Untitled187

August 15, 2024

1 Black friday dataset EDA and feature engineering

1.0.1 cleaning and preprocessing data for model training

[4]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

[13]:
```
## import the dataset
train_data = pd.read_csv("train.csv")
```

[14]:
```
train_data.head()
```

[14]:
```
   User_ID Product_ID Gender   Age  Occupation City_Category  \
0  1000001  P00069042      F  0-17          10             A
1  1000001  P00248942      F  0-17          10             A
2  1000001  P00087842      F  0-17          10             A
3  1000001  P00085442      F  0-17          10             A
4  1000002  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
1                 6.0                14.0     15200
2                 NaN                 NaN      1422
3                14.0                 NaN      1057
4                 NaN                 NaN      7969
```

2 problem statement is predict the purchase amount of customer

[15]:
```python
test_data = pd.read_csv("test.csv")
```

[16]:
```python
test_data.head()
```

[16]:
```
   User_ID Product_ID Gender   Age  Occupation City_Category  \
0  1000004  P00128942      M  46-50           7             B
1  1000009  P00113442      M  26-35          17             C
2  1000010  P00288442      F  36-45           1             B
3  1000010  P00145342      F  36-45           1             B
4  1000011  P00053842      F  26-35           1             C

  Stay_In_Current_City_Years  Marital_Status  Product_Category_1  \
0                          2               1                   1
1                          0               0                   3
2                         4+               1                   5
3                         4+               1                   4
4                          1               0                   4

   Product_Category_2  Product_Category_3
0                11.0                 NaN
1                 5.0                 NaN
2                14.0                 NaN
3                 9.0                 NaN
4                 5.0                12.0
```

[18]:
```python
## merge both yrain and test data
data =train_data.append(test_data)
data.head()
```

```
C:\Users\Vikas\AppData\Local\Temp\ipykernel_23308\2230856511.py:2:
FutureWarning: The frame.append method is deprecated and will be removed from
pandas in a future version. Use pandas.concat instead.
  data =train_data.append(test_data)
```

[18]:
```
   User_ID Product_ID Gender   Age  Occupation City_Category  \
0  1000001  P00069042      F  0-17          10             A
1  1000001  P00248942      F  0-17          10             A
2  1000001  P00087842      F  0-17          10             A
3  1000001  P00085442      F  0-17          10             A
4  1000002  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
```

[19]:
```
##Basic
data.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
```

[20]:
```
data.drop(["User_ID"],axis=1,inplace=True)
```

[21]:
```
pd.get_dummies(data["Gender"])
```

[21]:
```
        F  M
0       1  0
1       1  0
2       1  0
3       1  0
4       0  1
…      .. ..
233594  1  0
233595  1  0
233596  1  0
233597  1  0
233598  1  0
```

```
[783667 rows x 2 columns]
```

[22]:
```
## Handling categorical feature  Gender
data['Gender'] = data['Gender'].map({'F':0,'M':1})
data.head()
```

[22]:
```
  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
```

[23]:
```
##Handle categorical feature Age
data['Age'].unique()
```

[23]:
```
array(['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'],
      dtype=object)
```

[25]:
```
## pd.get_dummies(data['Age'],drop_first=True)
data['Age']=data['Age'].map({'0-17':1,'18-25':2,'36-46':3,'46-50':4,'51-55':
 ↪5,'55+':7})
```

[27]:
```
## Second technic
from sklearn import preprocessing

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

##encoder label in column  "species"
data['Age']=label_encoder.fit_transform(data['Age'])
```

```
data['Age'].unique()
```

[27]:
```
array([0, 4, 5, 2, 3, 1], dtype=int64)
```

[28]:
```
data.head()
```

[28]:
```
  Product_ID  Gender  Age  Occupation City_Category  \
0  P00069042       0    0          10             A
1  P00248942       0    0          10             A
2  P00087842       0    0          10             A
3  P00085442       0    0          10             A
4  P00285442       1    4          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
```

[29]:
```
## fixing categorical City_category
data_city = pd.get_dummies(data['City_Category'],drop_first=True)
```

[30]:
```
data_city.head()
```

[30]:
```
   B  C
0  0  0
1  0  0
2  0  0
3  0  0
4  0  1
```

[31]:
```
data=pd.concat([data,data_city],axis=1)
data.head()
```

[31]:
```
  Product_ID  Gender  Age  Occupation City_Category  \
0  P00069042       0    0          10             A
1  P00248942       0    0          10             A
2  P00087842       0    0          10             A
3  P00085442       0    0          10             A
4  P00285442       1    4          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
```

