In [64]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [65]:
```
df=pd.read_csv('https://raw.githubusercontent.com/dsrscientist/Data-Science-ML-Capstone-Projects/master/Automobil
```

In [66]:
```
df.head(40)
```

Out[66]:

	months_as_customer	age	policy_number	policy_bind_date	policy_state	policy_csl	policy_deductable	policy_annual_premium	umbrella_lir
0	328	48	521585	17-10-2014	OH	250/500	1000	1406.91	
1	228	42	342868	27-06-2006	IN	250/500	2000	1197.22	50000
2	134	29	687698	06-09-2000	OH	100/300	2000	1413.14	50000
3	256	41	227811	25-05-1990	IL	250/500	2000	1415.74	60000
4	228	44	367455	06-06-2014	IL	500/1000	1000	1583.91	60000
5	256	39	104594	12-10-2006	OH	250/500	1000	1351.10	
6	137	34	413978	04-06-2000	IN	250/500	1000	1333.35	
7	165	37	429027	03-02-1990	IL	100/300	1000	1137.03	
8	27	33	485665	05-02-1997	IL	100/300	500	1442.99	
9	212	42	636550	25-07-2011	IL	100/300	500	1315.68	
10	235	42	543610	26-05-2002	OH	100/300	500	1253.12	40000
11	447	61	214618	29-05-1999	OH	100/300	2000	1137.16	
12	60	23	842643	20-11-1997	OH	500/1000	500	1215.36	30000
13	121	34	626808	26-10-2012	OH	100/300	1000	936.61	
14	180	38	644081	28-12-1998	OH	250/500	2000	1301.13	
15	473	58	892874	19-10-1992	IN	100/300	2000	1131.40	
16	70	26	558938	08-06-2005	OH	500/1000	1000	1199.44	50000
17	140	31	275265	15-11-2004	IN	500/1000	500	708.64	60000
18	160	37	921202	28-12-2014	OH	500/1000	500	1374.22	
19	196	39	143972	02-08-1992	IN	500/1000	2000	1475.73	
20	460	62	183430	25-06-2002	IN	250/500	1000	1187.96	40000
21	217	41	431876	27-11-2005	IL	500/1000	2000	875.15	
22	370	55	285496	27-05-1994	IL	100/300	2000	972.18	
23	413	55	115399	08-02-1991	IN	100/300	2000	1268.79	
24	237	40	736882	02-02-1996	IN	100/300	1000	883.31	
25	8	35	699044	05-12-2013	OH	100/300	2000	1266.92	
26	257	43	863236	20-09-1990	IN	100/300	2000	1322.10	
27	202	34	608513	18-07-2002	IN	100/300	500	848.07	30000
28	224	40	914088	08-02-1990	OH	100/300	2000	1291.70	
29	241	45	596785	04-03-2014	IL	500/1000	2000	1104.50	
30	64	25	908616	18-02-2000	IL	250/500	1000	954.16	
31	166	37	666333	19-06-2008	IL	100/300	2000	1337.28	80000
32	155	35	336614	01-08-2003	IL	500/1000	1000	1088.34	
33	114	30	584859	04-04-1992	IL	100/300	1000	1558.29	
34	149	37	990493	13-01-1991	IL	500/1000	500	1415.68	
35	147	33	129872	08-08-2010	OH	100/300	1000	1334.15	60000
36	62	28	200152	09-03-2003	IL	100/300	1000	988.45	
37	289	49	933293	03-02-1993	IL	500/1000	2000	1222.48	
38	431	54	485664	25-11-2002	IN	500/1000	2000	1155.55	
39	199	37	982871	27-07-1997	IN	250/500	500	1262.08	

40 rows × 40 columns

In [ ]:
```
```

In [67]:
```
df.shape
```

Out[67]:
```
(1000, 40)
```

In [68]:
```
df.dtypes
```

Out[68]:
```
months_as_customer              int64
age                             int64
policy_number                   int64
policy_bind_date               object
policy_state                   object
policy_csl                     object
policy_deductable               int64
policy_annual_premium         float64
umbrella_limit                  int64
insured_zip                     int64
insured_sex                    object
insured_education_level        object
insured_occupation             object
insured_hobbies                object
insured_relationship           object
capital-gains                   int64
capital-loss                    int64
incident_date                  object
incident_type                  object
collision_type                 object
incident_severity              object
authorities_contacted          object
incident_state                 object
incident_city                  object
incident_location              object
incident_hour_of_the_day        int64
number_of_vehicles_involved     int64
property_damage                object
bodily_injuries                 int64
witnesses                       int64
police_report_available        object
total_claim_amount              int64
injury_claim                    int64
property_claim                  int64
vehicle_claim                   int64
auto_make                      object
auto_model                     object
auto_year                       int64
fraud_reported                 object
_c39                          float64
dtype: object
```

In [69]:
```
df.isnull().sum()
```

Out[69]:
```
months_as_customer            0
age                           0
policy_number                 0
policy_bind_date              0
policy_state                  0
policy_csl                    0
policy_deductable             0
policy_annual_premium         0
umbrella_limit                0
insured_zip                   0
insured_sex                   0
insured_education_level       0
insured_occupation            0
insured_hobbies               0
insured_relationship          0
capital-gains                 0
capital-loss                  0
incident_date                 0
incident_type                 0
collision_type                0
incident_severity             0
authorities_contacted         0
incident_state                0
incident_city                 0
incident_location             0
incident_hour_of_the_day      0
```

```
number_of_vehicles_involved       0
property_damage                   0
bodily_injuries                   0
witnesses                         0
police_report_available           0
total_claim_amount                0
injury_claim                      0
property_claim                    0
vehicle_claim                     0
auto_make                         0
auto_model                        0
auto_year                         0
fraud_reported                    0
_c39                           1000
dtype: int64
```

In [70]:
```
## remove the LAST COLUMN
```

In [71]:
```
df=df.drop(['_c39','policy_bind_date'],axis=1)
df
```

Out[71]:

	months_as_customer	age	policy_number	policy_state	policy_csl	policy_deductable	policy_annual_premium	umbrella_limit	insured_zip
0	328	48	521585	OH	250/500	1000	1406.91	0	466132
1	228	42	342868	IN	250/500	2000	1197.22	5000000	468176
2	134	29	687698	OH	100/300	2000	1413.14	5000000	430632
3	256	41	227811	IL	250/500	2000	1415.74	6000000	608117
4	228	44	367455	IL	500/1000	1000	1583.91	6000000	610706
...	...	...	...	...	...	...	...	...	...
995	3	38	941851	OH	500/1000	1000	1310.80	0	431289
996	285	41	186934	IL	100/300	1000	1436.79	0	608177
997	130	34	918516	OH	250/500	500	1383.49	3000000	442797
998	458	62	533940	IL	500/1000	2000	1356.92	5000000	441714
999	456	60	556080	OH	250/500	1000	766.19	0	612260

1000 rows × 38 columns

In [72]:
```
df=df.drop(['incident_hour_of_the_day','number_of_vehicles_involved','bodily_injuries','witnesses','policy_numbe
df
```

Out[72]:

	months_as_customer	age	policy_state	policy_csl	policy_deductable	policy_annual_premium	umbrella_limit	insured_zip	insured_sex	ins
0	328	48	OH	250/500	1000	1406.91	0	466132	MALE	
1	228	42	IN	250/500	2000	1197.22	5000000	468176	MALE	
2	134	29	OH	100/300	2000	1413.14	5000000	430632	FEMALE	
3	256	41	IL	250/500	2000	1415.74	6000000	608117	FEMALE	
4	228	44	IL	500/1000	1000	1583.91	6000000	610706	MALE	
...	...	...	...	...	...	...	...	...	...	
995	3	38	OH	500/1000	1000	1310.80	0	431289	FEMALE	
996	285	41	IL	100/300	1000	1436.79	0	608177	FEMALE	
997	130	34	OH	250/500	500	1383.49	3000000	442797	FEMALE	
998	458	62	IL	500/1000	2000	1356.92	5000000	441714	MALE	
999	456	60	OH	250/500	1000	766.19	0	612260	FEMALE	

1000 rows × 33 columns

In [73]:
```
df
```

Out[73]:

	months_as_customer	age	policy_state	policy_csl	policy_deductable	policy_annual_premium	umbrella_limit	insured_zip	insured_sex	ins
0	328	48	OH	250/500	1000	1406.91	0	466132	MALE	
1	228	42	IN	250/500	2000	1197.22	5000000	468176	MALE	

2	134	29	OH	100/300	2000	1413.14	5000000	430632	FEMALE
3	256	41	IL	250/500	2000	1415.74	6000000	608117	FEMALE
4	228	44	IL	500/1000	1000	1583.91	6000000	610706	MALE
...	...	...	...	...	...	...	...	...	...
995	3	38	OH	500/1000	1000	1310.80	0	431289	FEMALE
996	285	41	IL	100/300	1000	1436.79	0	608177	FEMALE
997	130	34	OH	250/500	500	1383.49	3000000	442797	FEMALE
998	458	62	IL	500/1000	2000	1356.92	5000000	441714	MALE
999	456	60	OH	250/500	1000	766.19	0	612260	FEMALE

1000 rows × 33 columns

In [74]:
```
df.describe()
```

Out[74]:

	months_as_customer	age	policy_deductable	policy_annual_premium	umbrella_limit	insured_zip	capital-gains	capital-lo
count	1000.000000	1000.000000	1000.000000	1000.000000	1.000000e+03	1000.000000	1000.000000	1000.0000
mean	203.954000	38.948000	1136.000000	1256.406150	1.101000e+06	501214.488000	25126.100000	-26793.7000
std	115.113174	9.140287	611.864673	244.167395	2.297407e+06	71701.610941	27872.187708	28104.0966
min	0.000000	19.000000	500.000000	433.330000	-1.000000e+06	430104.000000	0.000000	-111100.0000
25%	115.750000	32.000000	500.000000	1089.607500	0.000000e+00	448404.500000	0.000000	-51500.0000
50%	199.500000	38.000000	1000.000000	1257.200000	0.000000e+00	466445.500000	0.000000	-23250.0000
75%	276.250000	44.000000	2000.000000	1415.695000	0.000000e+00	603251.000000	51025.000000	0.0000
max	479.000000	64.000000	2000.000000	2047.590000	1.000000e+07	620962.000000	100500.000000	0.0000

