navyas-insurance-case-study

May 14, 2023

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
pd.options.display.float_format = "{:.2f}".format
import datetime as dt
import seaborn as sns


sns.set_context('talk')
import scipy.stats as stats


# set seabor graphs to a better style
sns.set(style="ticks")

# for better visualization
plt.style.use('ggplot')


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

0.1 1. Import claims_data.csv and cust_data.csv which is provided to you and combine the two datasets appropriately to create a 360-degree view of the data. Use the same for the subsequent questions.

[2]:
```
cust=pd.read_csv('cust_demographics.csv')
cust
```

[2]:

	CUST_ID	gender	DateOfBirth	State	Contact	Segment
0	21868593	Female	12-Jan-79	VT	789-916-8172	Platinum
1	75740424	Female	13-Jan-70	ME	265-543-1264	Silver
2	30308357	Female	11-Mar-84	TN	798-631-4758	Silver
3	47830476	Female	01-May-86	MA	413-187-7945	Silver
4	19269962	Male	13-May-77	NV	956-871-8691	Gold
…	…	…	…	…	…	…

```
1080  79539873  Female   15-Mar-81    ND  459-425-4319  Platinum
1081  42364152  Female   07-Jul-96    ID  529-462-1635    Silver
1082  19888166    Male   11-Apr-90    WI  712-651-9613      Gold
1083  11256802  Female   22-Oct-64    LA  469-345-5617    Silver
1084  61575264    Male   12-Jul-95    WY  182-385-1392      Gold

[1085 rows x 6 columns]
```

[3]:
```
claims=pd.read_csv('claims.csv')
claims
```

[3]:
```
      claim_id  customer_id      incident_cause  claim_date claim_area  \
0     54004764     21868593        Driver error  11/27/2017       Auto
1     33985796     75740424               Crime  10/03/2018       Home
2     53522022     30308357  Other driver error  02/02/2018       Auto
3     13015401     47830476      Natural causes  06/17/2018       Auto
4     22890252     19269962               Crime  01/13/2018       Auto
…        …           …                   …          …        …
1095  97727122     35951012  Other driver error  06/11/2017       Auto
1096  10247193     14818669      Natural causes  03/14/2018       Auto
1097  79807493     85322831  Other driver error  02/09/2018       Auto
1098  69299345     73449366        Other causes  03/21/2018       Auto
1099  58809728     43020876  Other driver error  06/04/2017       Auto

     police_report           claim_type claim_amount  total_policy_claims  \
0               No        Material only        $2980                 1.00
1          Unknown        Material only        $2980                 3.00
2               No        Material only      $3369.5                 1.00
3               No        Material only        $1680                 1.00
4               No        Material only        $2680                 1.00
…              …                   …           …                    …
1095            No        Material only        $3059                 4.00
1096            No        Material only        $1520                 2.00
1097            No  Material and injury       $23575                 2.00
1098       Unknown  Material and injury       $25120                 1.00
1099           Yes  Material and injury       $36685                 3.00

     fraudulent
0            No
1            No
2           Yes
3            No
4            No
…           …
1095        Yes
1096         No
1097        Yes
```

```
1098         No
1099        Yes

[1100 rows x 10 columns]
```

[4]:
```
cust_claims = pd.merge(right = claims, left = cust, right_on = "customer_id",␣
 ↪left_on = "CUST_ID",how = "outer")
cust_claims.drop(columns = ["customer_id"], inplace = True)
cust_claims.head()
```

[4]:

	CUST_ID	gender	DateOfBirth	State	Contact	Segment	claim_id	incident_cause	claim_date	claim_area	police_report	claim_type	claim_amount	total_policy_claims	fraudulent
0	21868593.00	Female	12-Jan-79	VT	789-916-8172	Platinum	54004764.00	Driver error	11/27/2017	Auto	No	Material only	$2980	1.00	No
1	75740424.00	Female	13-Jan-70	ME	265-543-1264	Silver	33985796.00	Crime	10/03/2018	Home	Unknown	Material only	$2980	3.00	No
2	30308357.00	Female	11-Mar-84	TN	798-631-4758	Silver	53522022.00	Other driver error	02/02/2018	Auto	No	Material only	$3369.5	1.00	Yes
3	30308357.00	Female	11-Mar-84	TN	798-631-4758	Silver	63017412.00	Driver error	04/04/2018	Auto	No	Material only	$1950	6.00	No
4	47830476.00	Female	01-May-86	MA	413-187-7945	Silver	13015401.00	Natural causes	06/17/2018	Auto	No	Material only	$1680	1.00	No

