BlackFriday

December 23, 2023

0.1 Exploratory Data Analysis on Black Friday Dataset

0.2 Problem Statement

A retail company "ABC Private Limited" wants to understand the customer purchase behaviour (specifically, purchase amount) against various products of different categories. They have shared purchase summary of various customers for selected high volume products from last month. The data set also contains customer demographics (age, gender, marital status, city_type, stay_in_current_city), product details (product_id and product category) and Total purchase_amount from last month.

Now, they want to build a model to predict the purchase amount of customer against various products which will help them to create personalized offer for customers against different products.

[1]:
```
import numpy as np
import pandas as pd
import matplotlib as plt
import seaborn as sns
%matplotlib inline
```

[2]:
```
### importing datasets train and test
df_train=pd.read_csv("train.csv")
df_test=pd.read_csv("test.csv")
```

[3]:
```
df_train.columns
```

[3]:
```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category_1',
       'Product_Category_2', 'Product_Category_3', 'Purchase'],
      dtype='object')
```

[4]:
```
df_test.columns
```

[4]:
```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category_1',
       'Product_Category_2', 'Product_Category_3'],
      dtype='object')
```

[5]:
```
df=pd.concat([df_train,df_test])
```

[6]:
```
df.head()
```

[6]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
0	1000001	P00069042	F	0-17	10	A	2	0	3	NaN	NaN	8370.0
1	1000001	P00248942	F	0-17	10	A	2	0	1	6.0	14.0	15200.0
2	1000001	P00087842	F	0-17	10	A	2	0	12	NaN	NaN	1422.0
3	1000001	P00085442	F	0-17	10	A	2	0	12	14.0	NaN	1057.0
4	1000002	P00285442	M	55+	16	C	4+	0	8	NaN	NaN	7969.0

[7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 783667 entries, 0 to 233598
Data columns (total 12 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   User_ID                     783667 non-null  int64
 1   Product_ID                  783667 non-null  object
 2   Gender                      783667 non-null  object
 3   Age                         783667 non-null  object
 4   Occupation                  783667 non-null  int64
 5   City_Category               783667 non-null  object
 6   Stay_In_Current_City_Years  783667 non-null  object
 7   Marital_Status              783667 non-null  int64
 8   Product_Category_1          783667 non-null  int64
 9   Product_Category_2          537685 non-null  float64
 10  Product_Category_3          237858 non-null  float64
 11  Purchase                    550068 non-null  float64
dtypes: float64(3), int64(4), object(5)
memory usage: 77.7+ MB
```

[8]:
```
df.describe()
```

[8]:
```
            User_ID     Occupation  Marital_Status  Product_Category_1  \
count  7.836670e+05  783667.000000   783667.000000       783667.000000
mean   1.003029e+06       8.079300        0.409777            5.366196
std    1.727267e+03       6.522206        0.491793            3.878160
min    1.000001e+06       0.000000        0.000000            1.000000
25%    1.001519e+06       2.000000        0.000000            1.000000
50%    1.003075e+06       7.000000        0.000000            5.000000
75%    1.004478e+06      14.000000        1.000000            8.000000
max    1.006040e+06      20.000000        1.000000           20.000000

       Product_Category_2  Product_Category_3       Purchase
count       537685.000000       237858.000000  550068.000000
mean             9.844506           12.668605    9263.968713
std              5.089093            4.125510    5023.065394
min              2.000000            3.000000      12.000000
25%              5.000000            9.000000    5823.000000
50%              9.000000           14.000000    8047.000000
75%             15.000000           16.000000   12054.000000
max             18.000000           18.000000   23961.000000
```

[9]:
```
#Removing Unneccesary data
df.drop(["User_ID"],axis=1,inplace=True)
```

[10]:
```
df.head()
```

[10]:
```
  Product_ID Gender   Age  Occupation City_Category  \
0  P00069042      F  0-17          10             A
1  P00248942      F  0-17          10             A
2  P00087842      F  0-17          10             A
3  P00085442      F  0-17          10             A
4  P00285442      M   55+          16             C