In [75]:
```
# Checking number of unique values in each columns
count = 1
for x in df:
    print(f'{count}. {x}: {df[x].nunique()}')
    print(f'{df[x].value_counts()}', end = '\n----------\n\n' )
    count += 1
```

```
1. months_as_customer: 391
194    8
254    7
210    7
101    7
140    7
      ..
312    1
62     1
309    1
308    1
0      1
Name: months_as_customer, Length: 391, dtype: int64
----------

2. age: 46
43    49
39    48
41    45
34    44
38    42
30    42
31    42
37    41
33    39
40    38
32    38
29    35
46    33
35    32
44    32
36    32
42    32
28    30
45    26
26    26
48    25
47    24
```

```
27    24
57    16
55    14
25    14
49    14
50    13
53    13
24    10
54    10
61    10
51     9
60     9
56     8
58     8
23     7
21     6
59     5
52     4
62     4
63     2
64     2
20     1
22     1
19     1
Name: age, dtype: int64
----------

3. policy_state: 3
OH    352
IL    338
IN    310
Name: policy_state, dtype: int64
----------

4. policy_csl: 3
250/500     351
100/300     349
500/1000    300
Name: policy_csl, dtype: int64
----------

5. policy_deductable: 3
1000    351
500     342
2000    307
Name: policy_deductable, dtype: int64
----------

6. policy_annual_premium: 991
1215.36    2
1374.22    2
1389.13    2
1281.25    2
1074.07    2
          ..
1185.44    1
1243.84    1
1270.02    1
1023.11    1
1337.56    1
Name: policy_annual_premium, Length: 991, dtype: int64
----------

7. umbrella_limit: 11
 0           798
 6000000      57
 5000000      46
 4000000      39
 7000000      29
 3000000      12
 8000000       8
 9000000       5
 2000000       3
 10000000      2
-1000000       1
Name: umbrella_limit, dtype: int64
----------

8. insured_zip: 995
469429    2
446895    2
431202    2
456602    2
```

```
477695    2
          ..
445120    1
608963    1
449221    1
467654    1
448722    1
Name: insured_zip, Length: 995, dtype: int64
----------

9. insured_sex: 2
FEMALE    537
MALE      463
Name: insured_sex, dtype: int64
----------

10. insured_education_level: 7
JD              161
High School     160
Associate       145
MD              144
Masters         143
PhD             125
College         122
Name: insured_education_level, dtype: int64
----------

11. insured_occupation: 14
machine-op-inspct    93
prof-specialty       85
tech-support         78
exec-managerial      76
sales                76
craft-repair         74
transport-moving     72
priv-house-serv      71
other-service        71
armed-forces         69
adm-clerical         65
protective-serv      63
handlers-cleaners    54
farming-fishing      53
Name: insured_occupation, dtype: int64
----------

12. insured_hobbies: 20
reading           64
paintball         57
exercise          57
bungie-jumping    56
camping           55
golf              55
movies            55
kayaking          54
yachting          53
hiking            52
video-games       50
skydiving         49
base-jumping      49
board-games       48
polo              47
chess             46
dancing           43
sleeping          41
cross-fit         35
basketball        34
Name: insured_hobbies, dtype: int64
----------

13. insured_relationship: 6
own-child         183
other-relative    177
not-in-family     174
husband           170
wife              155
unmarried         141
Name: insured_relationship, dtype: int64
----------

14. capital-gains: 338
0        508
46300      5
68500      4
```

```
51500      4
45500      3
        ...
54700      1
40100      1
33200      1
37300      1
72700      1
Name: capital-gains, Length: 338, dtype: int64
----------

15. capital-loss: 354
 0        475
-53700      5
-50300      5
-31700      5
-51000      4
         ...
-43300      1
-66100      1
-55900      1
-66500      1
-48000      1
Name: capital-loss, Length: 354, dtype: int64
----------

16. incident_date: 60
02-02-2015    28
17-02-2015    26
07-01-2015    25
10-01-2015    24
04-02-2015    24
24-01-2015    24
19-01-2015    23
08-01-2015    22
13-01-2015    21
30-01-2015    21
22-02-2015    20
31-01-2015    20
06-02-2015    20
12-02-2015    20
14-01-2015    19
23-02-2015    19
21-02-2015    19
01-01-2015    19
12-01-2015    19
21-01-2015    19
01-02-2015    18
25-02-2015    18
20-01-2015    18
18-01-2015    18
28-02-2015    18
03-01-2015    18
14-02-2015    18
09-01-2015    17
26-02-2015    17
06-01-2015    17
08-02-2015    17
24-02-2015    17
13-02-2015    16
16-01-2015    16
16-02-2015    16
15-02-2015    16
05-02-2015    16
15-01-2015    15
18-02-2015    15
28-01-2015    15
17-01-2015    15
27-02-2015    14
20-02-2015    14
22-01-2015    14
03-02-2015    13
09-02-2015    13
23-01-2015    13
27-01-2015    13
01-03-2015    12
04-01-2015    12
02-01-2015    11
26-01-2015    11
29-01-2015    11
10-02-2015    10
25-01-2015    10
07-02-2015    10
```

```
19-02-2015    10
11-02-2015    10
11-01-2015     9
05-01-2015     7
Name: incident_date, dtype: int64
----------

17. incident_type: 4
Multi-vehicle Collision     419
Single Vehicle Collision    403
Vehicle Theft                94
Parked Car                   84
Name: incident_type, dtype: int64
----------

18. collision_type: 4
Rear Collision     292
Side Collision     276
Front Collision    254
?                  178
Name: collision_type, dtype: int64
----------

19. incident_severity: 4
Minor Damage      354
Total Loss        280
Major Damage      276
Trivial Damage     90
Name: incident_severity, dtype: int64
----------

20. authorities_contacted: 5
Police       292
Fire         223
Other        198
Ambulance    196
None          91
Name: authorities_contacted, dtype: int64
----------

21. incident_state: 7
NY    262
SC    248
WV    217
VA    110
NC    110
PA     30
OH     23
Name: incident_state, dtype: int64
----------

22. incident_city: 7
Springfield    157
Arlington      152
Columbus       149
Northbend      145
Hillsdale      141
Riverwood      134
Northbrook     122
Name: incident_city, dtype: int64
----------

23. incident_location: 1000
4910 1st Lane       1
3805 Lincoln Hwy    1
6608 MLK Hwy        1
1916 Elm St         1
7571 Elm Ridge      1
                   ..
6484 Tree Drive     1
8906 Elm Lane       1
1956 Apache St      1
8917 Tree Ridge     1
5639 1st Ridge      1
Name: incident_location, Length: 1000, dtype: int64
----------

24. property_damage: 3
?      360
NO     338
YES    302
Name: property_damage, dtype: int64
----------
```

```
25. police_report_available: 3
?      343
NO     343
YES    314
Name: police_report_available, dtype: int64
----------

26. total_claim_amount: 763
59400    5
2640     4
75400    4
60600    4
5940     4
        ..
51590    1
31700    1
53640    1
63100    1
88920    1
Name: total_claim_amount, Length: 763, dtype: int64
----------

27. injury_claim: 638
0        25
640       7
480       7
1180      5
860       5
         ..
10800     1
12580     1
6700      1
4650      1
16500     1
Name: injury_claim, Length: 638, dtype: int64
----------

28. property_claim: 626
0        19
860       6
640       5
480       5
10000     5
         ..
5350      1
6700      1
20550     1
4650      1
11260     1
Name: property_claim, Length: 626, dtype: int64
----------

29. vehicle_claim: 726
5040     7
3360     6
52080    5
44800    5
4720     5
        ..
46480    1
38290    1
5520     1
26720    1
22800    1
Name: vehicle_claim, Length: 726, dtype: int64
----------

30. auto_make: 14
Saab          80
Dodge         80
Suburu        80
Nissan        78
Chevrolet     76
BMW           72
Ford          72
Toyota        70
Audi          69
Accura        68
Volkswagen    68
Jeep          67
Mercedes      65
Honda         55
```

```
Name: auto_make, dtype: int64
----------

31. auto_model: 39
RAM               43
Wrangler          42
Neon              37
A3                37
MDX               36
Jetta             35
Passat            33
A5                32
Legacy            32
Pathfinder        31
Malibu            30
Camry             28
92x               28
Forrestor         28
F150              27
E400              27
95                27
Grand Cherokee    25
93                25
Maxima            24
Escape            24
Tahoe             24
Ultima            23
X5                23
Highlander        22
Silverado         22
Civic             22
Fusion            21
TL                20
CRV               20
ML350             20
Impreza           20
Corolla           20
3 Series          18
C300              18
X6                16
M5                15
Accord            13
RSX               12
Name: auto_model, dtype: int64
----------

32. auto_year: 21
1995    56
1999    55
2005    54
2006    53
2011    53
2007    52
2003    51
2010    50
2009    50
2013    49
2002    49
2015    47
1997    46
2012    46
2008    45
2014    44
2001    42
2000    42
1998    40
2004    39
1996    37
Name: auto_year, dtype: int64
----------

33. fraud_reported: 2
N    753
Y    247
Name: fraud_reported, dtype: int64
----------
```