[5]:
```
cust_claims.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1107 entries, 0 to 1106
Data columns (total 15 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   CUST_ID              1092 non-null   float64
 1   gender               1092 non-null   object
 2   DateOfBirth          1092 non-null   object
 3   State                1092 non-null   object
 4   Contact              1092 non-null   object
 5   Segment              1092 non-null   object
 6   claim_id             1100 non-null   float64
 7   incident_cause       1100 non-null   object
 8   claim_date           1100 non-null   object
```

```
 9   claim_area           1100 non-null   object
 10  police_report        1100 non-null   object
 11  claim_type           1100 non-null   object
 12  claim_amount         1035 non-null   object
 13  total_policy_claims  1090 non-null   float64
 14  fraudulent           1100 non-null   object
dtypes: float64(3), object(12)
memory usage: 138.4+ KB
```

0.2 2. Perform a data audit for the datatypes and find out if there are any mismatch within the current datatypes of the columns and their business significance.

[6]:
```
cust_claims["DateOfBirth"] = pd.to_datetime(cust_claims.DateOfBirth, format =␣
 ↪"%d-%b-%y")
cust_claims.loc[(cust_claims.DateOfBirth.dt.year >␣
 ↪2020),"DateOfBirth"]=cust_claims[cust_claims.DateOfBirth.dt.year >␣
 ↪2020]["DateOfBirth"].apply(lambda x: x - pd.DateOffset(years=100))
cust_claims["claim_date"] = pd.to_datetime(cust_claims.claim_date, format = "%m/
 ↪%d/%Y")
cust_claims["Contact"] = pd.to_numeric(cust_claims.Contact.str.
 ↪replace("-",""),downcast='float')
cust_claims["claim_amount"] = pd.to_numeric(cust_claims.claim_amount.str.
 ↪replace("$",""),downcast='float')
cust_claims.head()
```

[6]:

	CUST_ID	gender	DateOfBirth	State	Contact	Segment	claim_id	incident_cause	claim_date	claim_area	police_report	claim_type	claim_amount	total_policy_claims	fraudulent
0	21868593.00	Female	1979-01-12	VT	7899168256.00	Platinum	54004764.00	Driver error	2017-11-27	Auto	No	Material only	2980.00	1.00	No
1	75740424.00	Female	1970-01-13	ME	2655431168.00	Silver	33985796.00	Crime	2018-10-03	Home	Unknown	Material only	2980.00	3.00	No
2	30308357.00	Female	1984-03-11	TN	7986314752.00	Silver	53522022.00	Other driver error	2018-02-02	Auto	No	Material only	3369.50	1.00	Yes
3	30308357.00	Female	1984-03-11	TN	7986314752.00	Silver	63017412.00	Driver error	2018-04-04	Auto	No	Material only	1950.00	6.00	No
4	47830476.00	Female	1986-05-01	MA	4131877888.00	Silver	13015401.00	Natural causes	2018-06-17	Auto	No	Material only	1680.00	1.00	No

[7]:
```
cust_claims.dtypes
```

[7]:
```
CUST_ID                       float64
gender                         object
DateOfBirth            datetime64[ns]
State                          object
Contact                       float32
Segment                        object
claim_id                      float64
incident_cause                 object
claim_date             datetime64[ns]
claim_area                     object
police_report                  object
claim_type                     object
claim_amount                  float32
total_policy_claims           float64
fraudulent                     object
dtype: object
```

0.3 4 Of all the injury claims, some of them have gone unreported with the police. Create an alert flag (1,0) for all such claims¶

[8]:
```
cust_claims['unreported_claims']=  np.where(cust_claims.police_report==␣
 ↪'Unknown',1,0)
cust_claims['unreported_claims'].head(10)
```

[8]:
```
0    0
1    1
2    0
3    0
4    0
5    1
6    0
7    0
8    0
9    1
Name: unreported_claims, dtype: int32
```

0.4 5. One customer can claim for insurance more than once and in each claim,multiple categories of claims can be involved. However, customer ID should remain unique. Retain the most recent observation and delete any duplicated records in

the data based on the customer ID column.

[9]:
```
cust_claims = cust_claims.groupby('CUST_ID').first().reset_index(drop = True)
cust_claims.head()
```

[9]:
```
   gender DateOfBirth State       Contact   Segment    claim_id  \
0  Female  1978-05-23    DE 9628790784.00  Platinum 69348631.00
1    Male  1972-12-20    TX 1738926336.00    Silver 40953049.00
2    Male  1988-07-28    FL 3645981440.00    Silver 45780237.00
3    Male  1971-08-19    CA 1873488384.00      Gold 89833962.00
4  Female  1980-08-10    NC 7988625408.00      Gold 35782742.00

       incident_cause claim_date claim_area police_report  \
0        Driver error 2018-04-10       Auto            No
1        Other causes 2018-04-04       Auto            No
2      Natural causes 2017-10-17       Auto       Unknown
3        Other causes 2018-03-21       Auto           Yes
4  Other driver error 2018-07-27       Auto            No

            claim_type  claim_amount  total_policy_claims fraudulent
0          Injury only           NaN                 1.00        Yes
1  Material and injury      39192.00                 1.00        Yes
2        Material only       1621.50                 2.00        Yes
3  Material and injury      37040.00                 1.00         No
4          Injury only      35250.00                 3.00         No
```