[32]:
```
## drop city category feature
data.drop('City_Category',axis=1,inplace=True)
```

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

[34]:
```
  Product_ID  Gender  Age  Occupation Stay_In_Current_City_Years  \
0  P00069042       0    0          10                          2
1  P00248942       0    0          10                          2
2  P00087842       0    0          10                          2
3  P00085442       0    0          10                          2
4  P00285442       1    4          16                         4+

   Marital_Status  Product_Category_1  Product_Category_2  Product_Category_3  \
0               0                   3                 NaN                 NaN
1               0                   1                 6.0                14.0
2               0                  12                 NaN                 NaN
3               0                  12                14.0                 NaN
4               0                   8                 NaN                 NaN

   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]:
```
## Missing value
data.isnull().sum()
```

[35]:
```
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
```

[36]:
```
## Focus on replacing missing values
data['Product_Category_2'].unique()
```

[36]:
```
array([nan,  6., 14.,  2.,  8., 15., 16., 11.,  5.,  3.,  4., 12.,  9.,
       10., 17., 13.,  7., 18.])
```

[37]:
```
data['Product_Category_2'].value_counts()
```

[37]:
```
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
```

[42]:
```
##Replace the missing values with mode
data['Product_Category_2']=data['Product_Category_2'].
 ↪fillna(data['Product_Category_2'].mode()[0])
```

[44]:
```
data['Product_Category_2'].isnull().sum()
```

[44]:
```
0
```

[45]:
```
## product_category 3 replace missing values
data['Product_Category_3'].unique()
```

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

```
[46]: data['Product_Category_3'].value_counts()
```

```
[46]: 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
```

```
[48]: data['Product_Category_3']=data['Product_Category_3'].
    ↪fillna(data['Product_Category_3'].mode([0]))
```

```
[49]: data['Product_Category_3'].isnull().sum()
```

```
[49]: 545807
```

```
[50]: data.head()
```

```
[50]:   Product_ID  Gender  Age  Occupation Stay_In_Current_City_Years  \
0  P00069042       0    0          10                          2
1  P00248942       0    0          10                          2
2  P00087842       0    0          10                          2
3  P00085442       0    0          10                          2
4  P00285442       1    4          16                         4+

   Marital_Status  Product_Category_1  Product_Category_2  Product_Category_3  \
0               0                   3                 8.0                16.0
1               0                   1                 6.0                14.0
2               0                  12                 8.0                 NaN
3               0                  12                14.0                 NaN
4               0                   8                 8.0                 NaN

   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
```

[51]:
```python
data.shape
```

[51]:
```
(783667, 12)
```

[52]:
```python
data['Stay_In_Current_City_Years'].unique()
```

[52]:
```
array(['2', '4+', '3', '1', '0'], dtype=object)
```

[53]:
```python
data['Stay_In_Current_City_Years']=data['Stay_In_Current_City_Years'].str.
 ↪replace('+','')
```

```
C:\Users\Vikas\AppData\Local\Temp\ipykernel_23308\1369221623.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.
  data['Stay_In_Current_City_Years']=data['Stay_In_Current_City_Years'].str.repl
ace('+','')
```

[54]:
```python
data.head()
```

[54]:
```
  Product_ID  Gender  Age  Occupation Stay_In_Current_City_Years  \
0  P00069042       0    0          10                          2
1  P00248942       0    0          10                          2
2  P00087842       0    0          10                          2
3  P00085442       0    0          10                          2
4  P00285442       1    4          16                          4

   Marital_Status  Product_Category_1  Product_Category_2  Product_Category_3  \
0               0                   3                 8.0                16.0
1               0                   1                 6.0                14.0
2               0                  12                 8.0                 NaN
3               0                  12                14.0                 NaN
4               0                   8                 8.0                 NaN