In [76]:

```
## extracting the continuous features
```

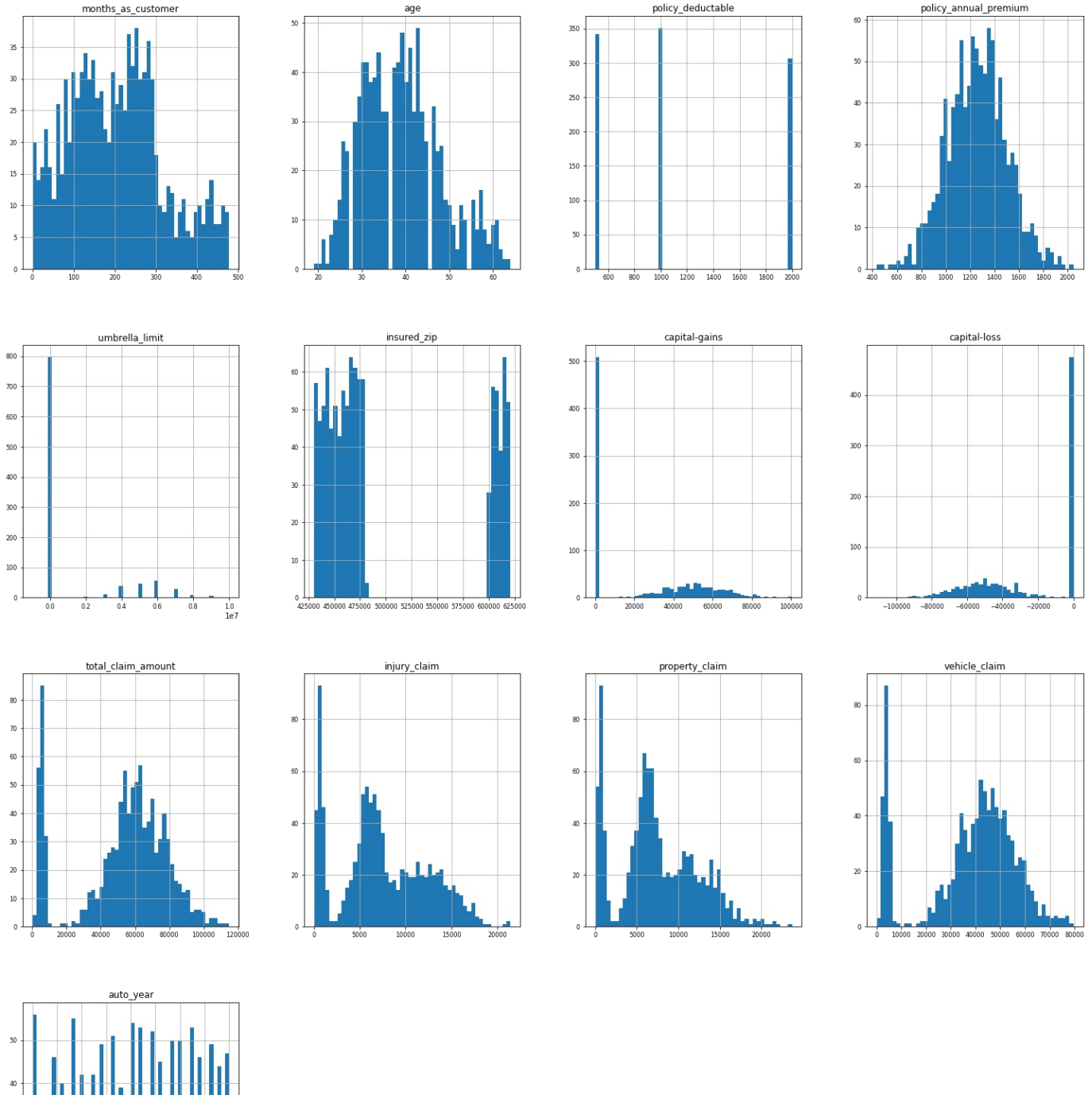
In [77]:
```
cont_data = df.select_dtypes(exclude = ['object'] )
cont_data
```

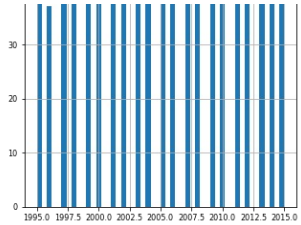
Out[77]:

	months_as_customer	age	policy_deductable	policy_annual_premium	umbrella_limit	insured_zip	capital-gains	capital-loss	total_claim_amount	inj
0	328	48	1000	1406.91	0	466132	53300	0	71610	
1	228	42	2000	1197.22	5000000	468176	0	0	5070	
2	134	29	2000	1413.14	5000000	430632	35100	0	34650	
3	256	41	2000	1415.74	6000000	608117	48900	-62400	63400	
4	228	44	1000	1583.91	6000000	610706	66000	-46000	6500	
...	...	...	...	...	...	...	...	...	...	
995	3	38	1000	1310.80	0	431289	0	0	87200	
996	285	41	1000	1436.79	0	608177	70900	0	108480	
997	130	34	500	1383.49	3000000	442797	35100	0	67500	
998	458	62	2000	1356.92	5000000	441714	0	0	46980	
999	456	60	1000	766.19	0	612260	0	0	5060	

1000 rows × 13 columns

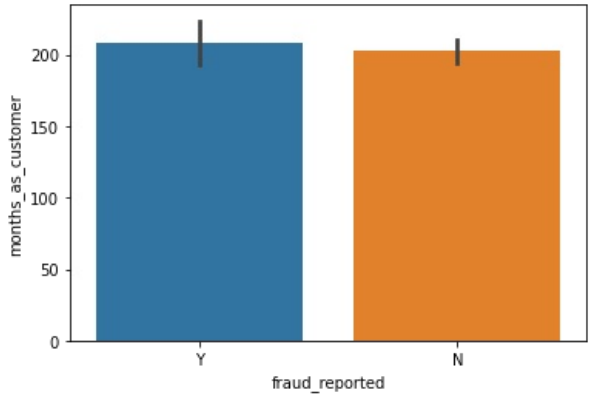
In [78]:
```
cont_data.hist(figsize = (25, 30), bins = 50, xlabelsize = 8, ylabelsize = 8)
plt.show()
```

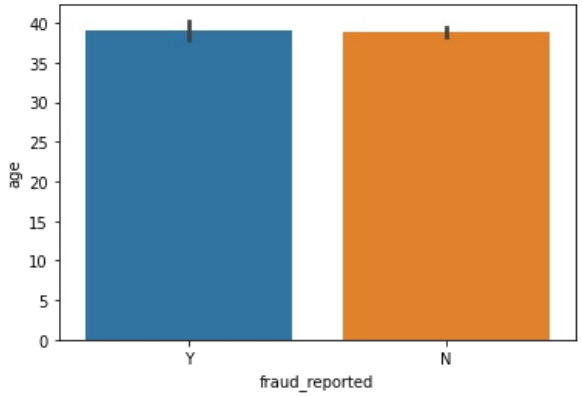


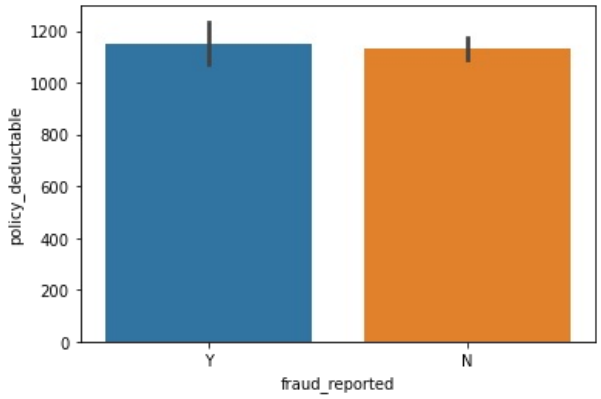


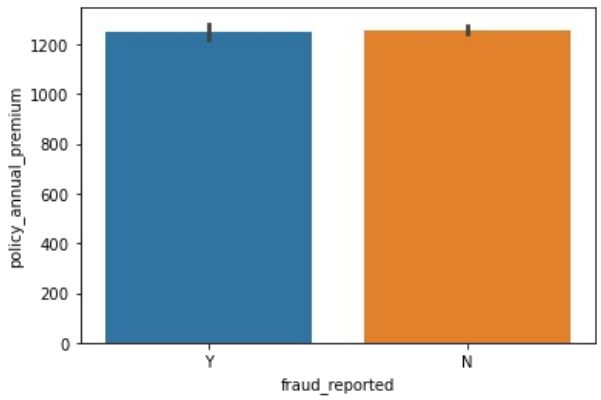
In [79]:

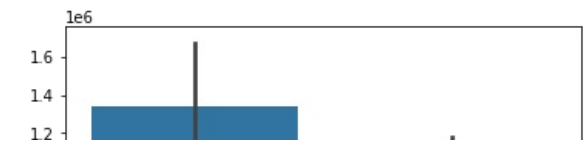
```
for i in cont_data:
    sns.barplot(y = cont_data[i], x = df['fraud_reported'])
    plt.show()
```