  Stay_In_Current_City_Years  Marital_Status  Product_Category_1  \
0                          2               0                   3
1                          2               0                   1
2                          2               0                  12
3                          2               0                  12
4                         4+               0                   8

   Product_Category_2  Product_Category_3  Purchase
0                 NaN                 NaN    8370.0
1                 6.0                14.0   15200.0
2                 NaN                 NaN    1422.0
3                14.0                 NaN    1057.0
4                 NaN                 NaN    7969.0
```

[11]:
```
pd.get_dummies(["Gender"])
```

[11]:
```
   Gender
0    True
```

[12]:
```
##Converting Categorical values to Numerical values
##HAndling categorical feature Gender
df['Gender']=df['Gender'].map({'F':0,'M':1})
```

[13]:
```
df.head()
```

[13]:
```
  Product_ID  Gender   Age  Occupation City_Category  \
0  P00069042       0  0-17          10             A
1  P00248942       0  0-17          10             A
2  P00087842       0  0-17          10             A
3  P00085442       0  0-17          10             A
4  P00285442       1   55+          16             C

  Stay_In_Current_City_Years  Marital_Status  Product_Category_1  \
0                          2               0                   3
1                          2               0                   1
2                          2               0                  12
3                          2               0                  12
4                         4+               0                   8

   Product_Category_2  Product_Category_3  Purchase
0                 NaN                 NaN    8370.0
1                 6.0                14.0   15200.0
2                 NaN                 NaN    1422.0
3                14.0                 NaN    1057.0
4                 NaN                 NaN    7969.0
```

[14]:
```
df['Age'].unique()
```

[14]:
```
array(['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'],
      dtype=object)
```

[15]:
```
##HAndling categorical feature Age
#pd.get_dummies(df['Age'],drop_first=True)
df['Age']=df['Age'].map({'0-17':1,'18-25':2,'26-35':3,'36-45':4,'46-50':
↪5,'51-55':6,'55+':7})
```

##second technqiue from sklearn import preprocessing

label_encoder object knows how to understand word labels. label_encoder = preprocessing.LabelEncoder()

Encode labels in column 'species'. df['Age']= label_encoder.fit_transform(df['Age'])

df['Age'].unique()

[16]:
```
df.head()
```

[16]:
```
  Product_ID  Gender  Age  Occupation City_Category  \
0  P00069042       0    1          10             A
1  P00248942       0    1          10             A
2  P00087842       0    1          10             A
3  P00085442       0    1          10             A
4  P00285442       1    7          16             C

  Stay_In_Current_City_Years  Marital_Status  Product_Category_1  \
0                          2               0                   3
1                          2               0                   1
2                          2               0                  12
3                          2               0                  12
4                         4+               0                   8

   Product_Category_2  Product_Category_3  Purchase
0                 NaN                 NaN    8370.0
1                 6.0                14.0   15200.0
2                 NaN                 NaN    1422.0
3                14.0                 NaN    1057.0
4                 NaN                 NaN    7969.0
```

[17]:
```
df_city=pd.get_dummies(df['City_Category'],drop_first=True)
df_city=df_city.astype(int)
```

[18]:
```
df_city.head()
```

[18]:
```
   B  C
0  0  0
1  0  0
2  0  0
3  0  0
4  0  1
```

[19]:
```
df=pd.concat([df,df_city],axis=1)
```

0.3 deleting last two columns df = df.iloc[:, :-2]

[20]:
```
df.head()
```

[20]:
```
  Product_ID  Gender  Age  Occupation City_Category  \
0  P00069042       0    1          10             A
1  P00248942       0    1          10             A
2  P00087842       0    1          10             A
3  P00085442       0    1          10             A
4  P00285442       1    7          16             C

  Stay_In_Current_City_Years  Marital_Status  Product_Category_1  \
```

```
0                          2                  0                   3
1                          2                  0                   1
2                          2                  0                  12
3                          2                  0                  12
4                         4+                  0                   8