   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
```

[55]:
```python
data.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          237860 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
```

[56]:
```
## Stay_In_Current_City_Years convert this col object into integer
data['Stay_In_Current_City_Years']=data['Stay_In_Current_City_Years'].
 ↪astype(int)
data.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          237860 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
```

[57]:
```
data['B']=data['B'].astype(int)
data['C']=data['C'].astype(int)
```

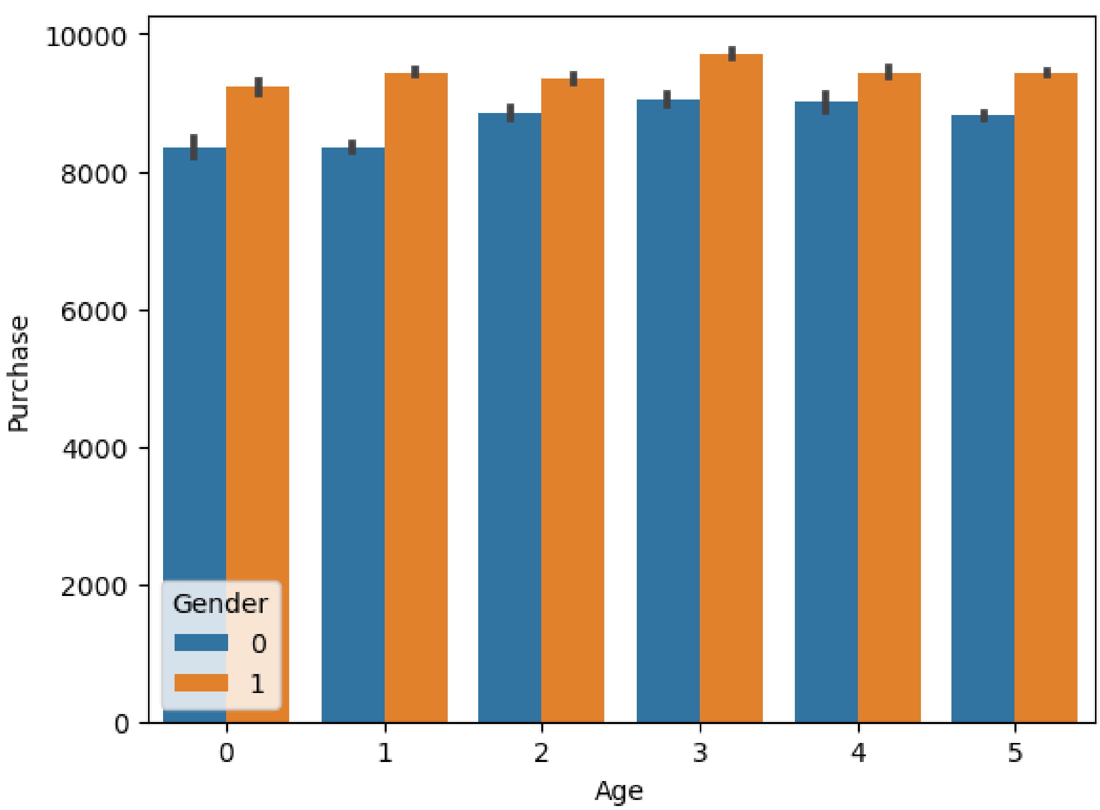
[58]:
```
data.info()
```

```
<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          237860 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
```

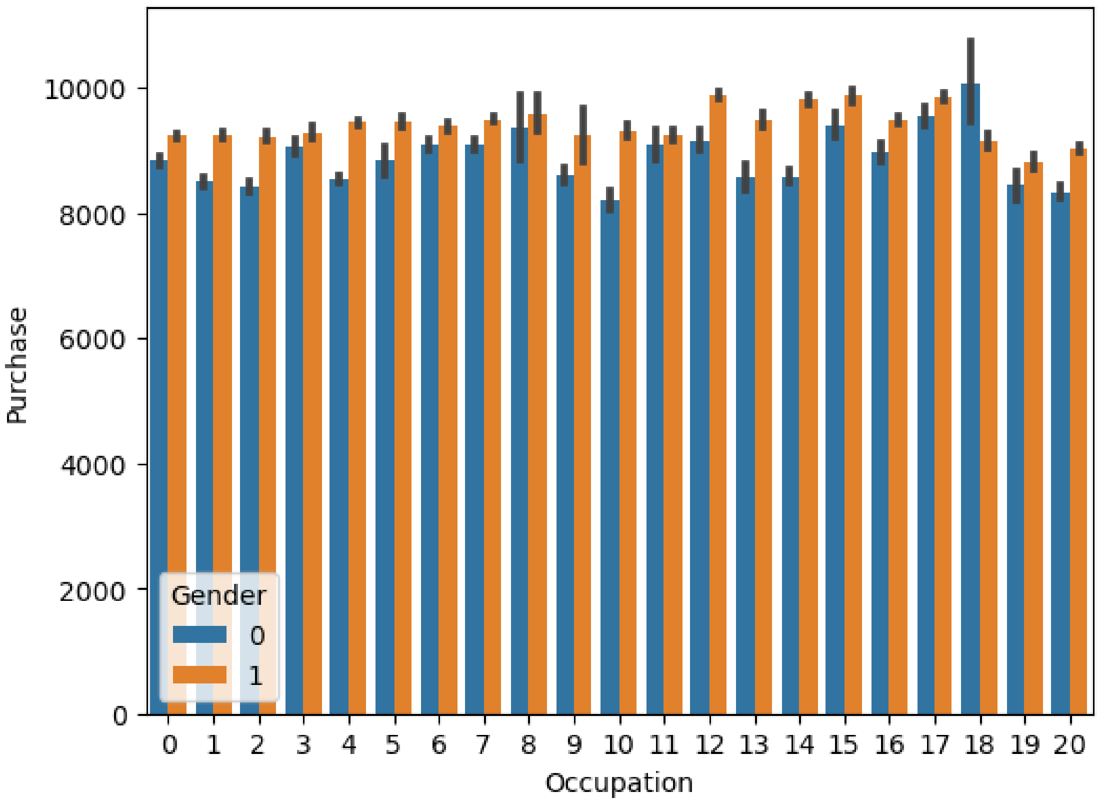
[65]:

```python
## Visualisations AGE VS PURCHASE
sns.barplot(x='Age', y='Purchase', hue='Gender', data=data)

# Display the plot
plt.show()
```

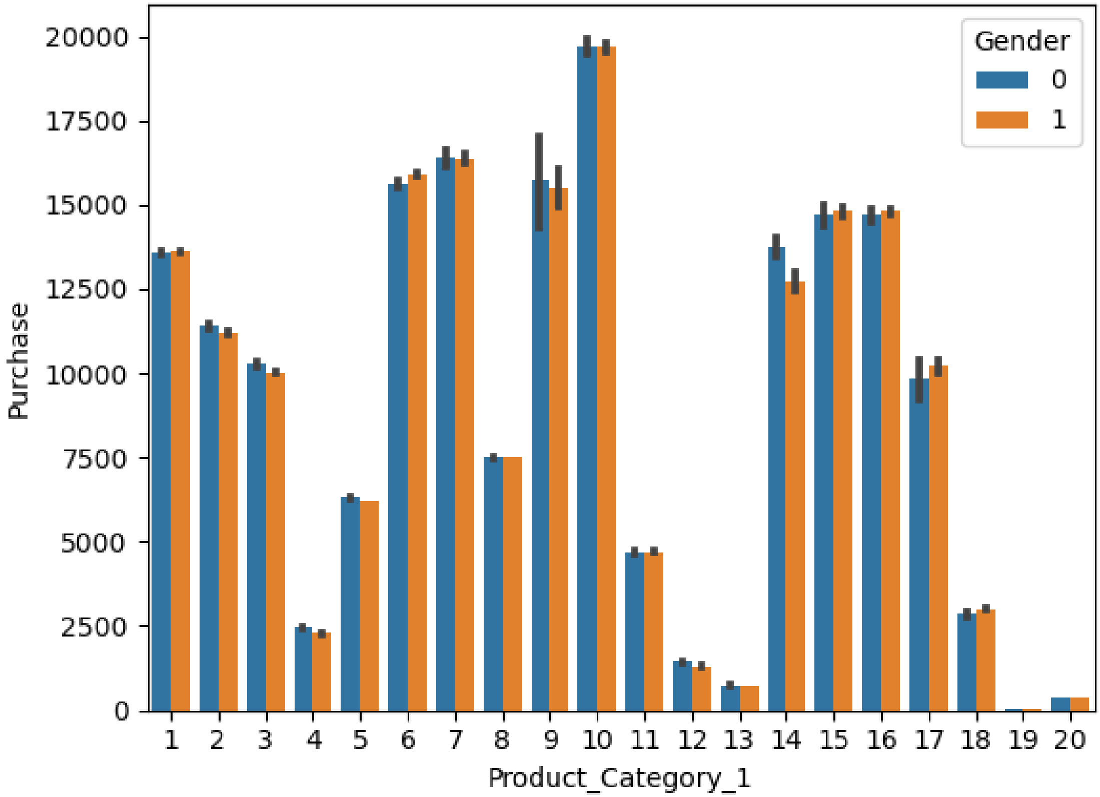


3 Purchasing of men is high then women

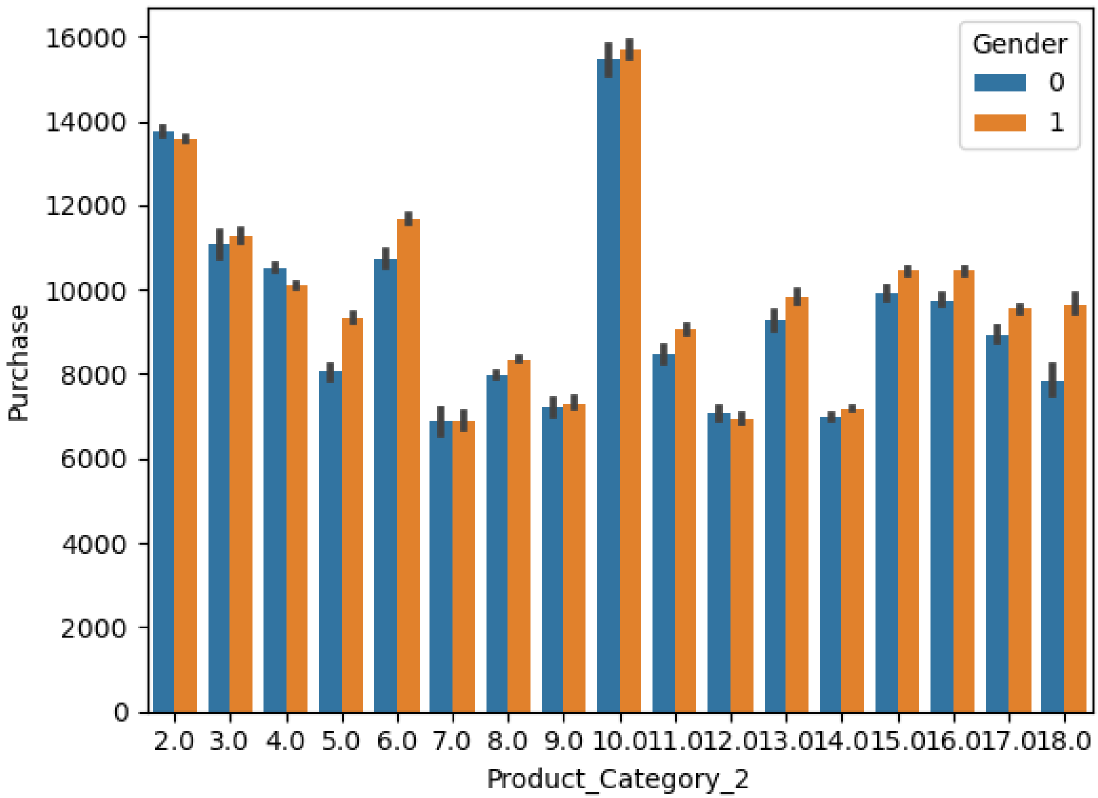
[69]:
```
## Visualisation purchase vs occupation
sns.barplot(x='Occupation', y='Purchase', hue='Gender', data=data)

