**Black Friday Dataset EDA And Feature Engineering**

**Cleaning and preparing the data for model training**

In [3]:

*## dataset link: https://www.kaggle.com/sdolezel/black-friday?select=train.csv*

**import** pandas **as** pd

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**%matplotlib** inline

**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.

In [2]:

*#importing the dataset*

df\_train**=**pd**.**read\_csv('blackFriday\_train.csv')

df\_train**.**head()

Out[2]:

|  | **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 |
| **1** | 1000001 | P00248942 | F | 0-17 | 10 | A | 2 | 0 | 1 | 6.0 | 14.0 | 15200 |
| **2** | 1000001 | P00087842 | F | 0-17 | 10 | A | 2 | 0 | 12 | NaN | NaN | 1422 |
| **3** | 1000001 | P00085442 | F | 0-17 | 10 | A | 2 | 0 | 12 | 14.0 | NaN | 1057 |
| **4** | 1000002 | P00285442 | M | 55+ | 16 | C | 4+ | 0 | 8 | NaN | NaN | 7969 |

In [4]:

*## import the test data*

df\_test**=**pd**.**read\_csv('blackFriday\_test.csv')

df\_test**.**head()

Out[4]:

|  | **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** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1000004 | P00128942 | M | 46-50 | 7 | B | 2 | 1 | 1 | 11.0 | NaN |
| **1** | 1000009 | P00113442 | M | 26-35 | 17 | C | 0 | 0 | 3 | 5.0 | NaN |
| **2** | 1000010 | P00288442 | F | 36-45 | 1 | B | 4+ | 1 | 5 | 14.0 | NaN |
| **3** | 1000010 | P00145342 | F | 36-45 | 1 | B | 4+ | 1 | 4 | 9.0 | NaN |
| **4** | 1000011 | P00053842 | F | 26-35 | 1 | C | 1 | 0 | 4 | 5.0 | 12.0 |

In [5]:

*##MErge both train and test data*

df**=**df\_train**.**append(df\_test)

df**.**head()

Out[5]:

|  | **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 |

In [6]:

*##Basic*

df**.**info()

<class 'pandas.core.frame.DataFrame'>

Int64Index: 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

In [7]:

df**.**describe()

Out[7]:

|  | **User\_ID** | **Occupation** | **Marital\_Status** | **Product\_Category\_1** | **Product\_Category\_2** | **Product\_Category\_3** | **Purchase** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **count** | 7.836670e+05 | 783667.000000 | 783667.000000 | 783667.000000 | 537685.000000 | 237858.000000 | 550068.000000 |
| **mean** | 1.003029e+06 | 8.079300 | 0.409777 | 5.366196 | 9.844506 | 12.668605 | 9263.968713 |
| **std** | 1.727267e+03 | 6.522206 | 0.491793 | 3.878160 | 5.089093 | 4.125510 | 5023.065394 |
| **min** | 1.000001e+06 | 0.000000 | 0.000000 | 1.000000 | 2.000000 | 3.000000 | 12.000000 |
| **25%** | 1.001519e+06 | 2.000000 | 0.000000 | 1.000000 | 5.000000 | 9.000000 | 5823.000000 |
| **50%** | 1.003075e+06 | 7.000000 | 0.000000 | 5.000000 | 9.000000 | 14.000000 | 8047.000000 |
| **75%** | 1.004478e+06 | 14.000000 | 1.000000 | 8.000000 | 15.000000 | 16.000000 | 12054.000000 |
| **max** | 1.006040e+06 | 20.000000 | 1.000000 | 20.000000 | 18.000000 | 18.000000 | 23961.000000 |

In [8]:

df**.**drop(['User\_ID'],axis**=**1,inplace**=True**)

In [9]:

df**.**head()

Out[9]:

|  | **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** | P00069042 | F | 0-17 | 10 | A | 2 | 0 | 3 | NaN | NaN | 8370.0 |
| **1** | P00248942 | F | 0-17 | 10 | A | 2 | 0 | 1 | 6.0 | 14.0 | 15200.0 |
| **2** | P00087842 | F | 0-17 | 10 | A | 2 | 0 | 12 | NaN | NaN | 1422.0 |
| **3** | P00085442 | F | 0-17 | 10 | A | 2 | 0 | 12 | 14.0 | NaN | 1057.0 |
| **4** | P00285442 | M | 55+ | 16 | C | 4+ | 0 | 8 | NaN | NaN | 7969.0 |