   Product_Category_2  Product_Category_3  Purchase  B  C
0                 NaN                 NaN    8370.0  0  0
1                 6.0                14.0   15200.0  0  0
2                 NaN                 NaN    1422.0  0  0
3                14.0                 NaN    1057.0  0  0
4                 NaN                 NaN    7969.0  0  1
```

[21]:
```
df.isnull().sum()
```

[21]:
```
Product_ID                        0
Gender                            0
Age                               0
Occupation                        0
City_Category                     0
Stay_In_Current_City_Years        0
Marital_Status                    0
Product_Category_1                0
Product_Category_2           245982
Product_Category_3           545809
Purchase                     233599
B                                 0
C                                 0
dtype: int64
```

[22]:
```
## Handling missing values
df['Product_Category_2'].unique()
```

[22]:
```
array([nan,  6., 14.,  2.,  8., 15., 16., 11.,  5.,  3.,  4., 12.,  9.,
       10., 17., 13.,  7., 18.])
```

[23]:
```
df['Product_Category_2'].value_counts()
```

[23]:
```
Product_Category_2
8.0     91317
14.0    78834
2.0     70498
16.0    61687
15.0    54114
5.0     37165
4.0     36705
6.0     23575
```

```
11.0    20230
17.0    19104
13.0    15054
9.0      8177
12.0     7801
10.0     4420
3.0      4123
18.0     4027
7.0       854
Name: count, dtype: int64
```

[24]:
```
df['Product_Category_2'].isnull().sum()
```

[24]:
```
245982
```

[25]:
```
df['Product_Category_2'].mode()[0]
```

[25]:
```
8.0
```

[26]:
```
df['Product_Category_2']=df['Product_Category_2'].
 ↪fillna(df['Product_Category_2'].mode()[0])
```

[27]:
```
df['Product_Category_2'].isnull().sum()
```

[27]:
```
0
```

[28]:
```
df.drop(['City_Category'],axis=1,inplace=True)
```

[29]:
```
df.head()
```

[29]:

	Product_ID	Gender	Age	Occupation	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase	B	C
0	P00069042	0	1	10	2	0	3	8.0	NaN	8370.0	0	0
1	P00248942	0	1	10	2	0	1	6.0	14.0	15200.0	0	0
2	P00087842	0	1	10	2	0	12	8.0	NaN	1422.0	0	0
3	P00085442	0	1	10	2	0	12	14.0	NaN			
4	P00285442	1	7	16	4+	0	8	8.0	NaN			

```
3    1057.0  0  0
4    7969.0  0  1
```

[30]:
```
df['Product_Category_3'].unique()
```

[30]:
```
array([nan, 14., 17.,  5.,  4., 16., 15.,  8.,  9., 13.,  6., 12.,  3.,
       18., 11., 10.])
```

[31]:
```
df['Product_Category_3'].value_counts()
```

[31]:
```
Product_Category_3
16.0    46469
15.0    39968
14.0    26283
17.0    23818
5.0     23799
8.0     17861
9.0     16532
12.0    13115
13.0     7849
6.0      6888
18.0     6621
4.0      2691
11.0     2585
10.0     2501
3.0       878
Name: count, dtype: int64
```

[32]:
```
df['Product_Category_3'].mode()[0]
```

[32]:
```
16.0
```

[33]:
```
df['Product_Category_3']=df['Product_Category_3'].
 ↪fillna(df['Product_Category_3'].mode()[0])
```

[34]:
```
df.head()
```

[34]:

	Product_ID	Gender	Age	Occupation	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3
0	P00069042	0	1	10	2	0	3	8.0	16.0
1	P00248942	0	1	10	2	0	1	6.0	14.0
2	P00087842	0	1	10	2	0	12	8.0	16.0
3	P00085442	0	1	10	2	0			
4	P00285442	1	7	16	4+				

```
3                0                  12                14.0                16.0
4                0                   8                 8.0                16.0