# Display the plot
plt.show()
```



[70]:
```
## Visualisation purchase vs product category1
sns.barplot(x='Product_Category_1', y='Purchase', hue='Gender', data=data)

# Display the plot
plt.show()
```



[71]:
```
## Visualisation purchase vs product category2
sns.barplot(x='Product_Category_2', y='Purchase', hue='Gender', data=data)

# Display the plot
plt.show()
```



[85]:
```
## Feature scalling
df_test=data[data['Purchase'].isnull()]
```

[86]:
```
df_train=data[~data['Purchase'].isnull()]
```

[87]:
```
X=df_train.drop('Purchase',axis=1)
```

[88]:
```
X.head()
```

[88]:
```
  Product_ID  Gender  Age  Occupation  Stay_In_Current_City_Years  \
0  P00069042       0    0          10                           2
1  P00248942       0    0          10                           2
2  P00087842       0    0          10                           2
3  P00085442       0    0          10                           2
4  P00285442       1    4          16                           4

   Marital_Status  Product_Category_1  Product_Category_2  Product_Category_3  \
0               0                   3                 8.0                16.0
1               0                   1                 6.0                14.0
2               0                  12                 8.0                 NaN
3               0                  12                14.0                 NaN
```

```
4              0                   8                 8.0                NaN

   B  C
0  0  0
1  0  0
2  0  0
3  0  0
4  0  1
```

[94]:
```python
y=df_train['Purchase']
```

[95]:
```python
y
```

[95]:
```
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
```

[96]:
```python
from sklearn.model_selection import train_test_split

# Assuming X is your feature matrix and y is your target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,␣
↪random_state=42)
```

[99]:
```python
X_train.drop('Product_ID',axis=1,inplace=True)
X_test.drop('Product_ID',axis=1,inplace=True)
```

[100]:
```python
## Feature scalling
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.fit_transform(X_test)
```

[ ]:
```python
## train ur model
```