In [11]:

df['Gender']**=**pd**.**get\_dummies(df['Gender'],drop\_first**=**1)

Out[11]:

|  | **M** |
| --- | --- |
| **0** | 0 |
| **1** | 0 |
| **2** | 0 |
| **3** | 0 |
| **4** | 1 |
| **...** | ... |
| **233594** | 0 |
| **233595** | 0 |
| **233596** | 0 |
| **233597** | 0 |
| **233598** | 0 |

783667 rows × 1 columns

In [12]:

*##HAndling categorical feature Gender*

df['Gender']**=**df['Gender']**.**map({'F':0,'M':1})

df**.**head()

Out[12]:

|  | **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** | P00069042 | 0 | 0-17 | 10 | A | 2 | 0 | 3 | NaN | NaN | 8370.0 |
| **1** | P00248942 | 0 | 0-17 | 10 | A | 2 | 0 | 1 | 6.0 | 14.0 | 15200.0 |
| **2** | P00087842 | 0 | 0-17 | 10 | A | 2 | 0 | 12 | NaN | NaN | 1422.0 |
| **3** | P00085442 | 0 | 0-17 | 10 | A | 2 | 0 | 12 | 14.0 | NaN | 1057.0 |
| **4** | P00285442 | 1 | 55+ | 16 | C | 4+ | 0 | 8 | NaN | NaN | 7969.0 |

In [13]:

*## Handle categorical feature Age*

df['Age']**.**unique()

Out[13]:

array(['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'],

dtype=object)

In [17]:

*#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})

In [ ]:

*##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()

In [18]:

df**.**head()

Out[18]:

|  | **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** | P00069042 | 0 | 1 | 10 | A | 2 | 0 | 3 | NaN | NaN | 8370.0 |
| **1** | P00248942 | 0 | 1 | 10 | A | 2 | 0 | 1 | 6.0 | 14.0 | 15200.0 |
| **2** | P00087842 | 0 | 1 | 10 | A | 2 | 0 | 12 | NaN | NaN | 1422.0 |
| **3** | P00085442 | 0 | 1 | 10 | A | 2 | 0 | 12 | 14.0 | NaN | 1057.0 |
| **4** | P00285442 | 1 | 7 | 16 | C | 4+ | 0 | 8 | NaN | NaN | 7969.0 |

In [20]:

*##fixing categorical City\_categort*

df\_city**=**pd**.**get\_dummies(df['City\_Category'],drop\_first**=True**)

In [21]:

df\_city**.**head()

Out[21]:

|  | **B** | **C** |
| --- | --- | --- |
| **0** | 0 | 0 |
| **1** | 0 | 0 |
| **2** | 0 | 0 |
| **3** | 0 | 0 |
| **4** | 0 | 1 |

In [22]:

df**=**pd**.**concat([df,df\_city],axis**=**1)

df**.**head()

Out[22]:

|  | **Product\_ID** | **Gender** | **Age** | **Occupation** | **City\_Category** | **Stay\_In\_Current\_City\_Years** | **Marital\_Status** | **Product\_Category\_1** | **Product\_Category\_2** | **Product\_Category\_3** | **Purchase** | **B** | **C** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | P00069042 | 0 | 1 | 10 | A | 2 | 0 | 3 | NaN | NaN | 8370.0 | 0 | 0 |
| **1** | P00248942 | 0 | 1 | 10 | A | 2 | 0 | 1 | 6.0 | 14.0 | 15200.0 | 0 | 0 |
| **2** | P00087842 | 0 | 1 | 10 | A | 2 | 0 | 12 | NaN | NaN | 1422.0 | 0 | 0 |
| **3** | P00085442 | 0 | 1 | 10 | A | 2 | 0 | 12 | 14.0 | NaN | 1057.0 | 0 | 0 |
| **4** | P00285442 | 1 | 7 | 16 | C | 4+ | 0 | 8 | NaN | NaN | 7969.0 | 0 | 1 |

In [25]:

*##drop City Category Feature*

df**.**drop('City\_Category',axis**=**1,inplace**=True**)

In [26]:

df**.**head()

