
[Sinan Akbaba](#)

Black Friday Review

last run 14 days ago · IPython Notebook HTML · 350 views
using data from [Black Friday](#) ·  Public

5

voters

Notebook

Contents

1. [Introduction](#)
2. [Dataset](#)
3. [Import Modules and Reading the Data](#)
4. [Graphs and Reviews](#)

1. Introduction

First of all, This is my first kernel and of course i have many wrongs. Please excuse me.

I am taking the courses of ML and AI, and I am at the beginning. I search some datasets and i decide use Black Friday dataset. I will gradually make improvements on this data set. With this data set, I learn how to use kaggle and how to obtain meaningful data and how to use it. I regret that I found this site late. I wish I had already started processing data. But i think it's not too late :)

2. Dataset

More in <https://www.kaggle.com/mehdidag/black-friday>
(<https://www.kaggle.com/mehdidag/black-friday>)

3. Import Modules and Reading the Data

In [1]:

```
# This Python 3 environment comes with many helpful an  
alytics libraries installed  
# It is defined by the kaggle/python docker image: htt  
ps://github.com/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_csv)  
import matplotlib.pyplot as plt #
```

```
import seaborn as sns # visualization tool

# Input data files are available in the "../input/" di
rectory.
# For example, running this (by clicking run or pressi
ng Shift+Enter) will list the files in the input direc
tory

from subprocess import check_output
print(check_output(["ls", "../input"]).decode("utf8"
))

# Any results you write to the current directory are s
aved as output.
```

BlackFriday.csv

In [2]:

```
# read data in "BlackFriday.csv"
data = pd.read_csv("../input/BlackFriday.csv")
```

In [3]:

```
# get data information
data.info()
```

```
<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_Cateorv_2  370591 non-
```

```
dtype: object\nnull float64\nProduct_Category_3      164278 non-\nnull float64\nPurchase                537577 non-\nnull int64\ndtypes: float64(2), int64(5), object(5)\nmemory usage: 49.2+ MB
```

In [4]:

```
# show data columns\ndata.columns
```

Out[4]:

```
Index(['User_ID', 'Product_ID', 'Gender',\n      'Age', 'Occupation', 'City_Category',\n      'Stay_In_Current_City_Years', 'Mar-\nital_Status', 'Product_Category_1',\n      'Product_Category_2', 'Product_Cat-\negory_3', 'Purchase'],\n      dtype='object')
```

In [5]:

```
# show first 10 data\data.head(10)
```

Out[5]:

	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
5	1000003	P00193542	M	26-35	15	A
6	1000004	P00184942	M	46-50	7	B
7	1000004	P00346142	M	46-50	7	B
8	1000004	P0097242	M	46-50	7	B
9	1000005	P00274942	M	26-35	20	A