0.5 6. Check for missing values and impute the missing values with an appropriate value. (mean for continuous and mode for categorical)

[10]:
```
cust_claims.isna().sum()
```

[10]:
```
gender                 0
DateOfBirth            0
State                  0
Contact                0
Segment                0
claim_id               7
incident_cause         7
claim_date             7
claim_area             7
police_report          7
claim_type             7
claim_amount          72
total_policy_claims   17
fraudulent             7
dtype: int64
```

[11]:
```
catagorical_col =␣
 ↪["gender","State","Segment","incident_cause","claim_area","claim_type","fraudulent"]
continious_col = ["claim_amount"]
```

[12]:
```
for col in catagorical_col:
    cust_claims[col] = cust_claims[col].fillna(cust_claims[col].mode()[0])
cust_claims[continious_col] = cust_claims[continious_col].
 ↪fillna(cust_claims[continious_col].mean())
cust_claims.head()
```

[12]:

	gender	DateOfBirth	State	Contact	Segment	claim_id	\
0	Female	1978-05-23	DE	9628790784.00	Platinum	69348631.00	
1	Male	1972-12-20	TX	1738926336.00	Silver	40953049.00	
2	Male	1988-07-28	FL	3645981440.00	Silver	45780237.00	
3	Male	1971-08-19	CA	1873488384.00	Gold	89833962.00	
4	Female	1980-08-10	NC	7988625408.00	Gold	35782742.00	

	incident_cause	claim_date	claim_area	police_report	\
0	Driver error	2018-04-10	Auto	No	
1	Other causes	2018-04-04	Auto	No	
2	Natural causes	2017-10-17	Auto	Unknown	
3	Other causes	2018-03-21	Auto	Yes	
4	Other driver error	2018-07-27	Auto	No	

	claim_type	claim_amount	total_policy_claims	fraudulent
0	Injury only	12470.50	1.00	Yes
1	Material and injury	39192.00	1.00	Yes
2	Material only	1621.50	2.00	Yes
3	Material and injury	37040.00	1.00	No
4	Injury only	35250.00	3.00	No

[13]:
```
cust_claims.isna().sum()
```

[13]:
```
gender                  0
DateOfBirth             0
State                   0
Contact                 0
Segment                 0
claim_id                7
incident_cause          0
claim_date              7
claim_area              0
police_report           7
claim_type              0
claim_amount            0
total_policy_claims    17
fraudulent              0
dtype: int64
```

0.6 7. Calculate the age of customers in years. Based on the age, categorize thecustomers according to the below criteria

Children < 18

Youth 18-30

Adult 30-60

Senior > 60

[10]:
```
cust_claims['Age']=(dt.datetime.now().year-pd.
 ↪DatetimeIndex(cust_claims['DateOfBirth']).year)
cust_claims
```

[10]:

	CUST_ID	gender	DateOfBirth	State	Contact	Segment	\
0	21868593.00	Female	1979-01-12	VT	7899168256.00	Platinum	
1	75740424.00	Female	1970-01-13	ME	2655431168.00	Silver	
2	30308357.00	Female	1984-03-11	TN	7986314752.00	Silver	
3	30308357.00	Female	1984-03-11	TN	7986314752.00	Silver	
4	47830476.00	Female	1986-05-01	MA	4131877888.00	Silver	
…	…	…	…	…	…	…	
1102	NaN	NaN	NaT	NaN	NaN	NaN	
1103	NaN	NaN	NaT	NaN	NaN	NaN	
1104	NaN	NaN	NaT	NaN	NaN	NaN	
1105	NaN	NaN	NaT	NaN	NaN	NaN	
1106	NaN	NaN	NaT	NaN	NaN	NaN	

	claim_id	incident_cause	claim_date	claim_area	police_report	\
0	54004764.00	Driver error	2017-11-27	Auto	No	
1	33985796.00	Crime	2018-10-03	Home	Unknown	
2	53522022.00	Other driver error	2018-02-02	Auto	No	
3	63017412.00	Driver error	2018-04-04	Auto	No	
4	13015401.00	Natural causes	2018-06-17	Auto	No	
…	…	…	…	…	…	
1102	97727122.00	Other driver error	2017-06-11	Auto	No	
1103	10247193.00	Natural causes	2018-03-14	Auto	No	
1104	79807493.00	Other driver error	2018-02-09	Auto	No	
1105	69299345.00	Other causes	2018-03-21	Auto	Unknown	
1106	58809728.00	Other driver error	2017-06-04	Auto	Yes	