   Purchase  B  C
0    8370.0  0  0
1   15200.0  0  0
2    1422.0  0  0
3    1057.0  0  0
4    7969.0  0  1
```

[35]:
```
df['Stay_In_Current_City_Years']
```

[35]:
```
0          2
1          2
2          2
3          2
4         4+
          ..
233594    4+
233595    4+
233596    4+
233597    4+
233598    4+
Name: Stay_In_Current_City_Years, Length: 783667, dtype: object
```

[36]:
```
# Replacing 4+ to 4
df['Stay_In_Current_City_Years']=df['Stay_In_Current_City_Years'].str.
 ↪replace('+','')
```

[37]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 783667 entries, 0 to 233598
Data columns (total 12 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   Product_ID                  783667 non-null  object
 1   Gender                      783667 non-null  int64
 2   Age                         783667 non-null  int64
 3   Occupation                  783667 non-null  int64
 4   Stay_In_Current_City_Years  783667 non-null  object
 5   Marital_Status              783667 non-null  int64
 6   Product_Category_1          783667 non-null  int64
 7   Product_Category_2          783667 non-null  float64
 8   Product_Category_3          783667 non-null  float64
 9   Purchase                    550068 non-null  float64
 10  B                           783667 non-null  int32
 11  C                           783667 non-null  int32
```

```
dtypes: float64(3), int32(2), int64(5), object(2)
memory usage: 71.7+ MB
```

[38]:
```
## converting object into int
df['Stay_In_Current_City_Years']=df['Stay_In_Current_City_Years'].astype(int)
```

[39]:
```
df['B']=df['B'].astype(int)
```

[40]:
```
df['C']=df['C'].astype(int)
```

[41]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 783667 entries, 0 to 233598
Data columns (total 12 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   Product_ID                  783667 non-null  object
 1   Gender                      783667 non-null  int64
 2   Age                         783667 non-null  int64
 3   Occupation                  783667 non-null  int64
 4   Stay_In_Current_City_Years  783667 non-null  int32
 5   Marital_Status              783667 non-null  int64
 6   Product_Category_1          783667 non-null  int64
 7   Product_Category_2          783667 non-null  float64
 8   Product_Category_3          783667 non-null  float64
 9   Purchase                    550068 non-null  float64
 10  B                           783667 non-null  int32
 11  C                           783667 non-null  int32
dtypes: float64(3), int32(3), int64(5), object(1)
memory usage: 68.8+ MB
```

[42]:
```
df.head()
```

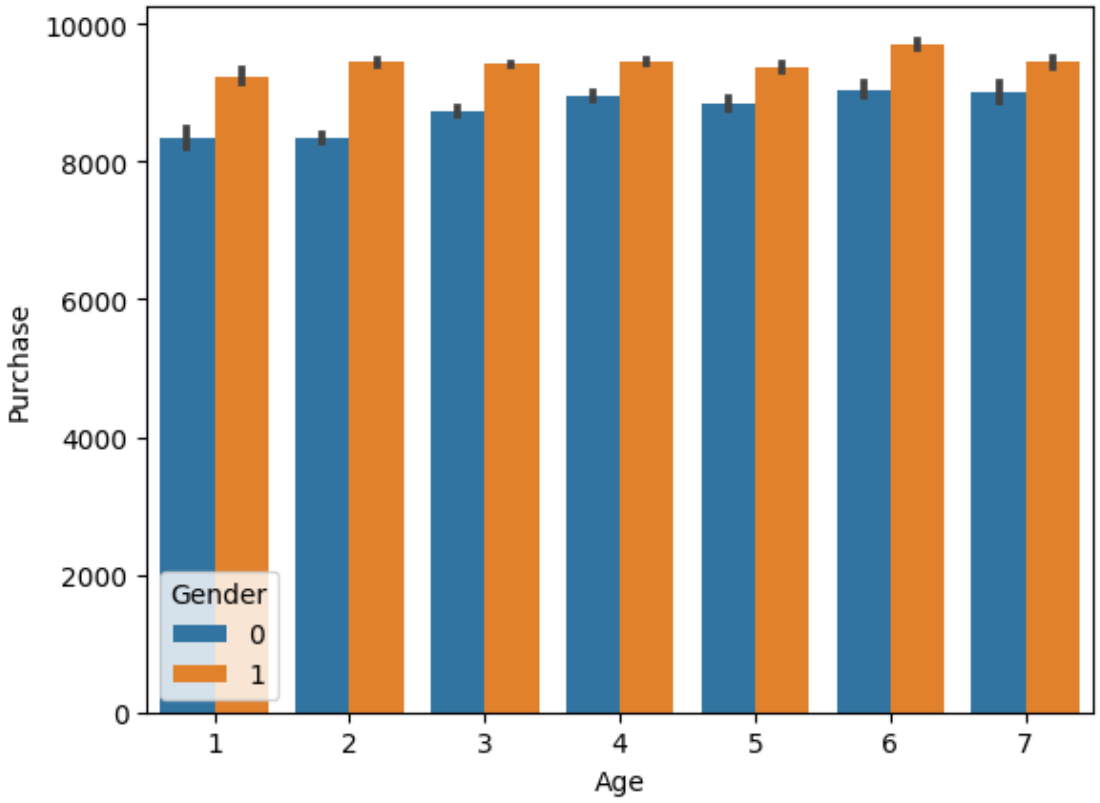
[42]:

	Product_ID	Gender	Age	Occupation	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3
0	P00069042	0	1	10	2	0	3	8.0	16.0
1	P00248942	0	1	10	2	0	1	6.0	14.0
2	P00087842	0	1	10	2	0	12	8.0	16.0
3	P00085442	0	1	10	2	0	12	14.0	16.0
4	P00285442	1	7	16	4	0	8	8.0	16.0