4. Graphs and Reviews

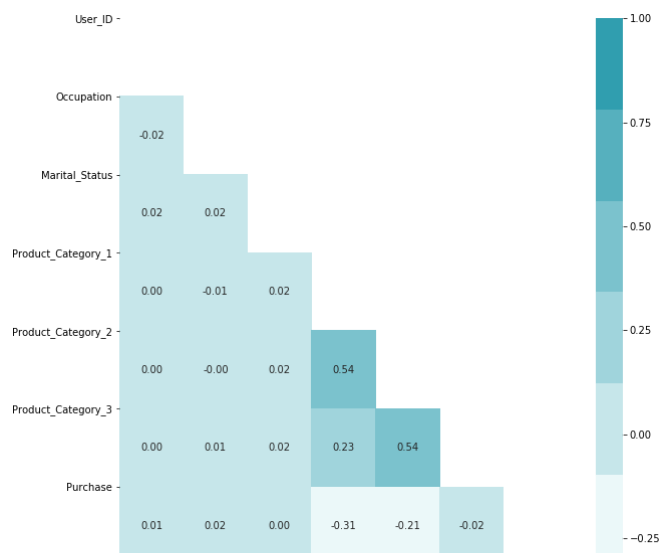
Correlation

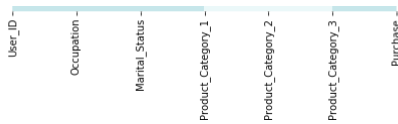
In [6]:

```
# correlation in consol scrren
#data.corr()

# correlation visualization square
#f,ax = plt.subplots(figsize=(10,5))
#sns.heatmap(data.corr(), annot=True, linewidths=.5, f
mt= '.1f',ax=ax)
#plt.show()

# correlation visualization triangle
corr = data.corr()
# Plot figsize
fig, ax = plt.subplots(figsize=(10, 10))
dropSelf = np.zeros_like(corr)
dropSelf[np.triu_indices_from(dropSelf)] = True
# Generate Color Map
colormap = sns.light_palette((210, 90, 60), input="hu
sl")
# Generate Heat Map, allow annotations and place float
s in map
sns.heatmap(corr, cmap=colormap, annot=True, fmt=".2
f", mask=dropSelf)
# Apply xticks
plt.xticks(range(len(corr.columns)), corr.columns);
# Apply yticks
plt.yticks(range(len(corr.columns)), corr.columns)
plt.show()
```

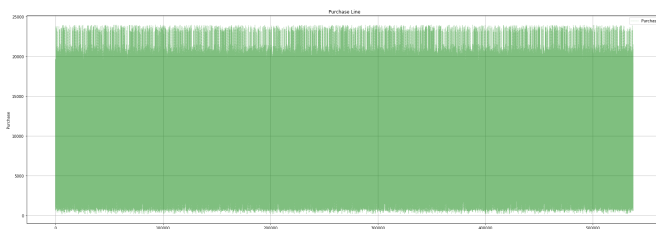




"Product_Category_1" and "Product_Category_2" seem correlation.
Same time "Product_Category_2" and "Product_Category_3" seem correlation.

In [7]:

```
# plot Purchase
# kind = kind, color = color, label = label, linewidth
# = width of line, alpha = opacity,
# grid = grid, linestyle = sytle of line, figsize = fi
# gure size
data.Purchase.plot(kind = 'line', color = 'g', label
= 'Purchase', linewidth=1, alpha = 0.5, grid = True,
linestyle = ':', figsize=(30, 10))
plt.legend(loc='upper right')      # legend = puts labe
l into plot
plt.xlabel('x axis')              # label = name of la
bel
plt.ylabel('Purchase')
plt.title('Purchase Line')        # title = title of p
lot
plt.show()
```



Hardly understanding this graph. Let's filter method.

Filtering

In [8]:

```
# filtering Pandas data frame
myFilterParam = data.Purchase < 1000
data[myFilterParam].head(10)

# or
#data[ data.Purchase < 1000].head(10)
```

Out[8]:

	User_ID	Product_ID	Gender	Age	Occupation
80	1000018	P0094142	F	18-25	3
122	1000023	P00112342	M	36-45	0
147	1000028	P00084442	F	26-35	1
175	1000033	P00219242	M	46-50	3
371	1000060	P00132042	M	51-55	1
536	1000097	P00089042	F	36-45	3
557	1000102	P00084642	M	36-45	19
739	1000139	P00227542	F	26-35	20
744	1000140	P00084642	F	36-45	1
774	1000146	P00112342	F	36-45	20

In [9]:

```
# filtering Pandas with Numpy logical and
#data[ np.logical_and( data.Purchase > 23900, data.Gender == 'F') ]

# multi filtering (i don't know how right it is?)
data[ np.logical_and( np.logical_and( data.Purchase > 23900, data.Gender == 'F'), data.Age == '18-25' ) ]
```

Out[9]:

	User_ID	Product_ID	Gender	Age	Occupation
69228	1004579	P00119342	F	18-25	4
88905	1001688	P00159542	F	18-25	4

