Tanmay Lata

Black Friday Dataset analysis and Predictions

4
voters

last run a month ago · IPython Notebook HTML · 1,573 views
using data from Black Friday · Public

Notebook

In [1]:

```
# This Python 3 environment comes with many helpful analytics librar
ies installed
# It is defined by the kaggle/python docker image: https://github.co
m/kaggle/docker-python
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_cs
v)
import matplotlib.pyplot as plt
import seaborn as sns

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Ente
r) will list the files in the input directory

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

# Any results you write to the current directory are saved as outpu
t.
```

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

In [2]:

```
#importing the dataset
data = pd.read_csv('../input/BlackFriday.csv')
```

In [3]:

```
data.head()
```

Out[3]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Y
0	1000001	P00069042	F	0-17	10	A	2
1	1000001	P00248942	F	0-17	10	A	2
2	1000001	P00087842	F	0-17	10	A	2
3	1000001	P00085442	F	0-17	10	A	2
4	1000002	P00285442	M	55+	16	C	4+

DATA CLEANING

In [4]:

```
missing_values = data.isnull().sum().sort_values(ascending = False)
missing_values = missing_values[missing_values > 0]/data.shape[0]
print(f'{missing_values *100} %')
```

```
Product_Category_3    69.441029
Product_Category_2    31.062713
dtype: float64 %
```

I believe that the NaN values for **Product_Category_2** and **Product_Categrory_3** would mean that the concerned person did not buy the products from these categories.

Hence, I believe that it would be safe to replace them with 0.

In [5]:

```
data = data.fillna(0)
```

In [6]:

```
missing_values = data.isnull().sum().sort_values(ascending = False)
missing_values = missing_values[missing_values > 0]/data.shape[0]
print(f'{missing_values *100} %')
```

```
Series([], dtype: float64) %
```

So, we have taken care of the missing values. Let's move on and see what all data types are avialable to us inour dataset.

In [7]:

```
data.dtypes
```

Out[7]:

```
User_ID                         int64
Product_ID                     object
Gender                         object
Age                            object
Occupation                      int64
City_Category                  object
Stay_In_Current_City_Years     object
Marital_Status                  int64
Product_Category_1              int64
Product_Category_2            float64
Product_Category_3            float64
Purchase                        int64
dtype: object
```

So, the available datatypes are : int64, float64 and objects. We will leave the numeric datatypes alone and focus on object datatypes as the cannot be directly fen into a Machine Learning Model alone and focus on object datatypes as the cannot be directly fed into a Machine Learning Model

Let's get **Gender** first.

In [8]:
```
#unique values in Gender parameter
gender = np.unique(data['Gender'])
gender
```

Out[8]:
```
array(['F', 'M'], dtype=object)
```

So, we do not have any 'Other' gender type. I will create a fuction and map M=1 and F=0. No sexism intended.

In [9]:
```
def map_gender(gender):
    if gender == 'M':
        return 1
    else:
        return 0
data['Gender'] = data['Gender'].apply(map_gender)
```

Let's see what's cooking in the **Age** parameter

In [10]:
```
age = np.unique(data['Age'])
age
```

Out[10]:
```
array(['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+'],
      dtype=object)
```

So, we are having bins. Lets make these bins into numeric values

In [11]:
```
def map_age(age):
    if age == '0-17':
        return 0
    elif age == '18-25':
        return 1
    elif age == '26-35':
        return 2
    elif age == '36-45':
        return 3
    elif age == '46-50':
```

```
    elif age == '46-50':
        return 4
    elif age == '51-55':
        return 5
    else:
        return 6
data['Age'] = data['Age'].apply(map_age)
```

Well, that's taken care of.

Lets tend to the needs of **City_Category**.