```
   Purchase  B  C
0    8370.0  0  0
1   15200.0  0  0
2    1422.0  0  0
3    1057.0  0  0
4    7969.0  0  1
```

[43]:
```
## visulization  age vs purchase w.r.t gender
sns.barplot(x='Age', y='Purchase', hue='Gender', data=df)
```

[43]: <Axes: xlabel='Age', ylabel='Purchase'>

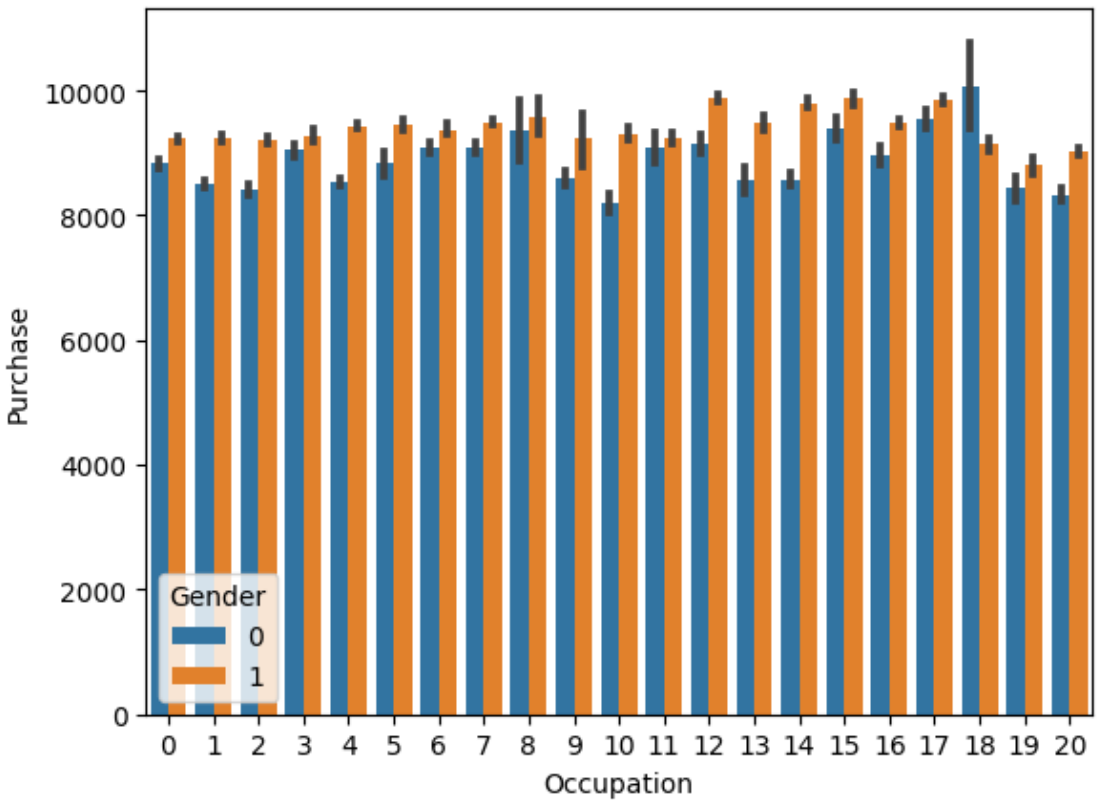


0.4 Observation :

from above Observation mens buys more tha womens

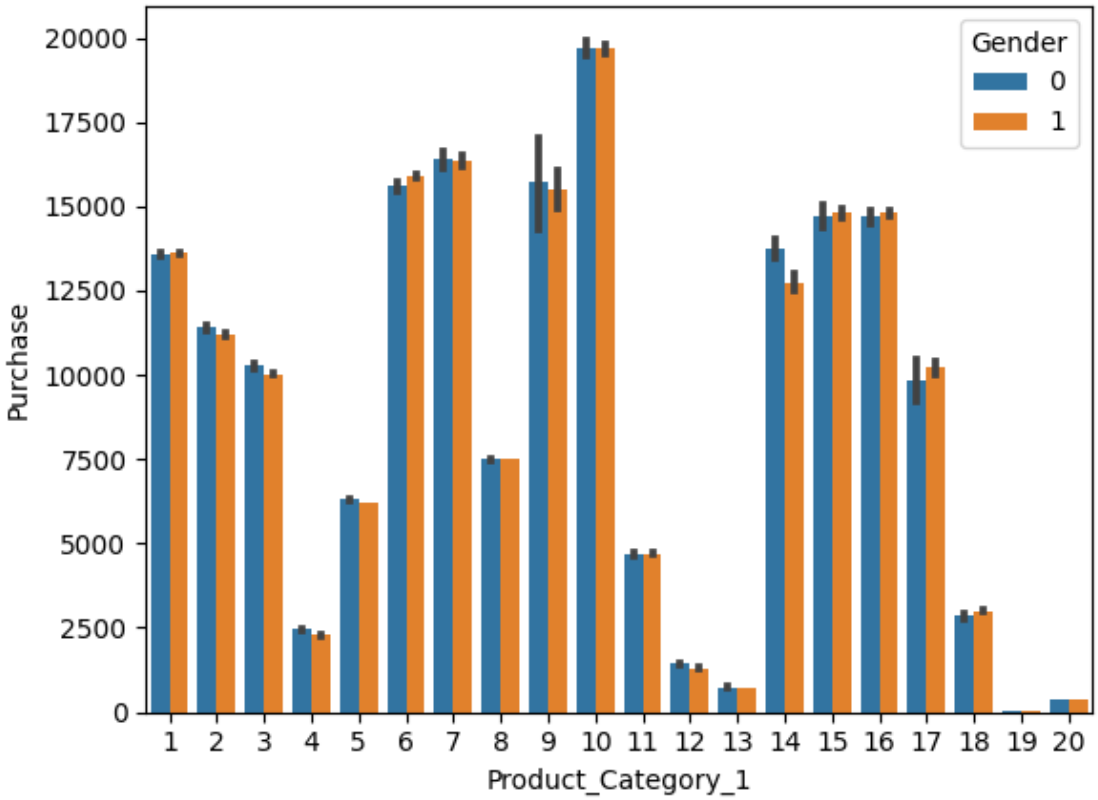
[44]:
```
sns.barplot(x='Occupation', y='Purchase', hue='Gender', data=df)
```

[44]: <Axes: xlabel='Occupation', ylabel='Purchase'>



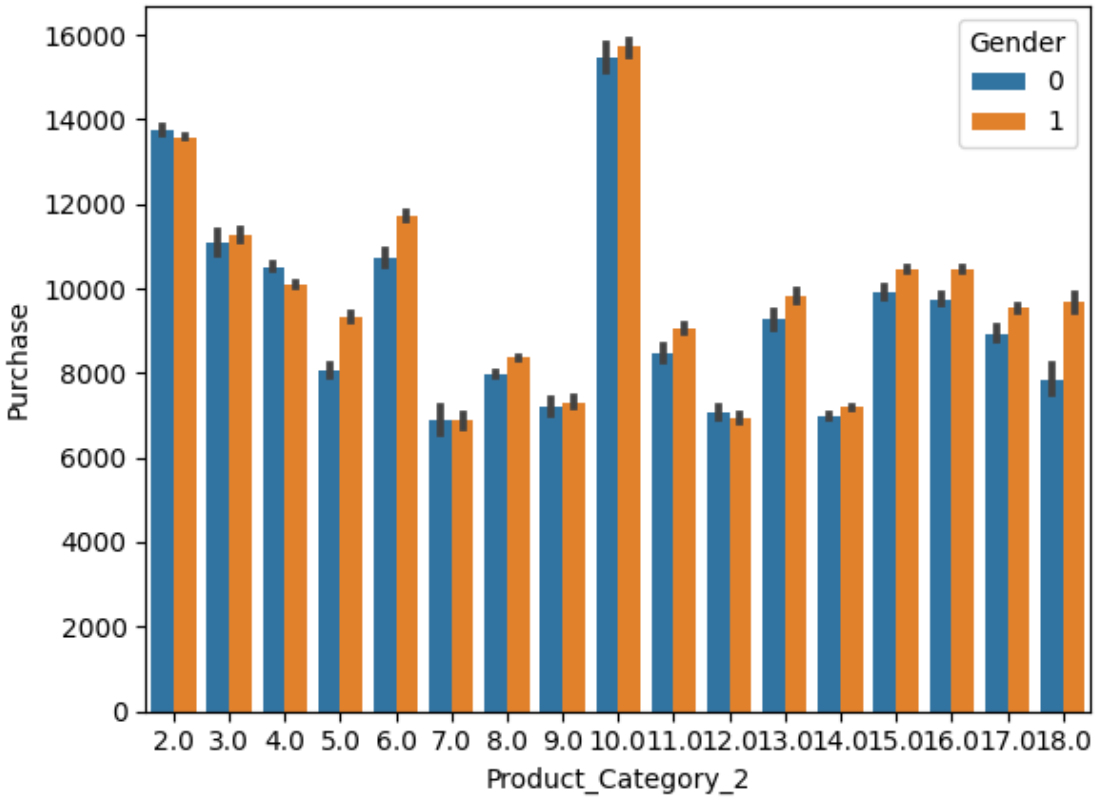