	claim_type	claim_amount	total_policy_claims	fraudulent	\
0	Material only	2980.00	1.00	No	
1	Material only	2980.00	3.00	No	
2	Material only	3369.50	1.00	Yes	
3	Material only	1950.00	6.00	No	
4	Material only	1680.00	1.00	No	
…	…	…	…	…	
1102	Material only	3059.00	4.00	Yes	

```
1103        Material only       1520.00                 2.00         No
1104  Material and injury      23575.00                 2.00        Yes
1105  Material and injury      25120.00                 1.00         No
1106  Material and injury      36685.00                 3.00        Yes

      unreported_claims   Age
0                     0 44.00
1                     1 53.00
2                     0 39.00
3                     0 39.00
4                     0 37.00
…                   …   …
1102                  0   NaN
1103                  0   NaN
1104                  0   NaN
1105                  1   NaN
1106                  0   NaN

[1107 rows x 17 columns]
```

[13]:
```
curr_year = pd.to_datetime('today').year
dob_year = pd.DatetimeIndex(cust_claims['DateOfBirth']).year          #extract␣
 ↪year from DateOfBirth
x = dob_year-100                                                    # for the years␣
 ↪which belongs to 60's
v = curr_year - x
y = curr_year - dob_year
cust_claims['age'] = (np.where(dob_year > curr_year,v,y))
#Categorising
cust_claims.loc[(cust_claims.age < 18),'AgeGroup'] = 'Children'
cust_claims.loc[(cust_claims.age >=18) & (cust_claims.age <30),'AgeGroup'] =␣
 ↪'Youth'
cust_claims.loc[(cust_claims.age >=30) & (cust_claims.age <60),'AgeGroup'] =␣
 ↪'Adult'
cust_claims.loc[(cust_claims.age >=60),'AgeGroup'] = 'Senior'
```

[14]:
```
cust_claims.groupby(["AgeGroup"])["age"].count()
```

[14]:
```
AgeGroup
Adult     813
Senior    103
Youth     176
Name: age, dtype: int64
```

0.7 8.What is the average amount claimed by the customers from various segments?

[40]:
```
cust_claims.groupby(by = "Segment")[["claim_amount"]].mean()
```

[40]:
```
           claim_amount
Segment
Gold          12755.71
Platinum      12370.14
Silver        12271.16
```

0.8 9 What is the total claim amount based on incident cause for all the claims that have been done at least 20 days prior to 1st of October, 2018¶

[15]:
```
cust_claims.loc[cust_claims.claim_date < "2018-09-10",:].
 ↪groupby("incident_cause")["claim_amount"].sum().add_prefix("total_")
```

[15]:
```
incident_cause
total_Crime                  667262.00
total_Driver error          3199350.00
total_Natural causes        1204461.50
total_Other causes          3543469.00
total_Other driver error    3222297.50
Name: claim_amount, dtype: float32
```

0.9 10. How many adults from TX, DE and AK claimed insurance for driver related issues and causes?

[41]:
```
cust_claims.loc[(cust_claims.incident_cause.str.lower().str.contains("driver")
    & ((cust_claims.State == "TX") | (cust_claims.State == "DE") | (cust_claims.
 ↪State == "AK"))),:].groupby(by = "State")["State"].count()
```

[41]:
```
State
AK    10
DE    15
TX    10
Name: State, dtype: int64
```

0.10 11. Draw a pie chart between the aggregated value of claim amount based on gender and segment. Represent the claim amount as a percentage on the pie chart.

[17]:
```
gender_segment_DF = cust_claims.groupby(by =␣
 ↪["gender","Segment"])["claim_amount"].sum().reset_index()
gender_segment_DF
```

[17]:

	gender	Segment	claim_amount
0	Female	Gold	1997529.00
1	Female	Platinum	2282210.00
2	Female	Silver	1739832.00
3	Male	Gold	2539294.00
4	Male	Platinum	1998561.50
5	Male	Silver	2159608.50

[18]:
```
gender_segment_DF.pivot(index="Segment" , columns="gender",␣
↪values="claim_amount")
gender_segment_DF
```

[18]:

	gender	Segment	claim_amount
0	Female	Gold	1997529.00
1	Female	Platinum	2282210.00
2	Female	Silver	1739832.00
3	Male	Gold	2539294.00
4	Male	Platinum	1998561.50
5	Male	Silver	2159608.50