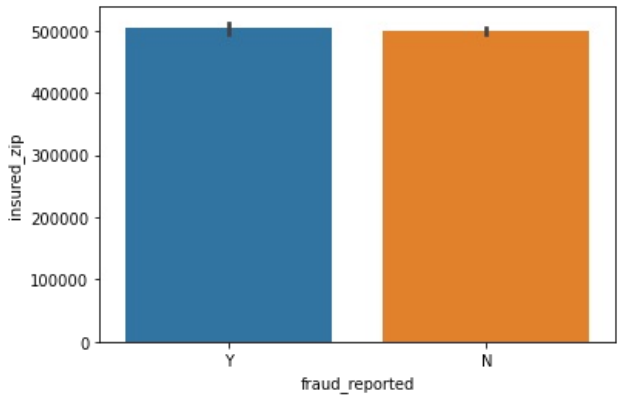


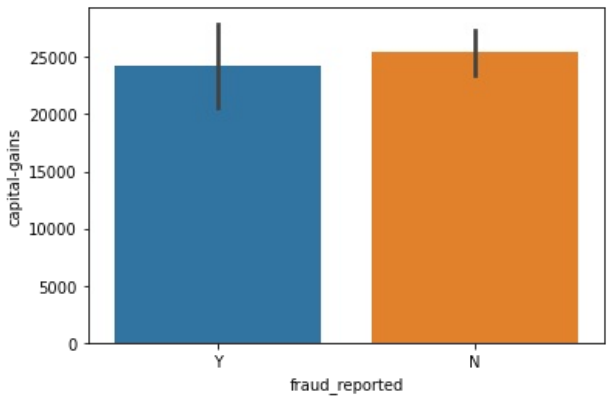


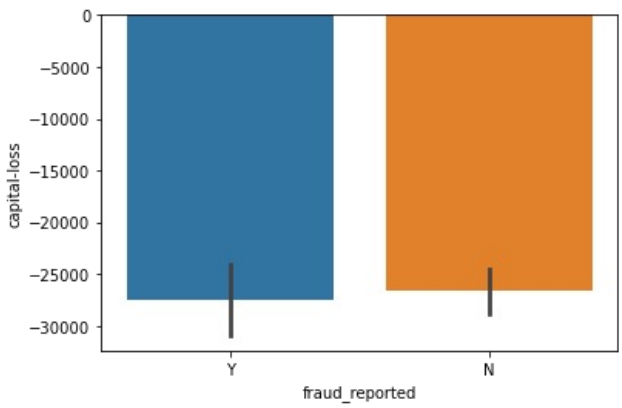




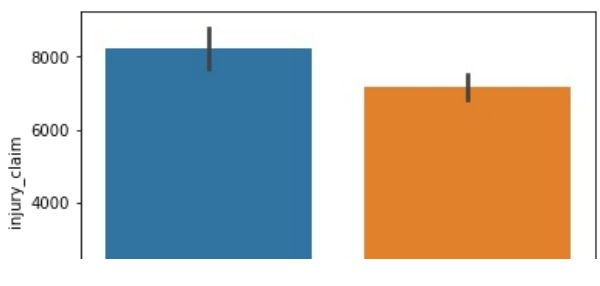

umbrella_limit
1.0
0.8
0.6
0.4
0.2
0.0
Y
N
fraud_reported

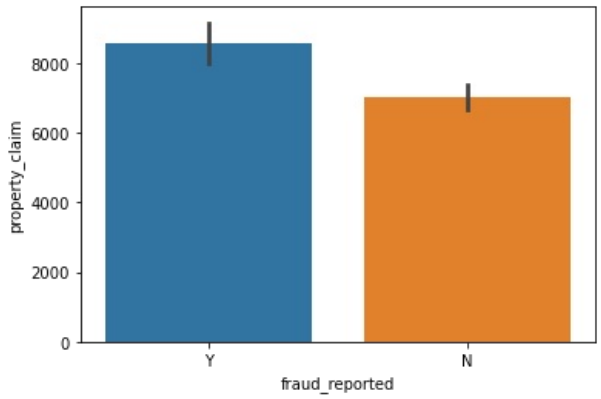

insured_zip
500000
400000
300000
200000
100000
0
Y
N
fraud_reported

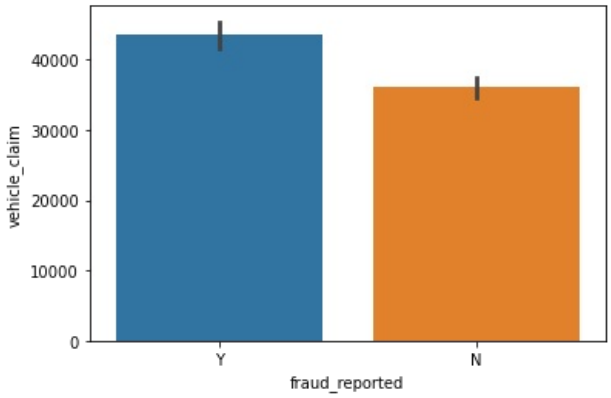

capital-gains
25000
20000
15000
10000
5000
0
Y
N
fraud_reported

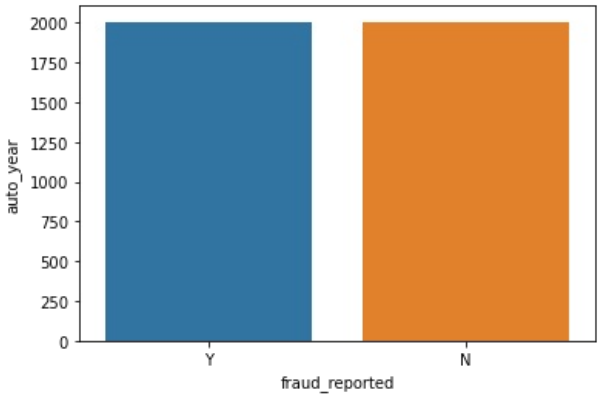

capital-loss
0
−5000
−10000
−15000
−20000
−25000
−30000
Y
N
fraud_reported


total_claim_amount
60000
50000
40000
30000
20000
10000
0
Y
N
fraud_reported


injury_claim
8000
6000
4000

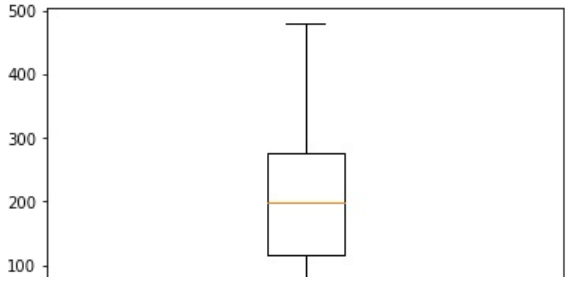




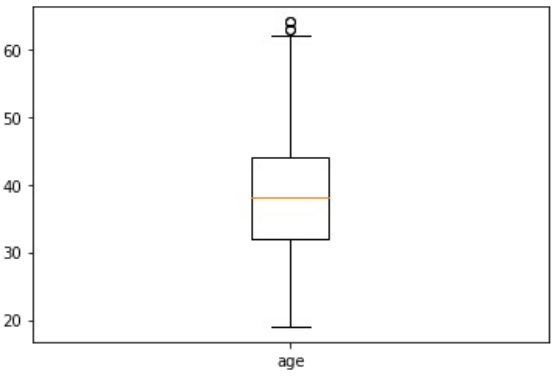


In [80]:
```
## checking for outliers
```

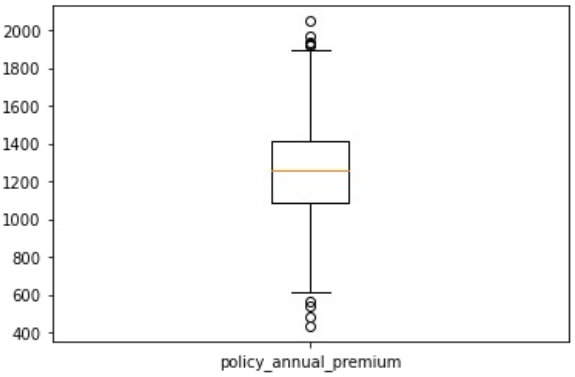
In [81]:
```
for i in cont_data:
    plt.boxplot(cont_data[i], labels = [i])
    plt.show()
```