In [12]:

```
city_category = np.unique(data['City_Category'])
city_category
```

Out[12]:

```
array(['A', 'B', 'C'], dtype=object)
```

Let's Start mapping

In [13]:

```
def map_city_categories(city_category):
    if city_category == 'A':
        return 2
    elif city_category == 'B':
        return 1
    else:
        return 0
data['City_Category'] = data['City_Category'].apply(map_city_catego
ries)
```

Let's do the final mapping : **Stay_In_Current_City_Years**

In [14]:

```
city_stay = np.unique(data['Stay_In_Current_City_Years'])
city_stay
```

Out[14]:

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

In [15]:

```
def map_stay(stay):
        if stay == '4+':
            return 4
        else:
            return int(stay)
```

```
#           current_years = stay
#           current_years = current_years.astype(int)
#           return current_years
data['Stay_In_Current_City_Years'] = data['Stay_In_Current_City_Yea
rs'].apply(map_stay)
```

We can drop **User_ID** and **Product_ID** parameters. Let's get going

In [16]:

```
cols = ['User_ID','Product_ID']
data.drop(cols, inplace = True, axis =1)
```

Lets see how are final dataframe looks like!

In [17]:

```
data.head()
```

Out[17]:

	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Pr
0	0	0	10	2	2	0	3
1	0	0	10	2	2	0	1
2	0	0	10	2	2	0	1:
3	0	0	10	2	2	0	1:
4	1	6	16	0	4	0	8

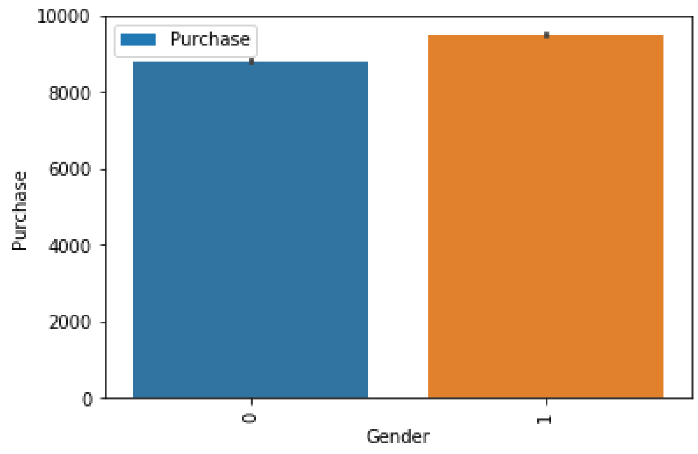
Beautiful!

EDA

In [18]:

```
data[['Gender','Purchase']].groupby('Gender').mean().plot.bar()
sns.barplot('Gender', 'Purchase', data = data)
plt.show()
```

```
/opt/conda/lib/python3.6/site-packages/scipy/stats/stats.py:1713: F
utureWarning: Using a non-tuple sequence for multidimensional index
ing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In
the future this will be interpreted as an array index, `arr[np.arra
y(seq)]`, which will result either in an error or a different resul
t.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumv
al
```

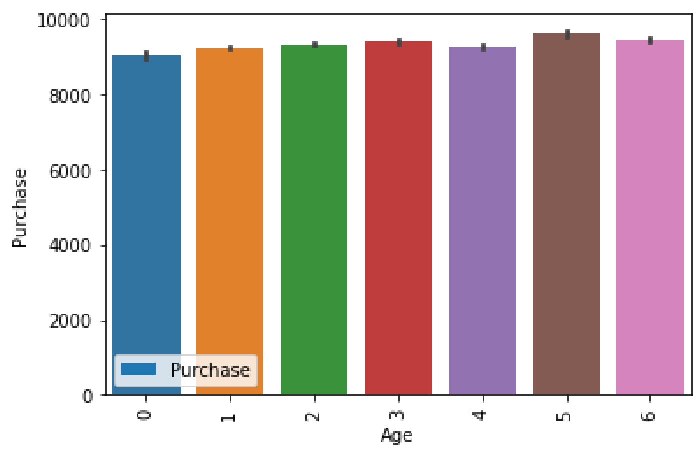


It looks like that men tend to spend more on Black Friday although women are not far behind.

Let's see how Age affects the Purchase. Of the top of my head I can say that people of higher age will tend to spen more as they would have more income. Let's see where this gets us.