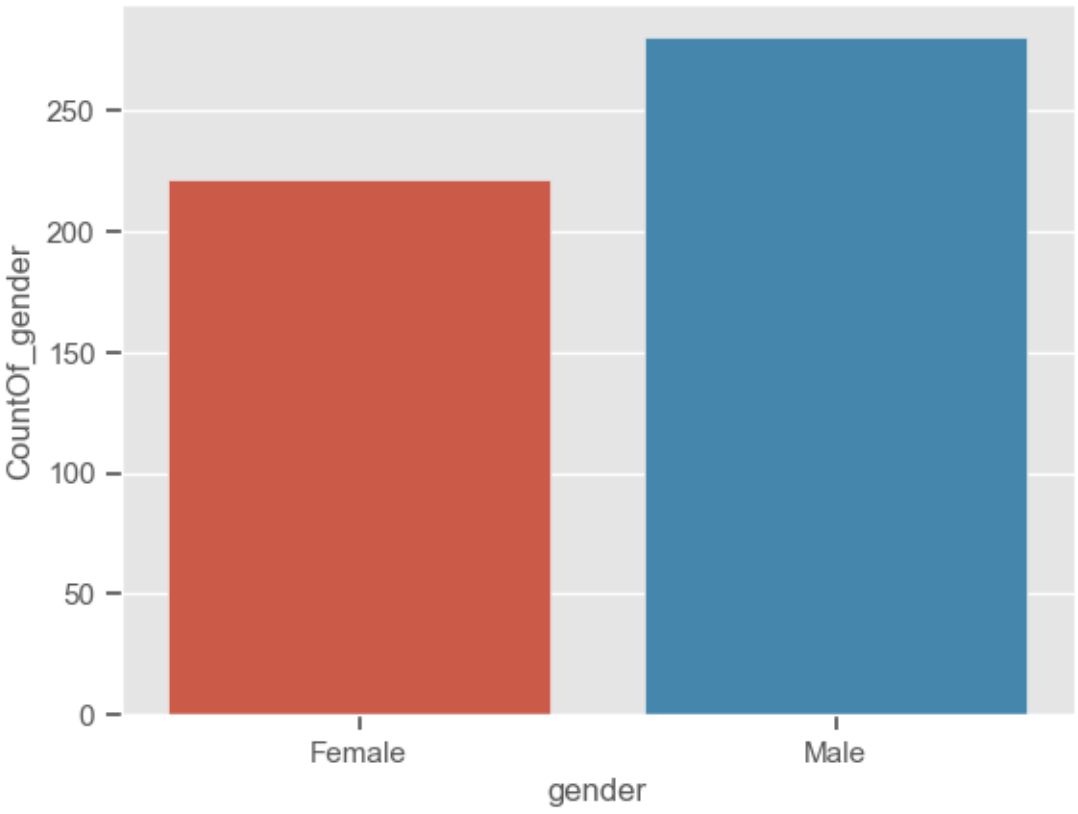
0.11 12 Among males and females, which gender had claimed the most for any type of driver related issues? E.g. This metric can be compared using a bar chart

[44]:
```
gender_count_DF = cust_claims.loc[(cust_claims.incident_cause.str.lower().str.
↪contains("driver"))].groupby(by = "gender")[["gender"]].count().
↪add_prefix("CountOf_").reset_index()
gender_count_DF.isna()
```

[44]:

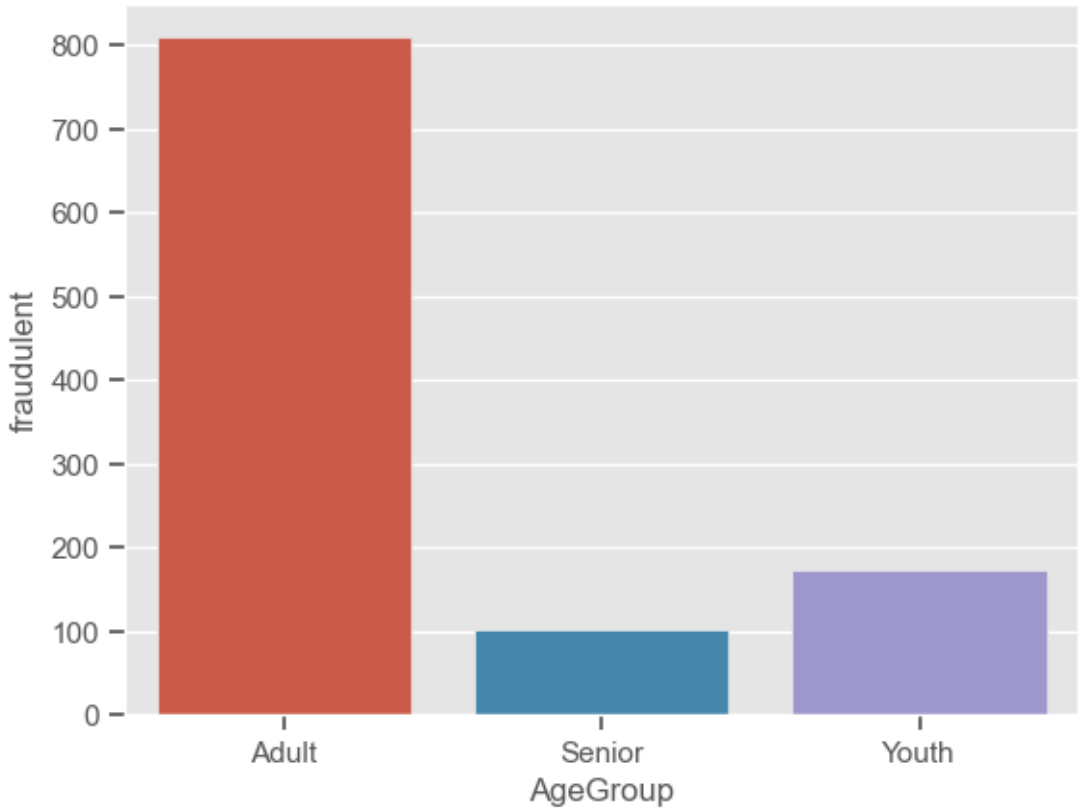
	gender	CountOf_gender
0	False	False
1	False	False

[45]:
```
sns.barplot(x = "gender", y = "CountOf_gender", data = gender_count_DF)
plt.show()
```



0.12 13 Which age group had the maximum fraudulent policy claims? Visualize it on a bar chart

[21]:
```
Age_Group_Fraud = cust_claims.groupby(by = "AgeGroup")[["fraudulent"]].count().
↪reset_index()
sns.barplot(x = "AgeGroup", y = "fraudulent", data = Age_Group_Fraud )
plt.show()
```



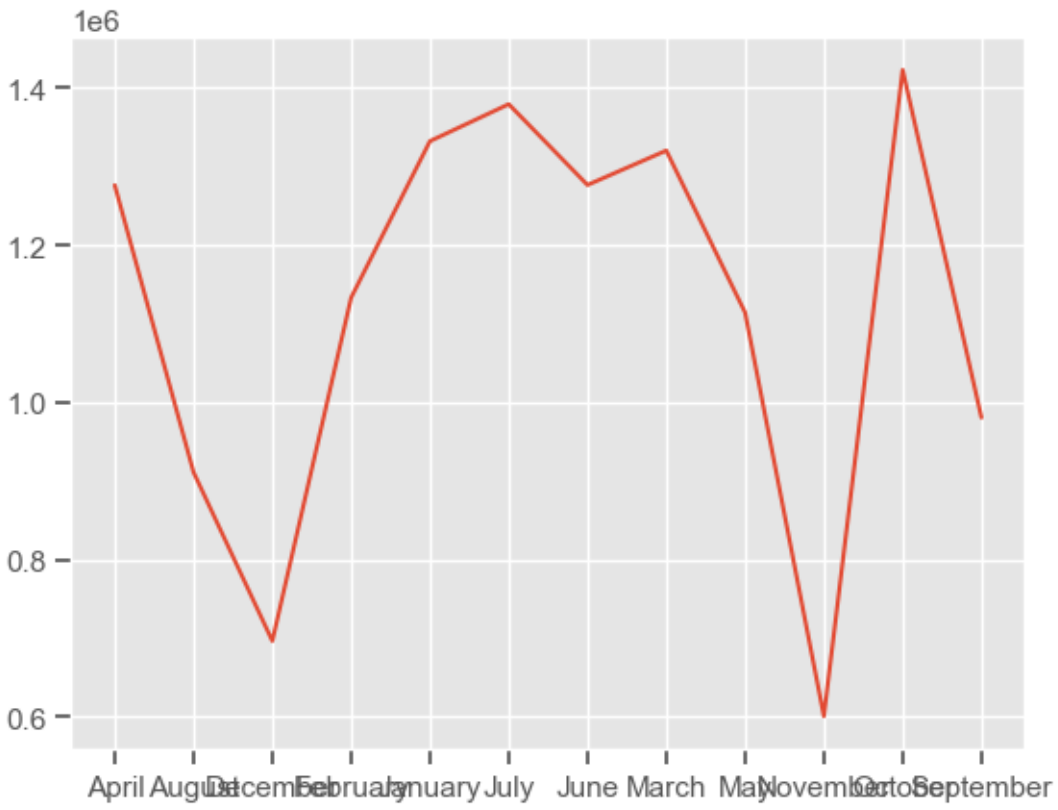
0.13 14 Visualize the monthly trend of the total amount that has been claimed by the customers.Ensure that on the "month" axis, the month is in a chronological order not alphabetical order

