Ab Oliveira and 1 collaborator

Analysis of Black Friday

6 voters

last run 14 days ago · IPython Notebook HTML · 729 views
using data from Black Friday · Public

Tags data visualization tutorial feature engineering data cleaning

Notebook

Black Friday

A study of sales trough consumer behaviours

https://www.kaggle.com/mehdidag/black-friday
(https://www.kaggle.com/mehdidag/black-friday)

Dataset of 550 000 observations about the black Friday in a retail store, it contains different kinds of variables either numerical and categorical.

This Kernel is still in progress, and we appreciate any constructive insights. If you found this helpful, please give this Kernel an upvote, as it keeps up motivated to continue to progress and share with the community.

Libraries

We will be using the Pandas, Numpy, Seaborn, and Matplotlib Python libraries for this analysis.

In [1]:

```
# Warnings
import warnings
warnings.filterwarnings('ignore')

# Data and analysis
import pandas as pd
import numpy as np

# Visualization
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import os
print(os.listdir("../input"))

sns.set(style='darkgrid')
plt.rcParams["patch.force_edgecolor"] = True
```

```
['BlackFriday.csv']
```

Data Import and Feature Exploration

Let's import the data into a Pandas dataframe and check out some of its broader aspects to see what we're working with.

In [2]:

```
df = pd.read_csv('../input/BlackFriday.csv')
```

In [3]:

```
# First 5 rows:
df.head(5)
```

Out[3]:

	User_ID	Product_ID	Gender	Age	Occupation	C
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

In [4]:

```
print(df.info())
print('Shape: ',df.shape)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 537577 entries, 0 to 537576
Data columns (total 12 columns):
User_ID                      537577 non-
null int64
Product_ID                   537577 non-
null object
Gender                       537577 non-
null object
Age                          537577 non-
null object
Occupation                   537577 non-
null int64
City_Category                537577 non-
null object
```

```
Stay_In_Current_City_Years    537577 non-
null object
Marital_Status                537577 non-
null int64
Product_Category_1            537577 non-
null int64
Product_Category_2            370591 non-
null float64
Product_Category_3            164278 non-
null float64
Purchase                      537577 non-
null int64
dtypes: float64(2), int64(5), object(5)
memory usage: 49.2+ MB
None
Shape:  (537577, 12)
```

Missing Values

In [5]:

```
total_miss = df.isnull().sum()
perc_miss = total_miss/df.isnull().count()*100

missing_data = pd.DataFrame({'Total missing':total_mi
ss,
                            '% missing':perc_miss})

missing_data.sort_values(by='Total missing',
                         ascending=False).head(3)
```

Out[5]:

	Total missing	% missing
Product_Category_3	373299	69.441029
Product_Category_2	166986	31.062713
User_ID	0	0.000000

Since most products will belong to only one category, it makes sense for less products to have a second category, let alone a third one.

Unique Values

Lets now explore the unique values in some of the features.
Remember there is a total of 537577 entries:

In [6]:

```
print('Unique Values for Each Feature: \n')
for i in df.columns:
    print(i, ':',df[i].nunique())
```

```
Unique Values for Each Feature:

User_ID : 5891
Product_ID : 3623
Gender : 2
Age : 7
Occupation : 21
City_Category : 3
Stay_In_Current_City_Years : 5
Marital_Status : 2
Product_Category_1 : 18
Product_Category_2 : 17
Product_Category_3 : 15
Purchase : 17959
```

In [7]:

```
# Info about products
print('Number of products:',df['Product_ID'].nunique
())
print('Number of categories:',df['Product_Category_1'
].unique().max())
print('Highest and lowest purchase:',
      df['Purchase'].max(),',',df['Purchase'].min())
```

```
Number of products: 3623
Number of categories: 18
Highest and lowest purchase: 23961 , 185
```

In [8]:

```
# Info about shoppers
print('Number of shoppers:',df['User_ID'].nunique())
print('Years in city:',df['Stay_In_Current_City_Year
s'].unique())
```

```
print('Age Groups:',df['Age'].unique())
```

```
Number of shoppers: 5891
Years in city: ['2' '4+' '3' '1' '0']
Age Groups: ['0-17' '55+' '26-35' '46-50'
'51-55' '36-45' '18-25']
```

Gender

Lets first find whether the data is uniformly distributed by gender by looking at how many entries belong to each one:

In [9]:

```
count_m = df[df['Gender']=='M'].count()[0]
count_f = df[df['Gender']=='F'].count()[0]
```

In [10]:

```
print('Number of male clients:',count_m)
print('Number of female clients:',count_f)
```

```
Number of male clients: 405380
Number of female clients: 132197
```

We can see that the number of male clients recorded exceeds the number of female clients recorded by almost 4 times. For this reason, it will be much more informational to analyze **Gender** by using ratios instead of counting each entry. Lets see how much each gender spent in regards to eachself:

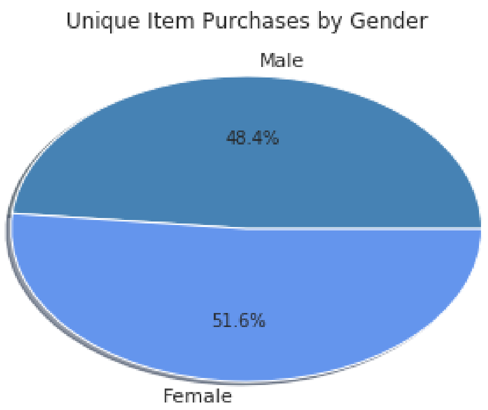
In [11]:

```
print('Female Purchases:',round(df[df['Gender']=='F']
['Purchase'].sum()/count_f,3))
print('Male Purchases:',round(df[df['Gender']=='M'][
'Purchase'].sum()/count_m,3))
```

```
Female Purchases: 8809.761
Male Purchases: 9504.772
```

In [12]:

```
plt.pie(df.groupby('Gender')['Product_ID'].nunique(),
labels=['Male','Female'],
       shadow=True, autopct='%1.1f%%',colors=['steelb
lue','cornflowerblue'])
plt.title('Unique Item Purchases by Gender')
plt.show()
```



Although almost even, women did purchase a slightly wider array of products than men did. Now, lets analyze the proportions of each gender's purchase in terms of the product categories:

In [13]:

```
# Individual groupby dataframes for each gender
gb_gender_m = df[df['Gender']=='M'][['Product_Categor
y_1','Gender']].groupby(by='Product_Category_1').coun
t()
gb_gender_f = df[df['Gender']=='F'][['Product_Categor
y_1','Gender']].groupby(by='Product_Category_1').coun
t()

# Concatenate and change column names
cat_bygender = pd.concat([gb_gender_m,gb_gender_f],ax
is=1)
cat_bygender.columns = ['M ratio','F ratio']

# Adjust to reflect ratios
cat_bygender['M ratio'] = cat_bygender['M ratio']/df[
df['Gender']=='M'].count()[0]
cat_bygender['F ratio'] = cat_bygender['F ratio']/df[
df['Gender']=='F'].count()[0]

# Create likelihood of one gender to buy over the othe
r
cat_bygender['Likelihood (M/F)'] = cat_bygender['M ra
tio']/cat_bygender['F ratio']
```

```python
cat_bygender['Total Ratio'] = cat_bygender['M ratio']
+cat_bygender['F ratio']
```

In [14]:

```python
cat_bygender.sort_values(by='Likelihood (M/F)',ascend
ing=False)
```

Out[14]:

	M ratio	F ratio	Likelihood (M/F)	Tot Rat
Product_Category_1				
17	0.001248	0.000461	2.705079	0.00
18	0.006658	0.002844	2.340854	0.00
15	0.012778	0.007738	1.651252	0.02
9	0.000824	0.000530	1.555993	0.00
1	0.281099	0.184581	1.522908	0.4
11	0.047612	0.035243	1.350972	0.08
6	0.038702	0.033851	1.143303	0.07
10	0.009606	0.008608	1.115868	0.0
2	0.044220	0.042157	1.048947	0.08
16	0.018092	0.017875	1.012130	0.03
7	0.006759	0.007020	0.962857	0.0
13	0.009897	0.010802	0.916204	0.02
5	0.264919	0.311649	0.850058	0.5
8	0.195335	0.249227	0.783766	0.4
3	0.034474	0.044434	0.775849	0.07
4	0.019722	0.027020	0.729905	0.04
12	0.005866	0.011324	0.518023	0.0
14	0.002188	0.004637	0.471870	0.00

This table tells us a lot about how likely a type of product is to be bought in regards of gender. For instance, men are almost 3 times as likely to buy an item in category 17, while women are almost 2 times as likely to buy a product in category 14.

Age

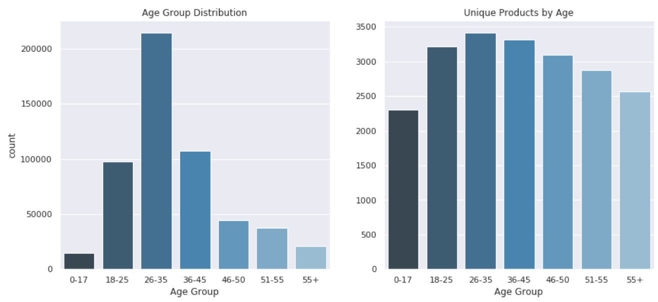
Since as of now, **Age** values are strings, lets encode each group so they can be represented with an integer value which a machine learning algorithm can understand:

In [15]:

```
# Encoding the age groups
df['Age_Encoded'] = df['Age'].map({'0-17':0,'18-25':1
,
                          '26-35':2,'36-45':3,
                          '46-50':4,'51-55':5,
                          '55+':6})
```

In [16]:

```
prod_byage = df.groupby('Age').nunique()['Product_ID'
]

fig,ax = plt.subplots(1,2,figsize=(14,6))
ax = ax.ravel()

sns.countplot(df['Age'].sort_values(),ax=ax[0], palet
te="Blues_d")
ax[0].set_xlabel('Age Group')
ax[0].set_title('Age Group Distribution')
sns.barplot(x=prod_byage.index,y=prod_byage.values,ax
=ax[1], palette="Blues_d")
ax[1].set_xlabel('Age Group')
ax[1].set_title('Unique Products by Age')

plt.show()
```

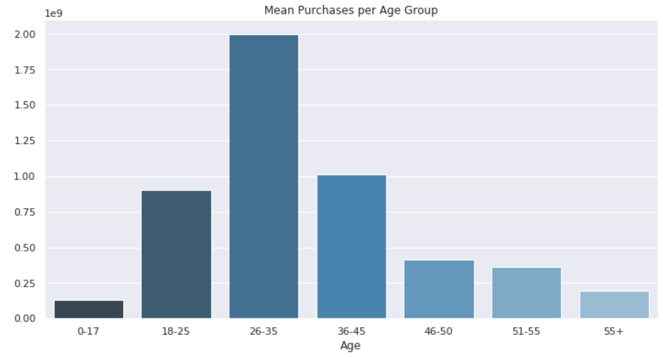


It's quite apparent that the largest age group amongst the customers is 26-35. Interestingly, the distribution of product purchase, in terms of quantity, does not vary greatly amongst the age groups. This means that, though the 26-35 age group is the most popular, the other age groups purchase almost as many unique items as them. But does this mean that the amount of money spent amongst the age groups is the same? Let's see...

In [17]:

```
spent_byage = df.groupby(by='Age').sum()['Purchase']
plt.figure(figsize=(12,6))

sns.barplot(x=spent_byage.index,y=spent_byage.values,
```

```
 palette="Blues_d")
plt.title('Mean Purchases per Age Group')
plt.show()
```



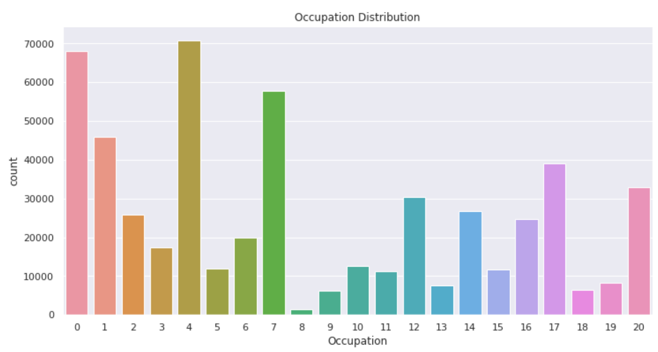
Our data clearly shows that the amount of money made from each age group correlates proportionally with the amount of customers within the age groups. This can be valuable information for the store, as it might want to add more products geared towards this age group in the future, or perhaps work on marketing different items to increase a broader diversity in the age groups of their customers.

Occupation

This sections draws some insights on our data in terms of the occupation of the customers.

In [18]:

```
plt.figure(figsize=(12,6))
sns.countplot(df['Occupation'])
plt.title('Occupation Distribution')
plt.show()
```