In [19]:
```
data[['Age','Purchase']].groupby('Age').mean().plot.bar()
sns.barplot('Age', 'Purchase', data = data)
plt.show()
```

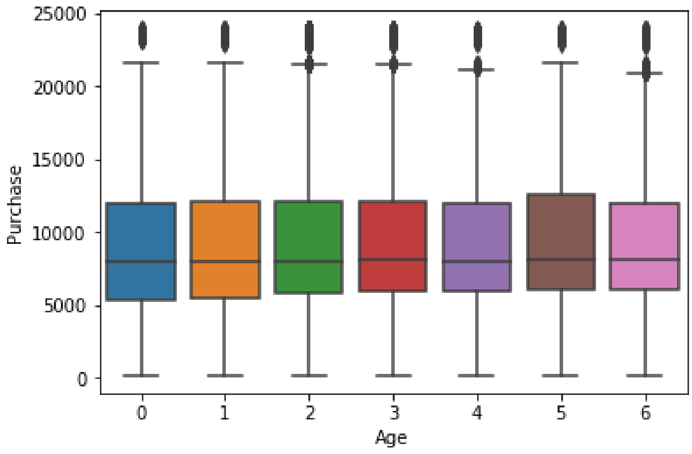
```
/opt/conda/lib/python3.6/site-packages/scipy/stats/stats.py:1713: F
utureWarning: Using a non-tuple sequence for multidimensional index
ing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In
the future this will be interpreted as an array index, `arr[np.arra
y(seq)]`, which will result either in an error or a different resul
t.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumv
al
```



I will also plot some boxplots to study the deviation in Age vs Purchase

In [20]:

```
sns.boxplot('Age','Purchase', data = data)
plt.show()
```



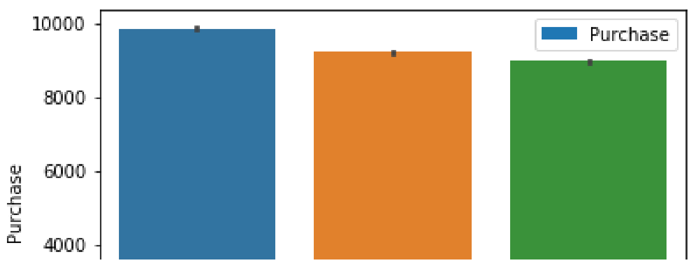
Not much of a deciation there. We can say that no matter what age group you belong to, you are gonna make full use of your purchasing power on a Black Friday. Maybe, because everything is so damn cheap (That's what I have heard! :P)

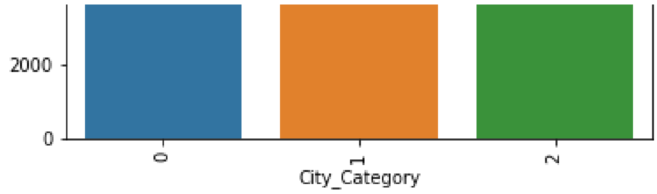
Lets see how city category affects the purchase.

In [21]:

```
data[['City_Category','Purchase']].groupby('City_Category').mean().
plot.bar()
sns.barplot('City_Category', 'Purchase', data = data)
plt.show()
```

```
/opt/conda/lib/python3.6/site-packages/scipy/stats/stats.py:1713: F
utureWarning: Using a non-tuple sequence for multidimensional index
ing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In
the future this will be interpreted as an array index, `arr[np.arra
y(seq)]`, which will result either in an error or a different resul
t.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumv
al
```





Okay so, the people belonging to category 0 tend to spend a little more. These may be the more developed cities that we are talking about here.