Out[26]:

|  | **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 | NaN | 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 | NaN | NaN | 1422.0 | 0 | 0 |
| **3** | P00085442 | 0 | 1 | 10 | 2 | 0 | 12 | 14.0 | NaN | 1057.0 | 0 | 0 |
| **4** | P00285442 | 1 | 7 | 16 | 4+ | 0 | 8 | NaN | NaN | 7969.0 | 0 | 1 |

In [28]:

*## Missing Values*

df**.**isnull()**.**sum()

Out[28]:

Product\_ID 0

Gender 0

Age 0

Occupation 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

In [30]:

*## Focus on replacing missing values*

df['Product\_Category\_2']**.**unique()

Out[30]:

array([nan, 6., 14., 2., 8., 15., 16., 11., 5., 3., 4., 12., 9.,

10., 17., 13., 7., 18.])

In [31]:

df['Product\_Category\_2']**.**value\_counts()

Out[31]:

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: Product\_Category\_2, dtype: int64

In [34]:

df['Product\_Category\_2']**.**mode()[0]

Out[34]:

8.0

In [35]:

*## Replace the missing values with mode*

df['Product\_Category\_2']**=**df['Product\_Category\_2']**.**fillna(df['Product\_Category\_2']**.**mode()[0])

In [36]:

df['Product\_Category\_2']**.**isnull()**.**sum()

Out[36]:

0

In [37]:

*## Product\_category 3 replace missing values*

df['Product\_Category\_3']**.**unique()

Out[37]:

array([nan, 14., 17., 5., 4., 16., 15., 8., 9., 13., 6., 12., 3.,

18., 11., 10.])

In [38]:

df['Product\_Category\_3']**.**value\_counts()

Out[38]:

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: Product\_Category\_3, dtype: int64

In [39]:

*## Replace the missing values with mode*

df['Product\_Category\_3']**=**df['Product\_Category\_3']**.**fillna(df['Product\_Category\_3']**.**mode()[0])

In [40]:

df**.**head()

Out[40]:

|  | **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 | 7969.0 | 0 | 1 |

In [41]:

df**.**shape

Out[41]:

(783667, 12)

In [42]:

df['Stay\_In\_Current\_City\_Years']**.**unique()

Out[42]:

array(['2', '4+', '3', '1', '0'], dtype=object)

In [44]:

df['Stay\_In\_Current\_City\_Years']**=**df['Stay\_In\_Current\_City\_Years']**.**str**.**replace('+','')

C:\Users\win10\AppData\Local\Temp/ipykernel\_24288/2063355665.py:1: FutureWarning: The default value of regex will change from True to False in a future version. In addition, single character regular expressions will \*not\* be treated as literal strings when regex=True.

df['Stay\_In\_Current\_City\_Years']=df['Stay\_In\_Current\_City\_Years'].str.replace('+','')

In [45]:

df**.**head()

Out[45]:

|  | **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 | 7969.0 | 0 | 1 |

In [46]:

df**.**info()

<class 'pandas.core.frame.DataFrame'>

Int64Index: 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 uint8

11 C 783667 non-null uint8

dtypes: float64(3), int64(5), object(2), uint8(2)

memory usage: 67.3+ MB

In [47]:

*##convert object into integers*

df['Stay\_In\_Current\_City\_Years']**=**df['Stay\_In\_Current\_City\_Years']**.**astype(int)

df**.**info()

<class 'pandas.core.frame.DataFrame'>

Int64Index: 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 uint8

11 C 783667 non-null uint8

dtypes: float64(3), int32(1), int64(5), object(1), uint8(2)

memory usage: 64.3+ MB

In [48]:

df['B']**=**df['B']**.**astype(int)

df['C']**=**df['C']**.**astype(int)

In [49]:

df**.**info()

<class 'pandas.core.frame.DataFrame'>

Int64Index: 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

In [53]:

*##Visualisation Age vs Purchased*

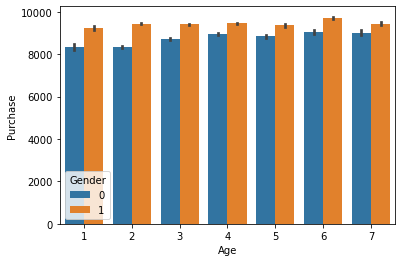
sns**.**barplot('Age','Purchase',hue**=**'Gender',data**=**df)