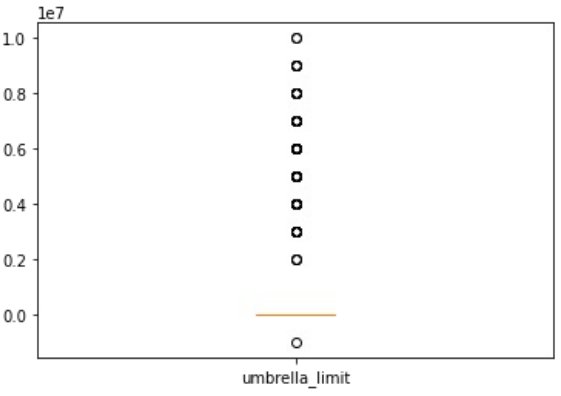


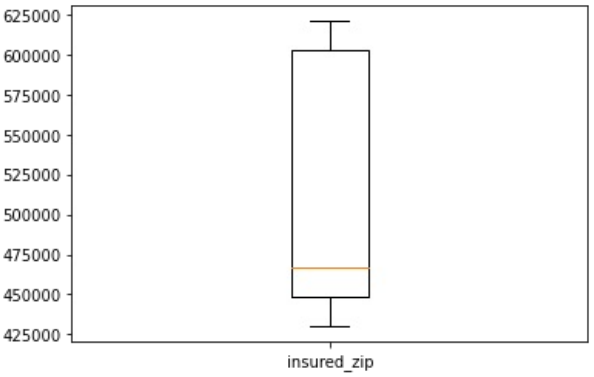

0
months_as_customer

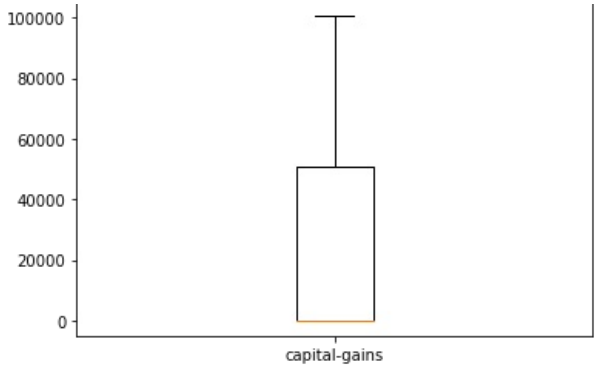

60
50
40
30
20
age

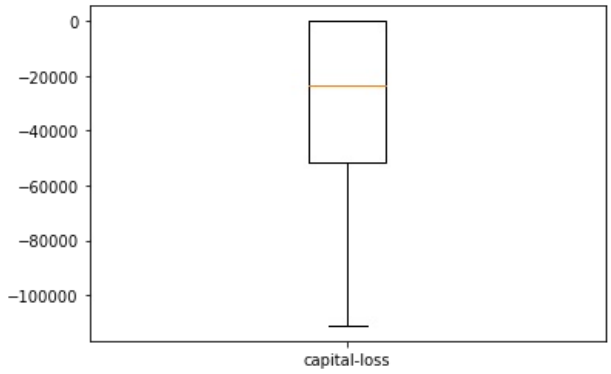

2000
1800
1600
1400
1200
1000
800
600
policy_deductable

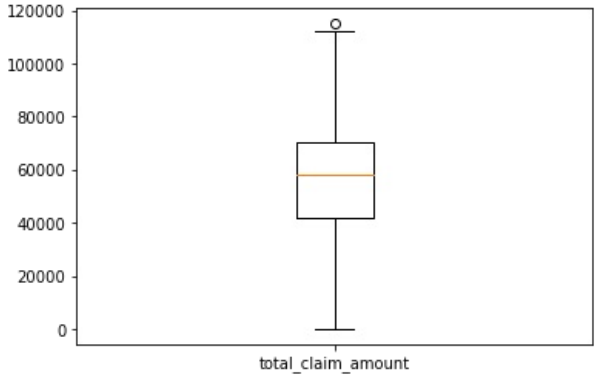

2000
1800
1600
1400
1200
1000
800
600
400
policy_annual_premium

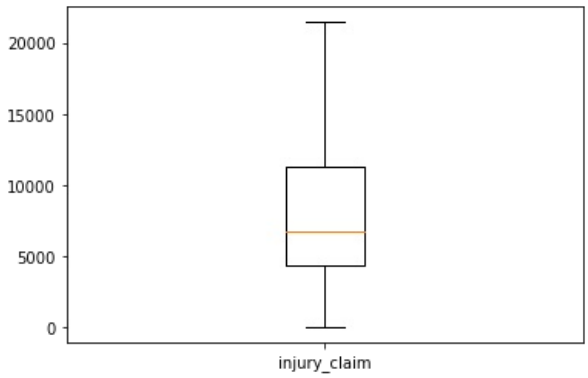

1e7
1.0
0.8
0.6
0.4
0.2
0.0
umbrella_limit

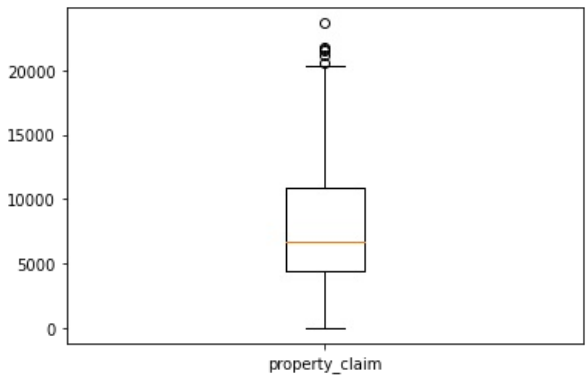

625000
600000
575000
550000
525000
500000
475000
450000
425000
insured_zip

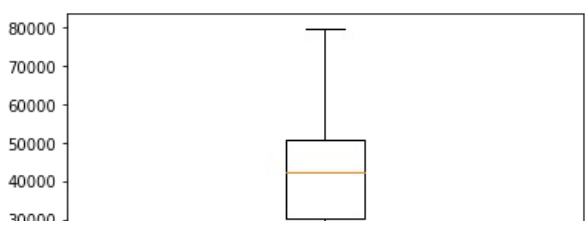

100000
80000
60000
40000
20000
0
capital-gains

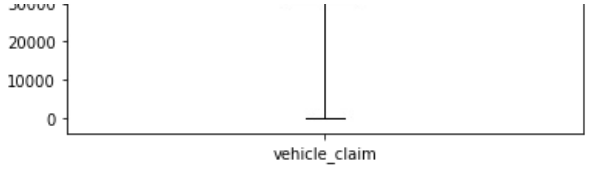

0
−20000
−40000
−60000
−80000
−100000
capital-loss

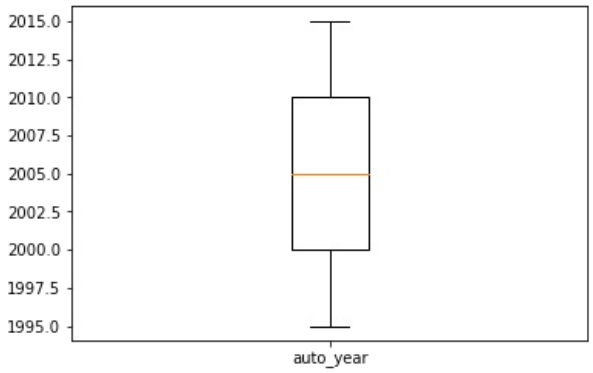

120000
100000
80000
60000
40000
20000
0
total_claim_amount


20000
15000
10000
5000
0
injury_claim


20000
15000
10000
5000
0
property_claim

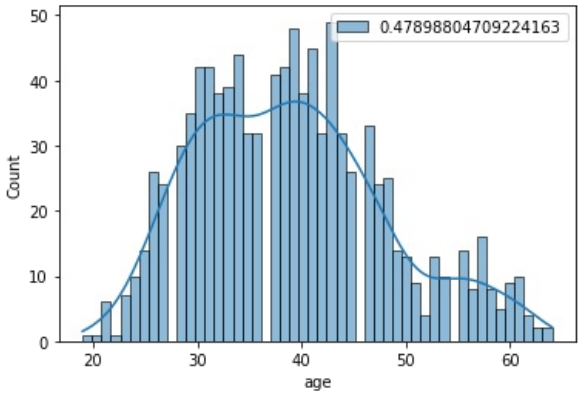

80000
70000
60000
50000
40000

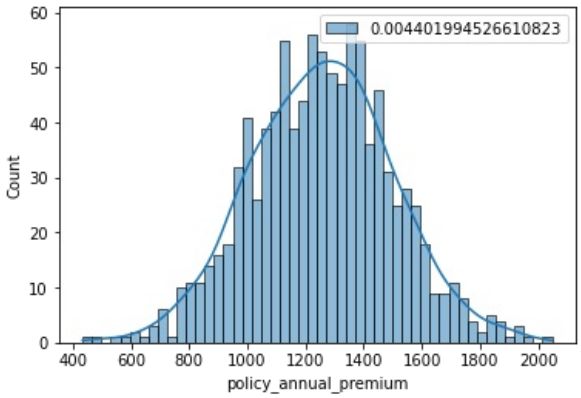


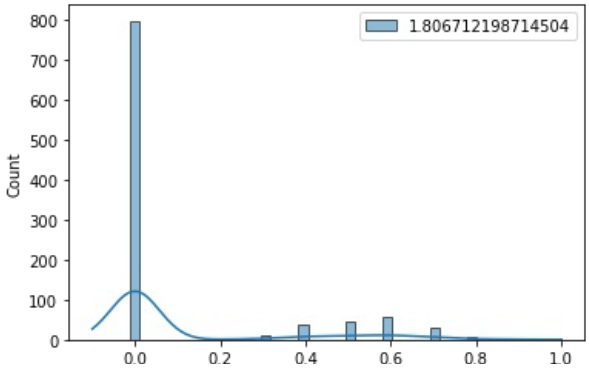


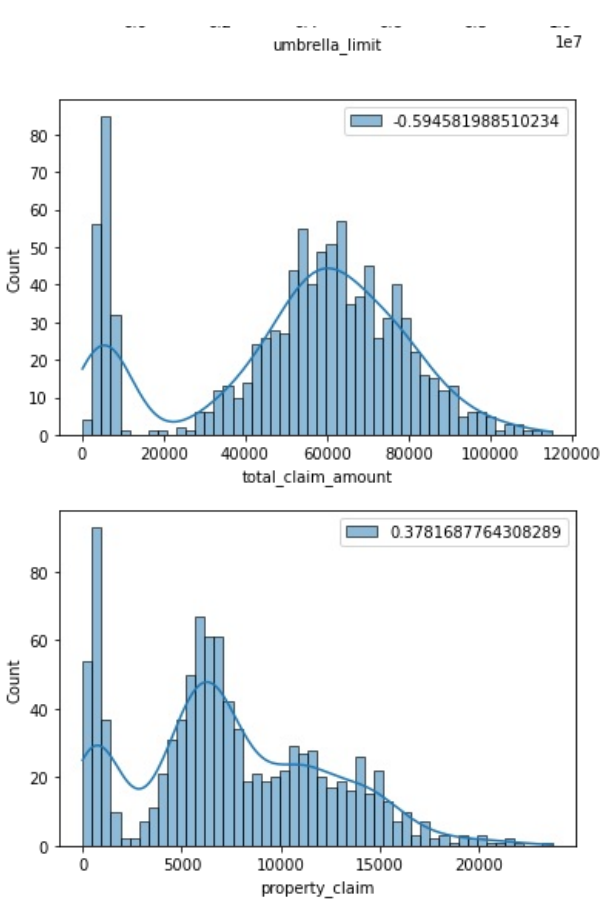
In [82]:
```
a=['age','policy_annual_premium','umbrella_limit','total_claim_amount','property_claim']
```

In [83]:
```
for i in a:
    sns.histplot(cont_data[i], kde = True, bins = 50, label = cont_data[i].skew())
    plt.legend(loc = 'upper right')
    plt.show()
```









In [85]:
```
out_vars=['age','policy_annual_premium','umbrella_limit','total_claim_amount','property_claim']
```

