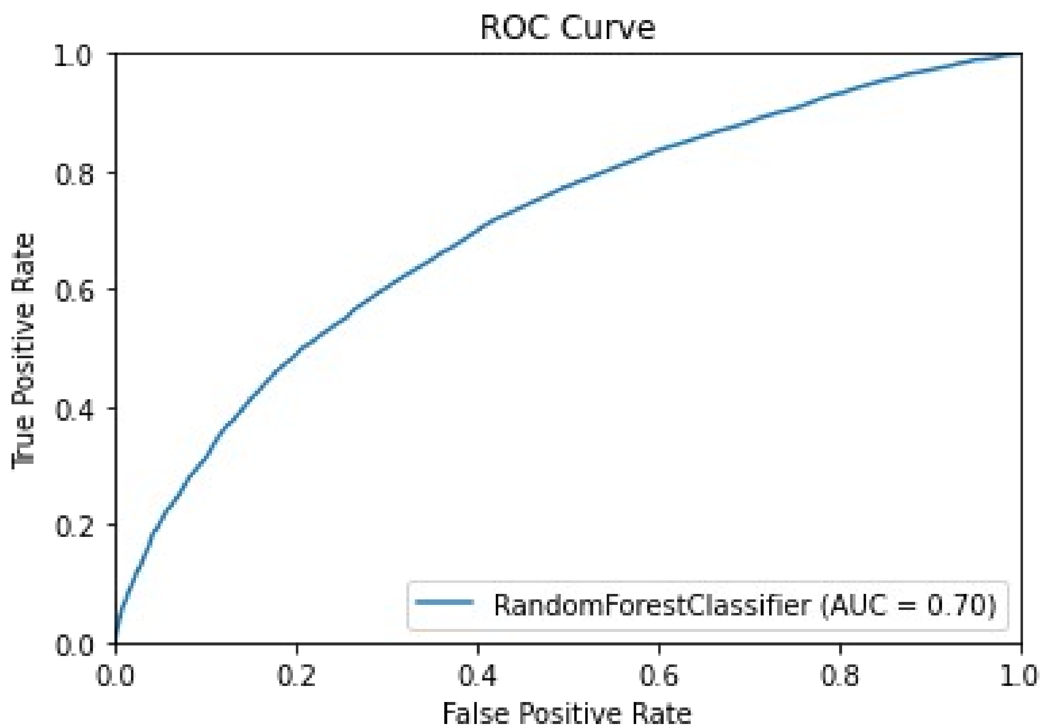
```
ax.set_xlabel("\nPredicted Value")
ax.set_ylabel("Actual Values")

## Ticket labels - list must be in alphabetical order
ax.xaxis.set_ticklabels(['1','2'])
ax.yaxis.set_ticklabels(['1','2'])

## Display the visualization of the Confusion Matric
#plt.savefig("cf_matrix_KNN.png",bbox_inches = 'tight')
plt.show()
print("accuracy of Random Forest :",RFC_acc)
```




```
accuracy of Random Forest : 65.57910547396529

from sklearn.metrics import RocCurveDisplay, roc_auc_score

#from sklearn import metrics
RFC = RFC.fit(x_train_under, y_train_under)
metrics.plot_roc_curve(RFC, x_test, y_test)
plt.plot([1, 2], [1, 2], 'g--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate')
```

```
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.savefig("ROC Curev.png")
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/
deprecation.py:87: FutureWarning: Function plot_roc_curve is
deprecated; Function :func:`plot_roc_curve` is deprecated in 1.0 and
will be removed in 1.2. Use one of the class
methods: :meth:`sklearn.metric.RocCurveDisplay.from_predictions`
or :meth:`sklearn.metric.RocCurveDisplay.from_estimator`.
  warnings.warn(msg, category=FutureWarning)
```




```
print(classification_report(y_test,pred_RFC))
```

```
              precision    recall  f1-score   support

           1       0.79      0.67      0.73      8193
           2       0.47      0.63      0.54      3791

    accuracy                           0.66     11984
   macro avg       0.63      0.65      0.63     11984
weighted avg       0.69      0.66      0.67     11984
```