[45]:
```
sns.barplot(x='Product_Category_1', y='Purchase', hue='Gender', data=df)
```

[45]:
```
<Axes: xlabel='Product_Category_1', ylabel='Purchase'>
```



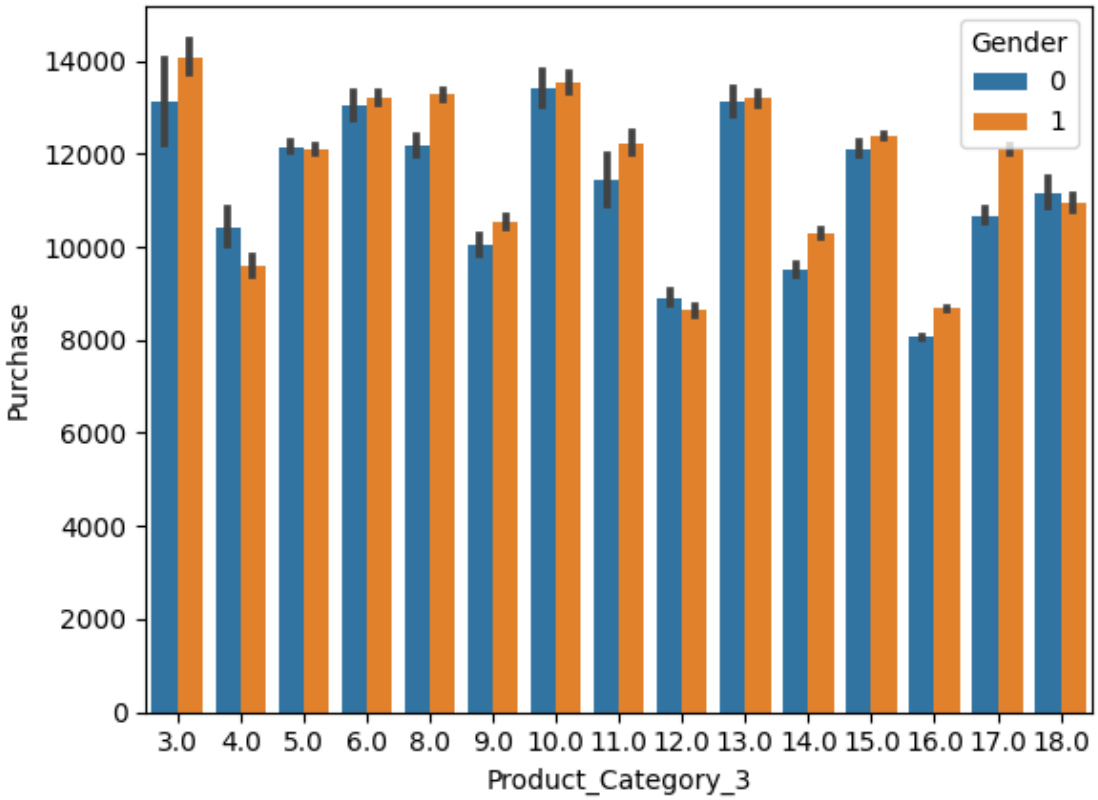
[46]:
```
sns.barplot(x='Product_Category_2', y='Purchase', hue='Gender', data=df)
```

[46]:
```
<Axes: xlabel='Product_Category_2', ylabel='Purchase'>
```



[47]:
```
sns.barplot(x='Product_Category_3', y='Purchase', hue='Gender', data=df)
```

[47]:
```
<Axes: xlabel='Product_Category_3', ylabel='Purchase'>
```



[48]:
```
df.head()
```

[48]:

	Product_ID	Gender	Age	Occupation	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase	B	C