In [86]:
```
def outlierTreat(x):
    upper = x.quantile(.75) + 1.5 * (x.quantile(.75) - x.quantile(.25))
    lower = x.quantile(.25) - 1.5 * (x.quantile(.75) - x.quantile(.25))
    return x.clip(lower, upper)
```

In [87]:
```
cont_data.loc[:, out_vars] = cont_data.loc[:, out_vars].apply(outlierTreat)
cont_data.loc[:, out_vars]
```

```
C:\Users\Rakesh Lodem\anaconda3\lib\site-packages\pandas\core\indexing.py:1787: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(loc, val, pi)
```

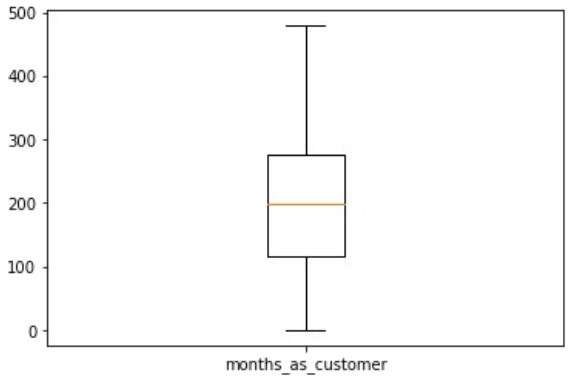
Out[87]:

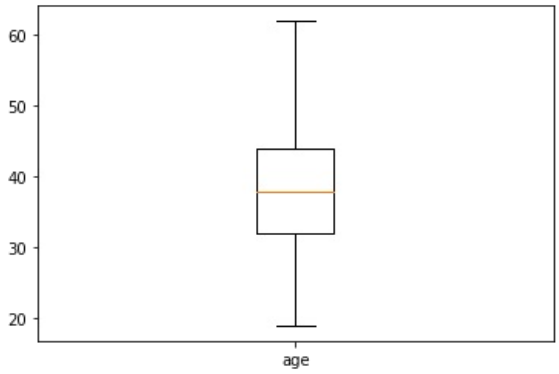
	age	policy_annual_premium	umbrella_limit	total_claim_amount	property_claim
0	48	1406.91	0	71610.0	13020
1	42	1197.22	0	5070.0	780
2	29	1413.14	0	34650.0	3850
3	41	1415.74	0	63400.0	6340
4	44	1583.91	0	6500.0	650
...	...	...	...	...	...
995	38	1310.80	0	87200.0	8720
996	41	1436.79	0	108480.0	18080
997	34	1383.49	0	67500.0	7500
998	62	1356.92	0	46980.0	5220
999	60	766.19	0	5060.0	920

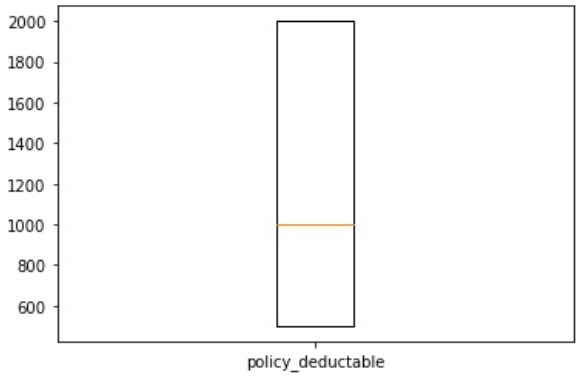
1000 rows × 5 columns

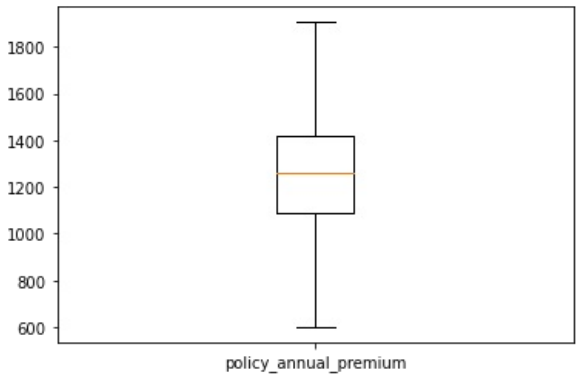
In [88]:
```
# Using box plot for checking the presence of outliers.
```

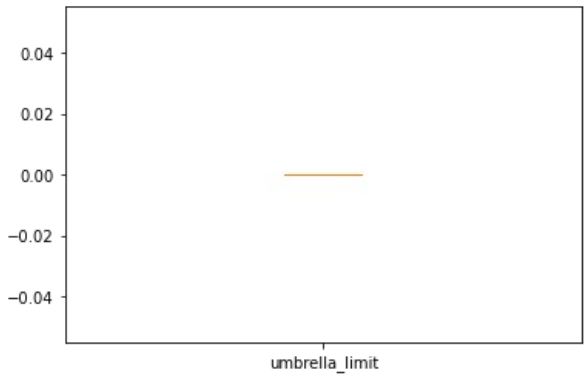
```
for i in cont_data:
    plt.boxplot(cont_data[i], labels = [i])
    plt.show()
```