Let's graph the data to a clear level

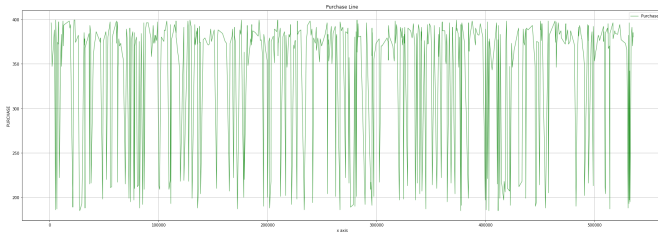
In [10]:

```
# Apply the filter to the data first, then draw the price from the filtered data
data[ data.Purchase < 400].Purchase.plot(figsize=(30, 10), kind = 'line', color = 'g', label = 'Purchase', linewidth=1, alpha = 0.8, grid = True, linestyle =
```

```

'-' )
plt.legend(loc='upper right')      # legend = puts label into plot
plt.xlabel('x axis')              # label = name of label
plt.ylabel('PURCHASE')
plt.title('Purchase Line')        # title = title of plot
plt.show()

```



How many different product category, product ID and occupation.

Unique Elements

In [11]:

```

# unique values
print('Unique Product Category: {0}'.format( len( data[
a["Product_Category_1"].unique() )))
print('Unique Product ID: {0}'.format( len( data["Product_ID"].unique() )))
print('Unique Occupation: {0}'.format( len( data["Occupation"].unique() )))

```

Unique Product Category: 18

Unique Product ID: 3623

Unique Occupation: 21

Let's we find maximum, minimum and average purchase

Maximum - Minimum Value

In [12]:

```

print("Maximum Purchase: {0} $".format( data.Purchase
.max()))
print("Minimum Purchase: {0} $".format( data.Purchase
.min()))

```



```
print("Average Purchase: {0:.2f} $".format( data.Purchase.mean()))
# {0:.2f} -> take two steps after zero
#print("Average Purchase: {0} $".format( data.Purchase.mean()))
```

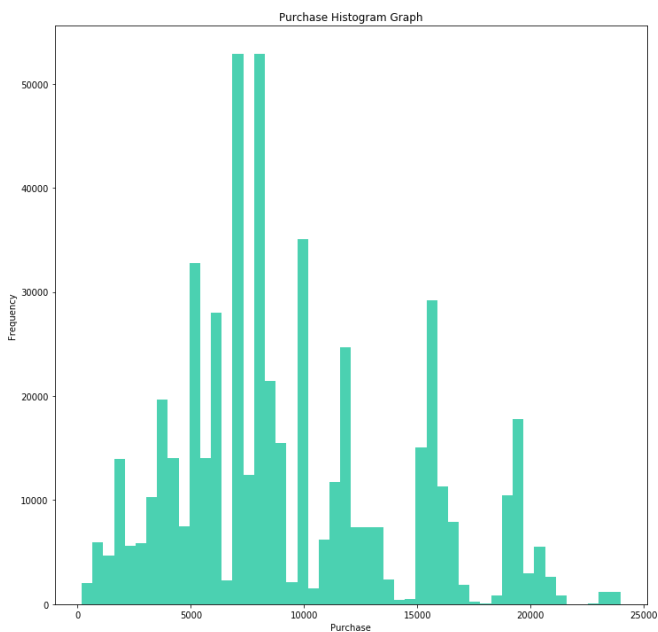
Maximum Purchase: 23961 \$
Minimum Purchase: 185 \$
Average Purchase: 9333.86 \$

I always thought Black Friday was a day of discount. I wouldn't have guessed that the average purchase would be so high. But still not understood much. So try histogram graph.

Histogram Graph

In [13]:

```
# plot Purchase
# kind = kind, color = color, bins = number of bar in figure, figsize = figure size
data.Purchase.plot(kind = 'hist', color = "#4bd1b1", bins = 50, figsize=(12, 12))
plt.title('Purchase Histogram Graph')
plt.xlabel('Purchase')
plt.show()
```



More understandable graph than line. We can see how much it is from every purchase. There are very few products over 20,000

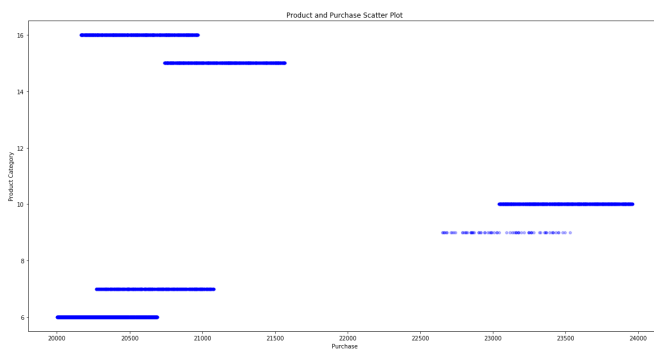
purchase.