[46]:
```
monthly_trend = cust_claims.groupby(["claim_date"])[["claim_amount"]].sum().
↪reset_index()
```

[47]:
```
monthly_trend['Monthly'] = monthly_trend['claim_date'].apply(lambda x:pd.
↪Timestamp.strftime(x,format="%B"))
monthly_trend['Yearly'] = monthly_trend['claim_date'].apply(lambda x:pd.
↪Timestamp.strftime(x,format="%Y"))
```

[48]:
```
monthly_trend_data = monthly_trend.groupby(["Monthly"])[["claim_amount"]].sum().
↪reset_index()
```

[49]:
```
plt.plot(monthly_trend_data['Monthly'], monthly_trend_data['claim_amount'],␣
↪label = 'Trend Line')
plt.show()
```



0.14 16. Is there any similarity in the amount claimed by males and females?

[50]:
```
cust_claims.gender.value_counts()
```

[50]:
```
Male      553
Female    532
Name: gender, dtype: int64
```

[51]:
```
cust_claims_gender =  cust_claims.groupby(["gender",␣
 ↪"claim_date"])[["claim_amount"]].sum().reset_index()
cust_claims_gender
```

[51]:

	gender	claim_date	claim_amount
0	Female	2017-01-01	119620.00
1	Female	2017-01-07	4576.50
2	Female	2017-01-14	85203.00
3	Female	2017-01-21	23379.50
4	Female	2017-01-27	6507.50
..	…	…	…
194	Male	2018-10-03	60291.00
195	Male	2018-10-09	145030.00

```
196    Male 2018-10-16      21610.00
197    Male 2018-10-23      83410.50
198    Male 2018-10-30      61135.00

[199 rows x 3 columns]
```

[52]:
```
cust_claims_gender['Monthly'] = cust_claims_gender['claim_date'].apply(lambda x:
 ↪pd.Timestamp.strftime(x,format="%B"))
cust_claims_gender['Yearly'] = cust_claims_gender['claim_date'].apply(lambda x:
 ↪pd.Timestamp.strftime(x,format="%Y"))
```

[53]:
```
cust_claims.head()
```

[53]:
```
   gender DateOfBirth State       Contact   Segment    claim_id  \
0  Female  1978-05-23    DE 9628790784.00  Platinum 69348631.00
1    Male  1972-12-20    TX 1738926336.00    Silver 40953049.00
2    Male  1988-07-28    FL 3645981440.00    Silver 45780237.00
3    Male  1971-08-19    CA 1873488384.00      Gold 89833962.00
4  Female  1980-08-10    NC 7988625408.00      Gold 35782742.00

       incident_cause claim_date claim_area police_report  \
0        Driver error 2018-04-10       Auto            No
1        Other causes 2018-04-04       Auto            No
2      Natural causes 2017-10-17       Auto       Unknown
3        Other causes 2018-03-21       Auto           Yes
4  Other driver error 2018-07-27       Auto            No

            claim_type  claim_amount  total_policy_claims fraudulent
0          Injury only      12470.50                 1.00        Yes
1  Material and injury      39192.00                 1.00        Yes
2        Material only       1621.50                 2.00        Yes
3  Material and injury      37040.00                 1.00         No
4          Injury only      35250.00                 3.00         No
```

[54]:
```
Claim_amt = 'claim_amount'

male_spend = cust_claims_gender.loc[ cust_claims_gender.gender == "Male",␣
 ↪Claim_amt ]
female_spend = cust_claims_gender.loc[ cust_claims_gender.gender == "Female",␣
 ↪Claim_amt ]

print( 'mean of male spend: ', male_spend.mean(), '| mean of female spend: ',␣
 ↪female_spend.mean() )
```

```
mean of male spend:  71367.390625 | mean of female spend:  63778.25390625
```

0.15 17. Is there any relationship between age category and segment?¶

[24]:
```
obs_freq = pd.crosstab( cust_claims.Segment, cust_claims.AgeGroup )
obs_freq
```

[24]:
```
AgeGroup  Adult  Senior  Youth
Segment
Gold        282      38     56
Platinum    271      34     60
Silver      260      31     60
```

0.16 18. The current year has shown a significant rise in claim amounts as compared to 2016-17 fiscal average which was $10,000.