C:\Users\win10\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(

Out[53]:

<AxesSubplot:xlabel='Age', ylabel='Purchase'>



**Purchasing of men is high then women**

In [54]:

*## Visualization of Purchase with occupation*

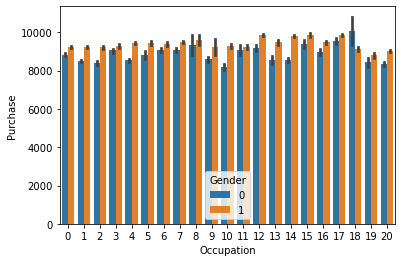
sns**.**barplot('Occupation','Purchase',hue**=**'Gender',data**=**df)

C:\Users\win10\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(

Out[54]:

<AxesSubplot:xlabel='Occupation', ylabel='Purchase'>



In [57]:

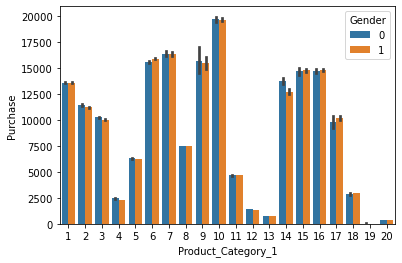
sns**.**barplot('Product\_Category\_1','Purchase',hue**=**'Gender',data**=**df)

C:\Users\win10\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(

Out[57]:

<AxesSubplot:xlabel='Product\_Category\_1', ylabel='Purchase'>



In [58]:

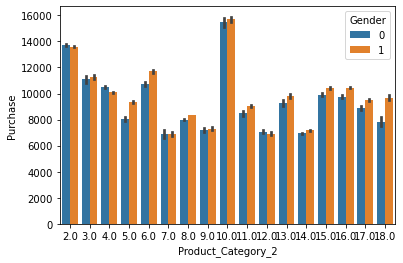
sns**.**barplot('Product\_Category\_2','Purchase',hue**=**'Gender',data**=**df)

C:\Users\win10\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(

Out[58]:

<AxesSubplot:xlabel='Product\_Category\_2', ylabel='Purchase'>



In [59]:

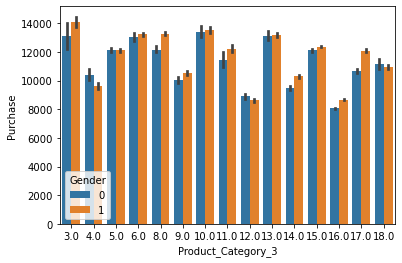
sns**.**barplot('Product\_Category\_3','Purchase',hue**=**'Gender',data**=**df)

C:\Users\win10\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(

Out[59]:

<AxesSubplot:xlabel='Product\_Category\_3', ylabel='Purchase'>



In [60]:

df**.**head()

Out[60]:

|  | **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 | 7969.0 | 0 | 1 |

In [64]:

*##Feature Scaling*

df\_test**=**df[df['Purchase']**.**isnull()]

In [67]:

df\_train**=**df[**~**df['Purchase']**.**isnull()]

In [84]:

X**=**df\_train**.**drop('Purchase',axis**=**1)

In [85]:

X**.**head()

Out[85]:

|  | **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 |

In [86]:

X**.**shape

Out[86]:

(550068, 11)

In [87]:

y**=**df\_train['Purchase']

In [88]:

y**.**shape

Out[88]:

(550068,)

In [83]:

y

Out[83]:

0 8370.0

1 15200.0

2 1422.0

3 1057.0

4 7969.0

...

550063 368.0

550064 371.0

550065 137.0

550066 365.0

550067 490.0

Name: Purchase, Length: 550068, dtype: float64

In [89]:

**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)

In [91]:

X\_train**.**drop('Product\_ID',axis**=**1,inplace**=True**)

X\_test**.**drop('Product\_ID',axis**=**1,inplace**=True**)

C:\Users\win10\anaconda3\lib\site-packages\pandas\core\frame.py:4906: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy

return super().drop(

In [92]:

*## feature Scaling*

**from** sklearn.preprocessing **import** StandardScaler

sc**=**StandardScaler()

X\_train**=**sc**.**fit\_transform(X\_train)

X\_test**=**sc**.**transform(X\_test)

In [ ]:

*## train ur model*