Try scatter plot for Purchase > 20000 and Product Category. So that we can learn the relationship between product and purchase.

Scatter Graph

In [14]:

```
# plot Scatter
# price filtering before
data[data.Purchase > 20000].plot(kind='scatter', x='Purchase', y='Product_Category_1', alpha = 0.3, color = 'b', figsize=(20, 10))
#data[data.Purchase < 500].plot(kind='scatter', x='Purchase', y='Product_Category_1', alpha = 0.3, color = 'red', figsize=(20, 10))
plt.xlabel('Purchase')
plt.ylabel('Product Category')
plt.title('Product and Purchase Scatter Plot')
plt.show()
```



Products category 6, 7, 9, 10, 15 and 16 have purchase for over \$ 20,000. There are fewer than 9 items and the most purchase product is the number 10 product. I wondered about 9 and 10.

We can also look at low purchase products.

Dictionary

In [15]:

```
# dictionary practice on python
myDictionary = {'hello': 'merhaba',
                'pen': 'kalem',
                'window': 'pencere'}
print( myDictionary.keys())
print( myDictionary.values())
print( '----'*20)
```

```

# add new element
myDictionary['computer'] = 'bilgisayar'
print( myDictionary)
# change element
myDictionary['hello'] = 'selam'
print( myDictionary)
# delete element
del myDictionary['hello']
print( myDictionary)
print('hello' in myDictionary)
# get 1 element value
print(myDictionary['window'])
# clear dictionary
myDictionary.clear()

```

```

dict_keys(['hello', 'pen', 'window'])
dict_values(['merhaba', 'kalem', 'pencere'])
-----
-----
{'hello': 'merhaba', 'pen': 'kalem', 'window': 'pencere', 'computer': 'bilgisayar'}
{'hello': 'selam', 'pen': 'kalem', 'window': 'pencere', 'computer': 'bilgisayar'}
{'pen': 'kalem', 'window': 'pencere', 'computer': 'bilgisayar'}
False
pencere

```

Let's look at the relationship between cities and buyers with a histogram graph

City

In [16]:

```

# how many different cities
data["City_Category"].unique()

```

Out[16]:

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

In [17]:

```

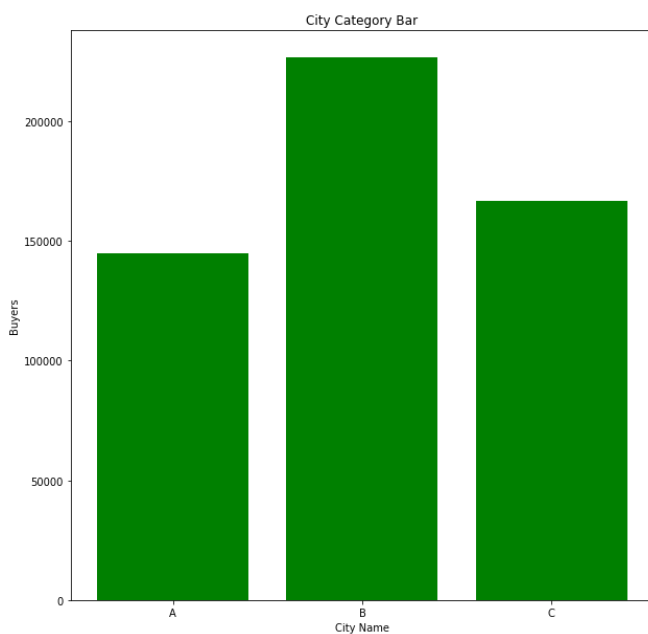
# find every city frequency
# P.S. -> I did this because the City_Category were string. I thought there are easier way!