0	P00069042	0	1	10	2	0	3	8.0	16.0	8370.0	0	0
1	P00248942	0	1	10	2	0	1	6.0	14.0	15200.0	0	0
2	P00087842	0	1	10	2	0	12	8.0	16.0	1422.0	0	0
3	P00085442	0	1	10	2	0	12	14.0	16.0	1057.0	0	0
4	P00285442	1	7	16	4	0	8	8.0	16.0			

```
4    7969.0  0  1
```

[76]:
```
# Feacture Scaling
df_test=df[df["Purchase"].isnull()]
```

[77]:
```
df_train=df[~df["Purchase"].isnull()]
```

[78]:
```
X=df_train.drop("Purchase",axis=1)
```

[79]:
```
X.head()
```

[79]:

	Product_ID	Gender	Age	Occupation	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	B	C
0	P00069042	0	1	10	2	0	3	8.0	16.0	0	0
1	P00248942	0	1	10	2	0	1	6.0	14.0	0	0
2	P00087842	0	1	10	2	0	12	8.0	16.0	0	0
3	P00085442	0	1	10	2	0	12	14.0	16.0	0	0
4	P00285442	1	7	16	4	0	8	8.0	16.0	0	1

[83]:
```
y=df_train["Purchase"]
```

[84]:
```
X.shape
```

[84]:
```
(550068, 11)
```

[85]:
```
y.shape
```

[85]:
```
(550068,)
```

[86]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
     X, y, test_size=0.33, random_state=42)
```

[87]:
```
X_train.drop('Product_ID',axis=1,inplace=True)
X_test.drop('Product_ID',axis=1,inplace=True)
```

[88]:
```
## feature Scaling
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)
```

[ ]:
```

```