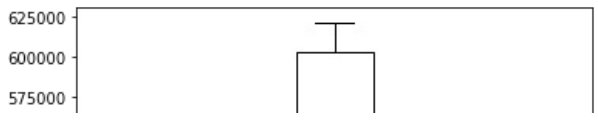


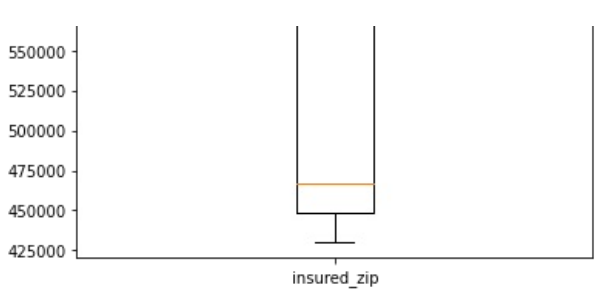


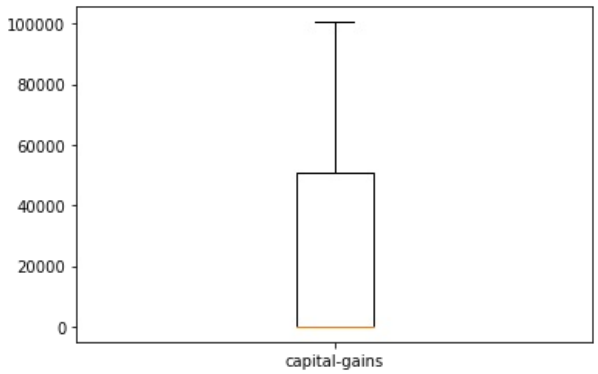


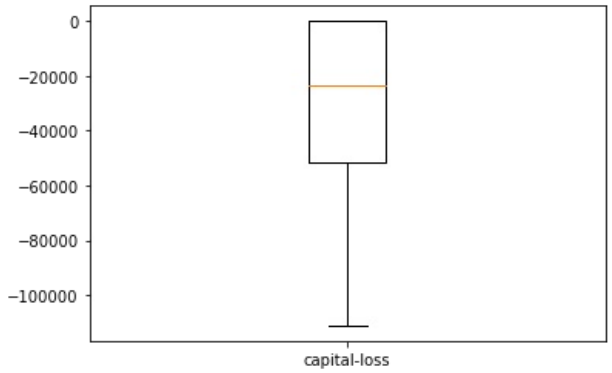


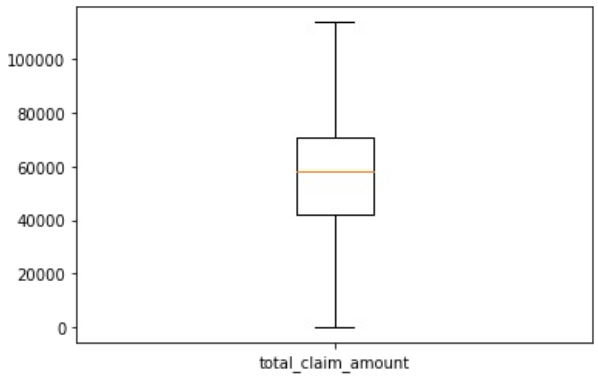


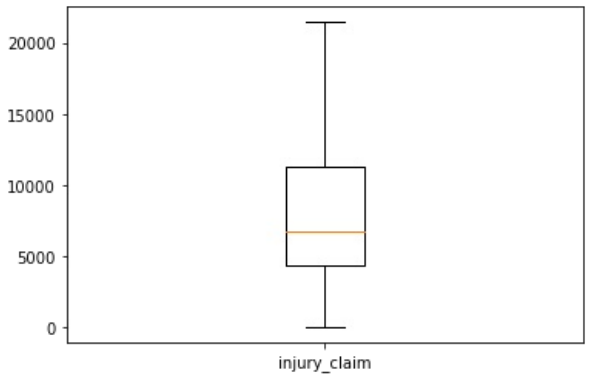


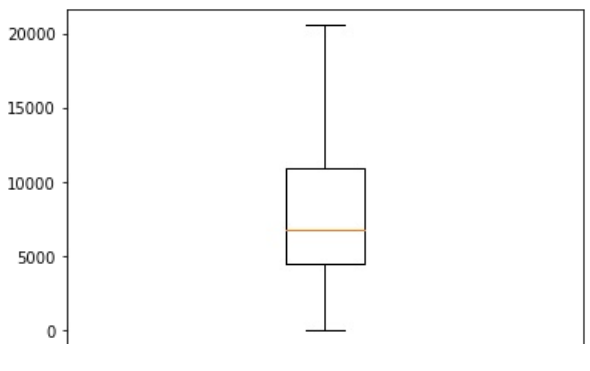

550000
525000
500000
475000
450000
425000
insured_zip

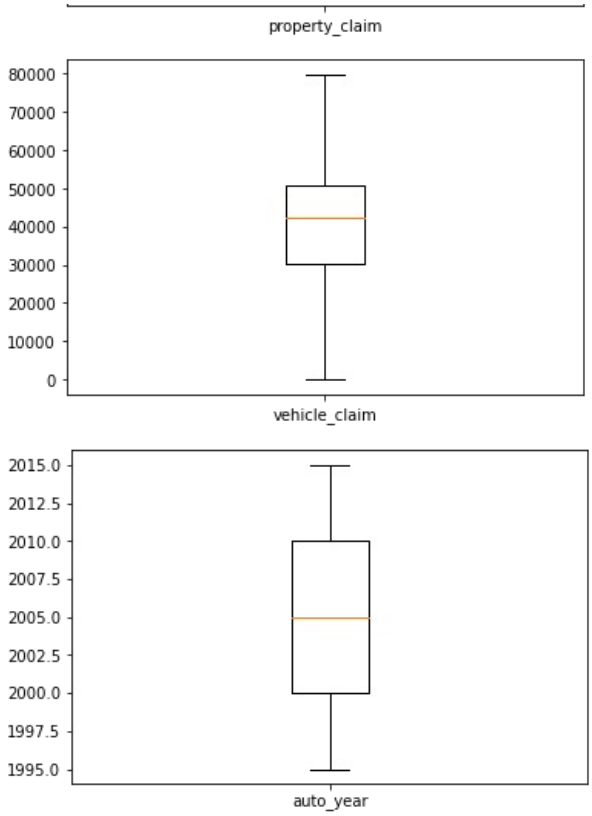

100000
80000
60000
40000
20000
0
capital-gains


0
−20000
−40000
−60000
−80000
−100000
capital-loss


100000
80000
60000
40000
20000
0
total_claim_amount

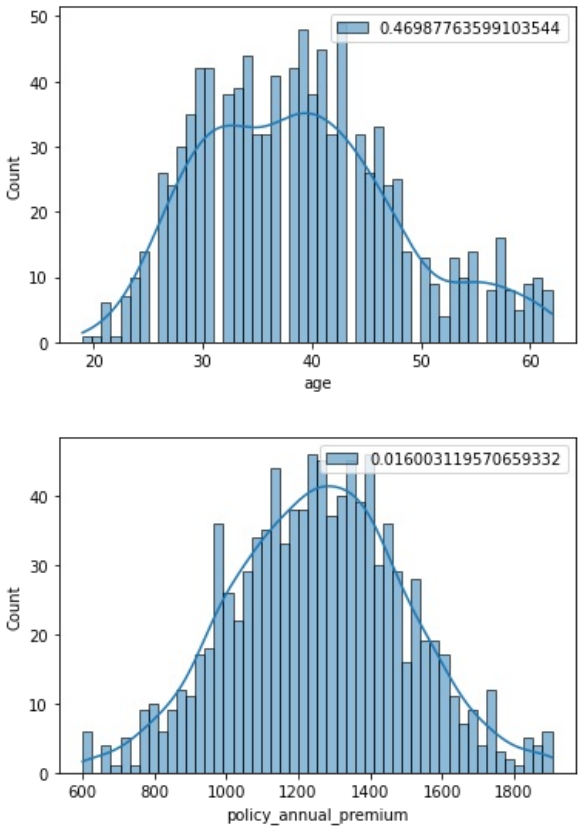

20000
15000
10000
5000
0
injury_claim


20000
15000
10000
5000
0



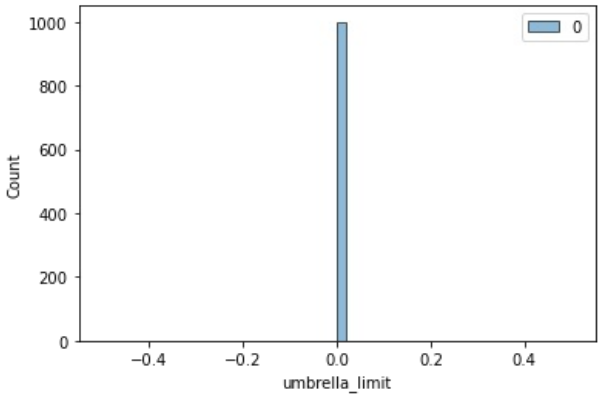
In [90]:

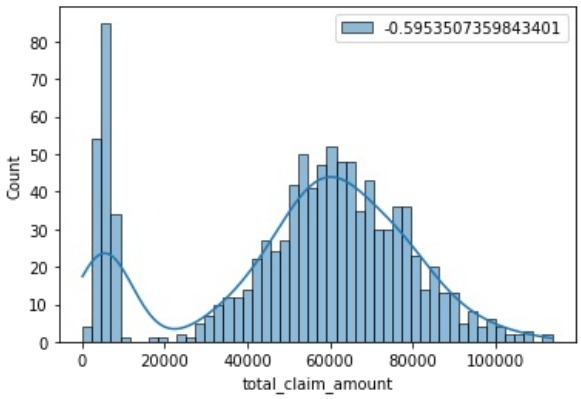
```
for i in out_vars:
    sns.histplot(cont_data[i], kde = True, bins = 50, label = cont_data[i].skew())
    plt.legend(loc = 'upper right')
    plt.show()
```

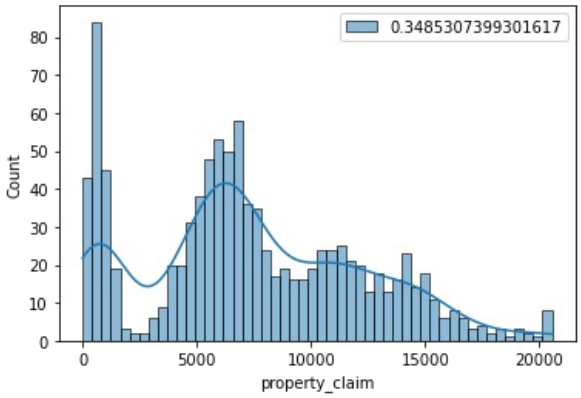



```
C:\Users\Rakesh Lodem\anaconda3\lib\site-packages\seaborn\distributions.py:306: UserWarning: Dataset has 0 varian
ce; skipping density estimate.
```

```
warnings.warn(msg, UserWarning)
```







In [92]:
```
cont_data
```

Out[92]:

	months_as_customer	age	policy_deductable	policy_annual_premium	umbrella_limit	insured_zip	capital-gains	capital-loss	total_claim_amount	inj
0	328	48	1000	1406.91	0	466132	53300	0	71610.0	
1	228	42	2000	1197.22	0	468176	0	0	5070.0	
2	134	29	2000	1413.14	0	430632	35100	0	34650.0	
3	256	41	2000	1415.74	0	608117	48900	-62400	63400.0	
4	228	44	1000	1583.91	0	610706	66000	-46000	6500.0	
...	...	...	...	...	...	...	...	...	...	
995	3	38	1000	1310.80	0	431289	0	0	87200.0	
996	285	41	1000	1436.79	0	608177	70900	0	108480.0	
997	130	34	500	1383.49	0	442797	35100	0	67500.0	
998	458	62	2000	1356.92	0	441714	0	0	46980.0	
999	456	60	1000	766.19	0	612260	0	0	5060.0	

1000 rows × 13 columns

In [91]:
```
# Finding the correlation.
corr = cont_data.corr()
```

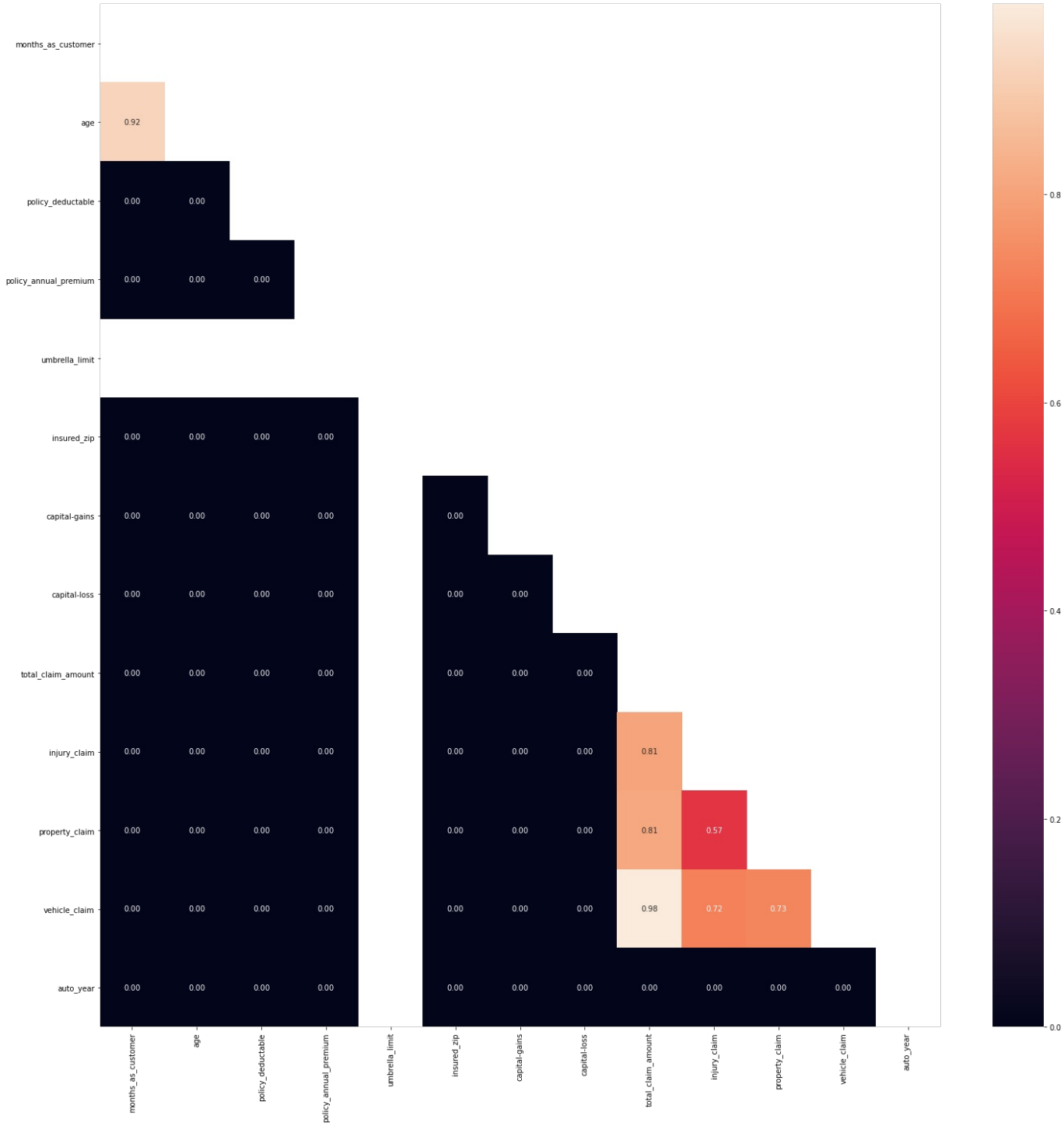
```
# Setting the size of figure.
plt.rcParams['figure.figsize'] = (25, 25)

# Argument Trimming out the values above the main diagonal.
mask = np.triu(corr)

# Setting low correlation value to 0.
corr[(corr.values < 0.3) & (corr.values > -0.3)] = 0

# Plotting the heatmap.
sns.heatmap(corr, annot = True, fmt = '.2f', mask = mask)
```