```

ring. I'm sure there are easier ways!

```
cityDict = {'A':0, 'B':0, 'C':0}
for each in data['City_Category']:
    cityDict[each] += 1

# draw figure
fig = plt.figure(figsize=(10,10))
# add graph
ax = fig.add_subplot(111)
# draw graph
ax.bar( cityDict.keys(), cityDict.values(), color='g'
)
plt.title('City Category Bar')
plt.xlabel('City Name')
plt.ylabel('Buyers')
plt.show()
```



Rich Man

In [18]:

```
# find most purchase user
print('Number of different customers: {0}\n'.format(
    len( data['User_ID'].unique() )))

userDict = {}
# look at both at the same time
for purch, userId in zip(data['Purchase'], data['User
_ID']):
    # if already added userId in dictionary, add to pu
rchase
    if userDict.get(userId, -1) != -1:
        userDict[userId] += purch
```

```

# else add user and purchase
else:
    userDict[userId] = purch

# find maximum purchase and buyer
print( max(zip(userDict.values(), userDict.keys() )))

#data[ data['User_ID'] == max(zip(userDict.values(), u
serDict.keys()))[1] ]
winnerUser = data[ data['User_ID'] == max(zip(userDic
t.values(), userDict.keys()))[1] ]
winnerUser.head()

#winnerUser['Purchase'].sum()

# different purchase Product
#print( len( winnerUser['Product_ID'].unique() ))

```

Number of different customers: 5891

(10536783, 1004277)

Out[18]:

	User_ID	Product_ID	Gender	Age	Occupation
27930	1004277	P00034742	M	36-45	16
27931	1004277	P00028542	M	36-45	16
27932	1004277	P00116842	M	36-45	16
27933	1004277	P00063342	M	36-45	16
27934	1004277	P00359042	M	36-45	16

He is amazing! He purchase is 10,536,783 \$. There are 978 different product. I would love to meet him :)

Now, let's examine the relationship between age and purchase

Age

In [19]:

```

# use dictionary because age features is string
ageDict = {}
for purch, userAge in zip( data['Purchase'], data['Ag

```

```
e']):
    if ageDict.get(userAge, -1) != -1:
        ageDict[userAge] += purch
    else:
        ageDict[userAge] = purch

# total purchase by age but no sort
print( ageDict)
print('---'*50)
# we sorted dictionary for more beatiful graph
from collections import OrderedDict
# key -> x[0]:sort dict keys, x[1]:sort dict values
age_sorted = OrderedDict( sorted( ageDict.items(), ke
y=lambda x: x[0]))
print( age_sorted)
#print('---'*50)
#age_sorted1 = OrderedDict( sorted( ageDict.items(), k
ey=lambda x: x[1]))
#print( age_sorted1)
```

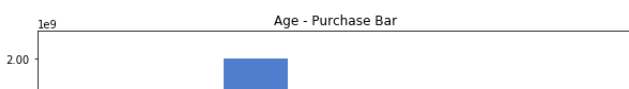
```
{'0-17': 132659006, '55+': 197614842, '26
-35': 1999749106, '46-50': 413418223, '51
-55': 361908356, '36-45': 1010649565, '18
-25': 901669280}
```

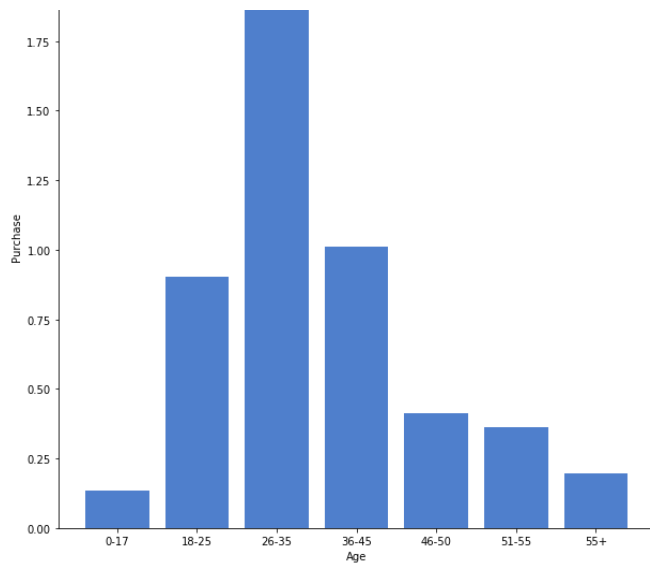
```
-----
-----
-----
-----
```

```
OrderedDict([('0-17', 132659006), ('18-2
5', 901669280), ('26-35', 1999749106),
('36-45', 1010649565), ('46-50', 41341822
3), ('51-55', 361908356), ('55+', 1976148
42)])
```