[55]:
```
new =  cust_claims.groupby(["claim_date"])[["claim_amount"]].sum().reset_index()
new['Monthly'] = new['claim_date'].apply(lambda x:pd.Timestamp.
 ↪strftime(x,format="%B"))
new['Yearly'] = new['claim_date'].apply(lambda x:pd.Timestamp.
 ↪strftime(x,format="%Y"))
new
```

[55]:
```
   claim_date  claim_amount  Monthly Yearly
0  2017-01-01     203227.00  January   2017
1  2017-01-07     137763.50  January   2017
2  2017-01-14     138807.50  January   2017
3  2017-01-21      72576.50  January   2017
4  2017-01-27      57928.00  January   2017
..        …            …      …    …
95 2018-10-03     162797.50  October   2018
96 2018-10-09     225007.50  October   2018
97 2018-10-16     119829.00  October   2018
98 2018-10-23     147084.00  October   2018
99 2018-10-30     120735.50  October   2018

[100 rows x 4 columns]
```

[56]:
```
new_2017 = new.loc[ new.Yearly == '2017', 'claim_amount' ].mean()
print( new_2017 )
```

```
133465.859375
```

[57]:
```
new_2018 = new.loc[ new.Yearly == '2018', 'claim_amount' ]
new_2018.count()
```

[57]:
```
45
```

0.17 19. Is there any difference between age groups and insurance claims?

[25]:
```
age_group_1 = cust_claims['total_policy_claims'].
 ↪loc[cust_claims['AgeGroup']=="Youth"]
age_group_2 = cust_claims['total_policy_claims'].
 ↪loc[cust_claims['AgeGroup']=="Adult"]
# Perfrom the Anova
anova = stats.f_oneway(age_group_1,age_group_2)
# Statistic :  F Value
f = anova.statistic
p = anova.pvalue
print("The f-value is {} and the p value is {}".format(f,p))
if(p<0.05):
    print('We reject null hypothesis')
else:
    print('We fail to reject null hypothesis')
```

```
The f-value is nan and the p value is nan
We fail to reject null hypothesis
```

0.18 20. Is there any relationship between total number of policy claims and the claimed amount?

[23]:
```
cust_claims.total_policy_claims.value_counts()
```

[23]:
```
1.00    781
2.00    104
3.00     86
4.00     58
5.00     27
6.00      9
7.00      2
8.00      1
Name: total_policy_claims, dtype: int64
```

[24]:
```
cust_claims.columns
```

[24]:
```
Index(['gender', 'DateOfBirth', 'State', 'Contact', 'Segment', 'claim_id',
       'incident_cause', 'claim_date', 'claim_area', 'police_report',
       'claim_type', 'claim_amount', 'total_policy_claims', 'fraudulent',
       'Age'],
      dtype='object')
```

[25]:
```
usage = 'claim_amount'
```

[27]:
```
s1 = cust_claims.loc[ cust_claims.total_policy_claims == 1.0, usage ]
s2 = cust_claims.loc[ cust_claims.total_policy_claims == 2.0, usage ]
```

```
s3 = cust_claims.loc[ cust_claims.total_policy_claims == 3.0, usage ]
s4 = cust_claims.loc[ cust_claims.total_policy_claims == 4.0, usage ]
s5 = cust_claims.loc[ cust_claims.total_policy_claims == 5.0, usage ]
s6 = cust_claims.loc[ cust_claims.total_policy_claims == 6.0, usage ]
s7 = cust_claims.loc[ cust_claims.total_policy_claims == 7.0, usage ]
s8 = cust_claims.loc[ cust_claims.total_policy_claims == 8.0, usage ]

print( 'mean s1:', s1.mean(), '| mean s2:', s2.mean(), '| mean s3:', s3.
 ↪mean(),'mean s4:', s4.mean(),
      '| mean s5:', s5.mean(), '| mean s6:', s6.mean(),'|mean s7:', s7.mean(),␣
 ↪'| mean s8:', s2.mean(), '| mean s8:', s3.mean() )
```

```
mean s1: 12403.6552734375 | mean s2: 13876.4521484375 | mean s3: 12928.19140625
mean s4: 11049.724609375 | mean s5: 8530.462890625 | mean s6: 15657.8330078125
|mean s7: 23033.0 | mean s8: 13876.4521484375 | mean s8: 12928.19140625
```

[28]:
```
stats.f_oneway( s1, s2, s3, s4, s5, s6, s7, s8 )
```

[28]:
```
F_onewayResult(statistic=0.9303871872119974, pvalue=0.4818261589585725)
```

[29]:
```
print('yes there is relation between total number of policy claims and the␣
 ↪claimed amount')
```

```
yes there is relation between total number of policy claims and the claimed
amount
```

[ ]:
```

```