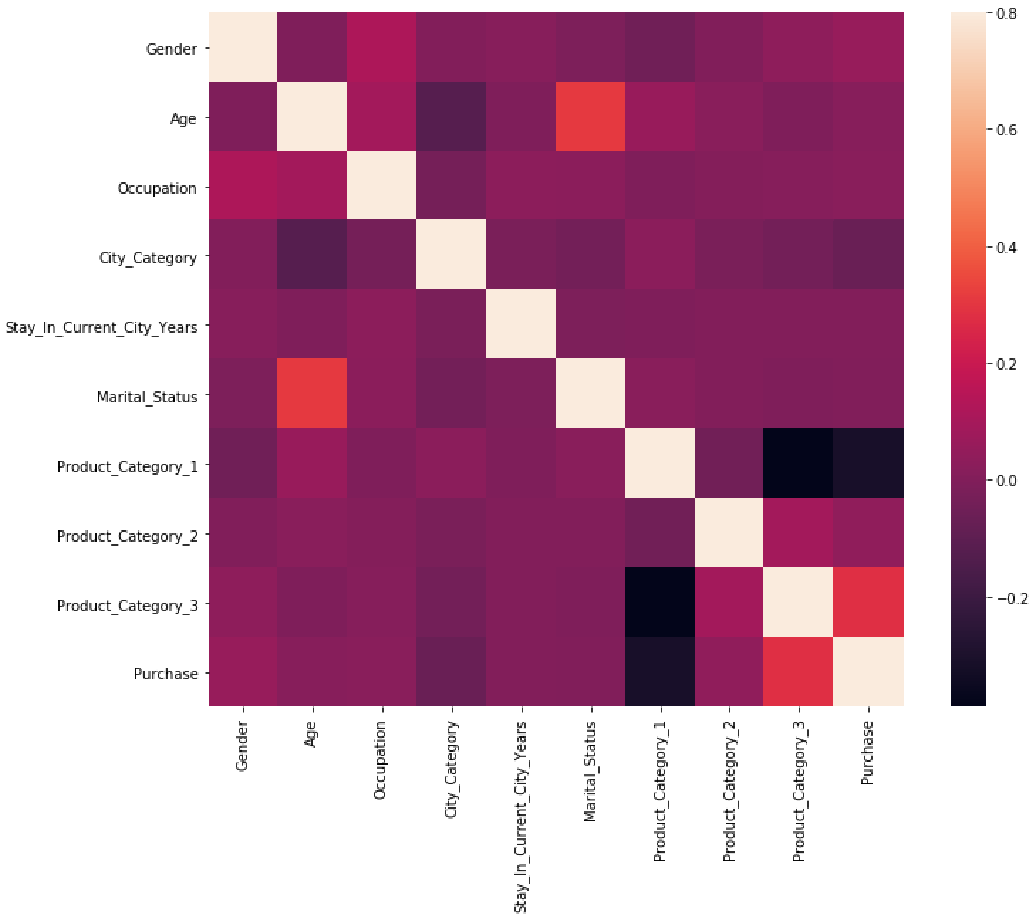
Let's now draw a heatmap to clearly see what are the correlations here.

In [22]:

```
corrmat = data.corr()
fig,ax = plt.subplots(figsize = (12,9))
sns.heatmap(corrmat, vmax=.8, square=True)
```

Out[22]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7efc1b8246d8>
```



It can be seen that nothing is highly correlated with the Purchase variable. Although a few conclusions can be drawn:

1. Product_Category_1 has a negative correlation with Purchase.

2. Maritial_Status and Age are strongly correlated. As Expected.
3. Product_Category_3 has a strong correlation with Purchase. Maybe the products in this category were cheap. Let's chrun out some number related to this.

In [23]:

```
mean_cat_1 = data['Product_Category_1'].mean()
mean_cat_2 = data['Product_Category_2'].mean()
mean_cat_3= data['Product_Category_3'].mean()
print(f"PC1: {mean_cat_1} \n PC2: {mean_cat_2} \n PC3 : {mean_cat_
```

Did you find this Kernel useful?
Show your appreciation with an upvote

4

Comments (6)

All Comments

Sort by Hotness

Click here to enter a comment...

ClamSalad • Posted on Latest Version • 14 days ago • Options • Reply

8

What is "occupation" representing? Is that related to annual income?

Aniketh Samson • Posted on Latest Version • 17 days ago • Options • Reply

7

What do the product categories represent? If the category doesn't represent the price, how can the last conclusion be drawn?

Bonolo • Posted on Latest Version • 20 days ago • Options • Reply

21

Where can i get the names of the product categories and types of occupations?

Chepkorir • Posted on Latest Version • a month ago • Options • Reply

0

Thanks ,this was very helpful in reminding myself what i just learnt in my DS class at the university last week

Tanmay Lata Kernel Author • Posted on Latest Version • a month ago • Options • Reply

0

Thanks a lot for your kind words! Glad I could be of help! :)

a month ago

This Comment was deleted.