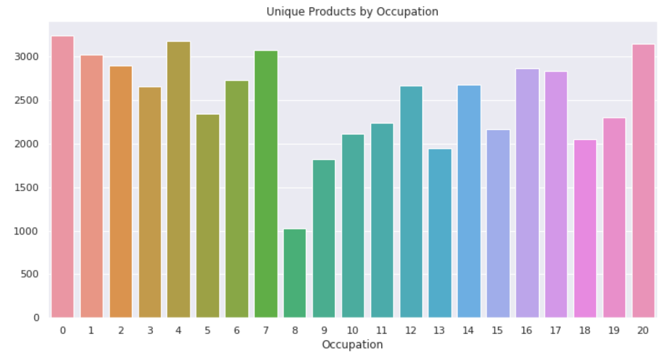


In [ ]:

```

```

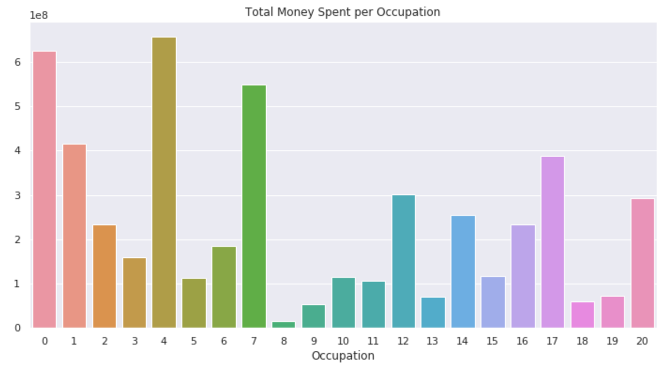
In [19]:

```
plt.figure(figsize=(12,6))
prod_by_occ = df.groupby(by='Occupation').nunique()[
'Product_ID']

sns.barplot(x=prod_by_occ.index,y=prod_by_occ.values)
plt.title('Unique Products by Occupation')
plt.show()
```



In [20]:

```
spent_by_occ = df.groupby(by='Occupation').sum()['Pur
chase']
plt.figure(figsize=(12,6))

sns.barplot(x=spent_by_occ.index,y=spent_by_occ.value
s)
plt.title('Total Money Spent per Occupation')
plt.show()
```



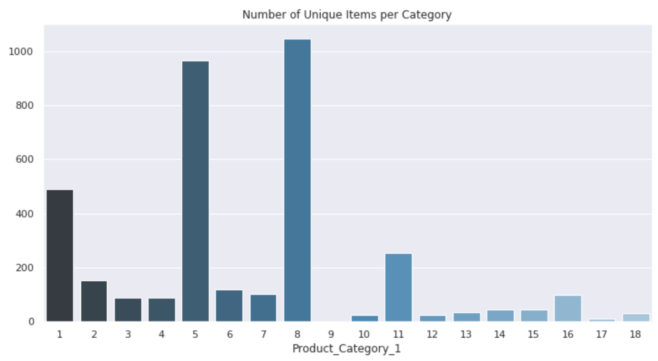
Once again, the distribution of the mean amount spent within each occupation appears to mirror the distribution of the amount of people within each occupation. This is fortunate from a data science perspective, as we are not working with odd or outstanding features. Our data, in terms of age and occupation seems to simply make sense.

Products

Here we explore the products themselves. This is important, as we do not have labeled items in this dataset. Theoretically, a customer could be spending $5,000 on 4 new TVs, or 10,000 pens. This difference matters for stores, as their profits are affected. Since we do not know what the items are, let's explore the categories of the items.

In [21]:

```
plt.figure(figsize=(12,6))
prod_by_cat = df.groupby('Product_Category_1')['Produ
ct_ID'].nunique()

sns.barplot(x=prod_by_cat.index,y=prod_by_cat.values,
 palette="Blues_d")
plt.title('Number of Unique Items per Category')
plt.show()
```

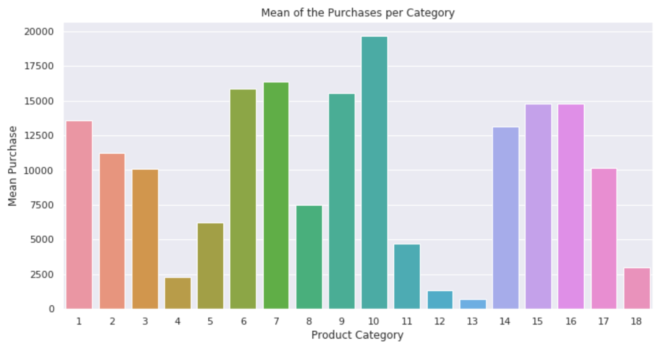


Category labels 1, 5, and 8 clearly have the most items within them. This could mean the store is known for that item, or that the category is a broad one.

In [22]:

```
category = []
mean_purchase = []


for i in df['Product_Category_1'].unique():
    category.append(i)
```

```
category.sort()

for e in category:
    mean_purchase.append(df[df['Product_Category_1']=
=e]['Purchase'].mean())

plt.figure(figsize=(12,6))

sns.barplot(x=category,y=mean_purchase)
plt.title('Mean of the Purchases per Category')
plt.xlabel('Product Category')
plt.ylabel('Mean Purchase')
plt.show()
```



Interestingly enough, our most popular categories are not the ones making the most money. This appears to be a big store, and they may be aware of this. Yet this same form of analysis can be used in the case of a smaller store that might not be aware, and it could be very useful.

Estimate of price and quantity of purchase

 Since the **Purchases** feature alludes to how much a customer paid for an unknown amount of a certain item, let's make a bold assumption that the lowest purchase paid by product is the price of said item:

In [23]:

```
# Dictionary of product IDs with minimum purchase
```

Click here to enter a comment...

Similar Kernels

Tidy TitaRnic

Machine Learning Workflow For House Prices

Extensive EDA Of Deal Probability

Bakery (Market Basket Analysis)

K-Mean Clustering For Wine Quality Data