In [20]:

```
# draw figure
fig = plt.figure(figsize=(10,10))
# add graph
ax = fig.add_subplot(111)
# draw graph
ax.bar( age_sorted.keys(), age_sorted.values(), color
='#4f7fcc')
plt.title('Age - Purchase Bar')
plt.xlabel('Age')
plt.ylabel('Purchase')
plt.show()
```





26-35 age range has the most purchases. I think most of the black Friday products cater for the 23-35 age range. It is obvious that the target group of the Black Friday is the 27-35 age group. 0-17 age range has the least purchases. Products sold in the 0-17 age range (ie children's products) are very few.

Let's examine the relationship between gender and purchase

Gender

In [21]:

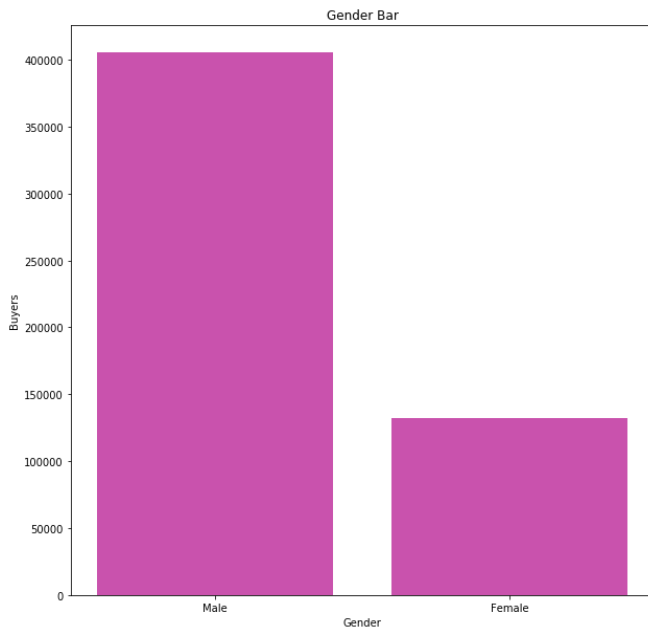
```
# get male and female user
male_user = data[ data[ 'Gender' ] == 'M' ].count()[0]
female_user = data[ data[ 'Gender' ] == 'F' ].count()[0]
]
print('Male user: {0}'.format(male_user))
print('Female user: {0}'.format(female_user))
```

```
Male user: 405380
Female user: 132197
```

In [22]:

```
# draw figure
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(111)
ax.bar( [ 'Male', 'Female' ], [male_user, female_user],
        color='#c952ad')
plt.title('Gender Bar')
plt.xlabel('Gender')
```

```
plt.ylabel('Buyers')
plt.show()
```



It is clear that men do more shopping than women. Let's look purchase by gender.

In [23]:

```
# total purchase by gender
male_purch = data[ data['Gender']=='M']['Purchase'].sum()
female_purch = data[ data['Gender']=='F']['Purchase'].sum()
print('Male user: {0}'.format(male_purch))
print('Female user: {0}'.format(female_purch))
```

```
Male user: 3853044357
Female user: 1164624021
```

In [24]:

```
# draw figure
fig = plt.figure(figsize=(15,10))

# graph total purchase by gender
ax1 = fig.add_subplot(121)
ax1.bar( ['Male', 'Female'], [male_purch, female_purch], color='#6152c9')
plt.title('Total Purchase by Gender')
plt.ylabel('Purchase')

# graph total purchase by gender
```



```
ax2 = fig.add_subplot(122)
```

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

5



Comments (0)

Sort by Select...



Click here to enter a comment...

© 2018 Kaggle Inc

[Our Team](#) [Terms](#) [Privacy](#) [Contact/Support](#)