Out[91]: <AxesSubplot:>



In [93]:

```
def correlation(dataset,threshold):
    col_corr=set()
    corr_matrix=dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i,j])>threshold:
                colname=corr_matrix.columns[i]
                col_corr.add(colname)
                return col_corr
```

In [95]:
```
corr_features=correlation(cont_data,0.7)
len(set(corr_features))
```

Out[95]: 1

In [96]:
```
corr_features
```

Out[96]: {'age'}

In [97]:
```
cont_data.drop(['age'],axis=1)
```

Out[97]:

	months_as_customer	policy_deductable	policy_annual_premium	umbrella_limit	insured_zip	capital-gains	capital-loss	total_claim_amount	injury_cl
0	328	1000	1406.91	0	466132	53300	0	71610.0	6
1	228	2000	1197.22	0	468176	0	0	5070.0	
2	134	2000	1413.14	0	430632	35100	0	34650.0	7
3	256	2000	1415.74	0	608117	48900	-62400	63400.0	6
4	228	1000	1583.91	0	610706	66000	-46000	6500.0	1
...	...	...	...	...	...	...	...	...	
995	3	1000	1310.80	0	431289	0	0	87200.0	17
996	285	1000	1436.79	0	608177	70900	0	108480.0	18
997	130	500	1383.49	0	442797	35100	0	67500.0	7
998	458	2000	1356.92	0	441714	0	0	46980.0	5
999	456	1000	766.19	0	612260	0	0	5060.0	

1000 rows × 12 columns

In [98]:
```
## exploring the categorical variable
```

In [99]:
```
cat_vars = df.select_dtypes(include = ['object'])
cat_vars
```

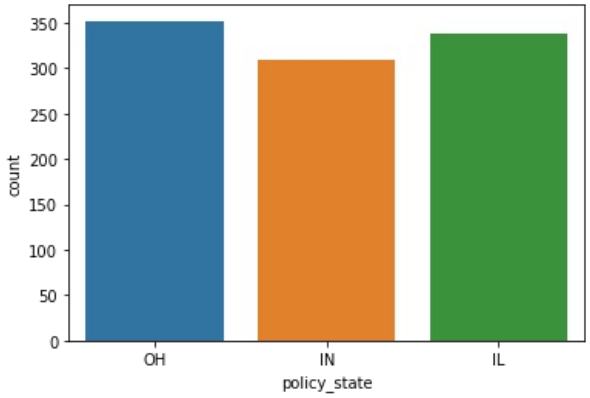
Out[99]:

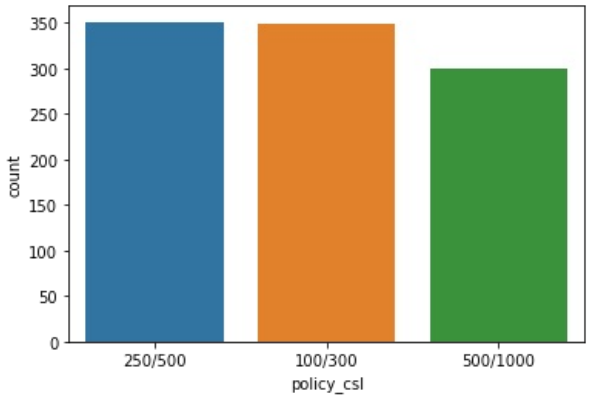
	policy_state	policy_csl	insured_sex	insured_education_level	insured_occupation	insured_hobbies	insured_relationship	incident_date	inc
0	OH	250/500	MALE	MD	craft-repair	sleeping	husband	25-01-2015	Sir
1	IN	250/500	MALE	MD	machine-op-inspct	reading	other-relative	21-01-2015	V
2	OH	100/300	FEMALE	PhD	sales	board-games	own-child	22-02-2015	M
3	IL	250/500	FEMALE	PhD	armed-forces	board-games	unmarried	10-01-2015	Sir
4	IL	500/1000	MALE	Associate	sales	board-games	unmarried	17-02-2015	V
...	...	...	...	...	...	...	...	...	
995	OH	500/1000	FEMALE	Masters	craft-repair	paintball	unmarried	22-02-2015	Sir
996	IL	100/300	FEMALE	PhD	prof-specialty	sleeping	wife	24-01-2015	Sir
997	OH	250/500	FEMALE	Masters	armed-forces	bungie-jumping	other-relative	23-01-2015	M
998	IL	500/1000	MALE	Associate	handlers-cleaners	base-jumping	wife	26-02-2015	Sir
999	OH	250/500	FEMALE	Associate	sales	kayaking	husband	26-02-2015	

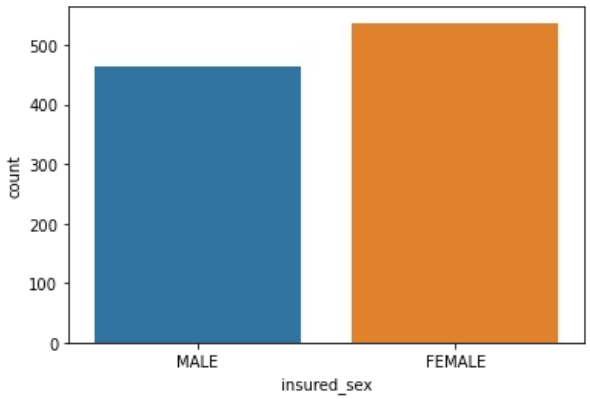
1000 rows × 20 columns

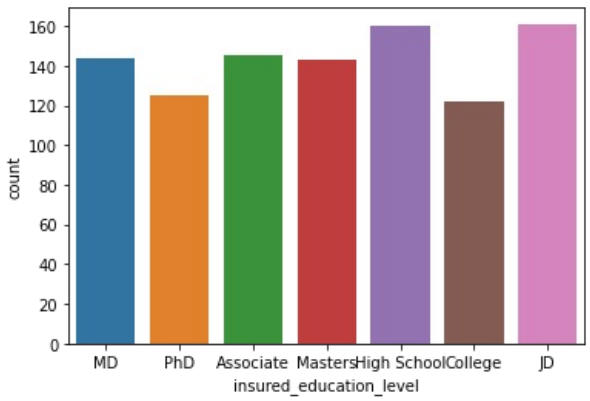
In [100…
```
# Looking at the data distribution for different values.
plt.rcParams['figure.figsize'] = (6, 4)
for i in cat_vars:
```

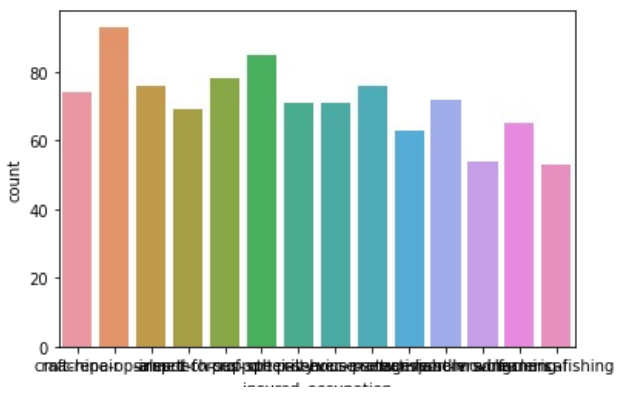
```
    sns.countplot(x = cat_vars[i])
    plt.show()
```



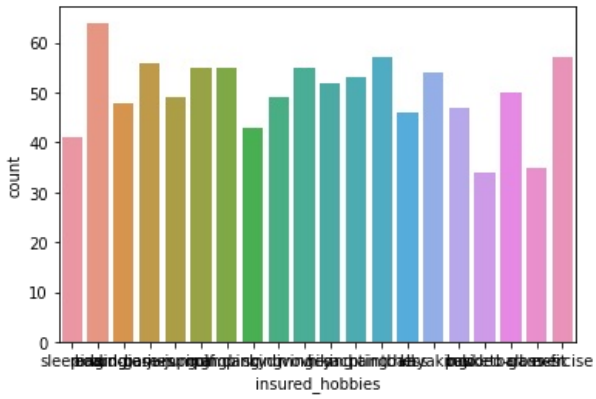


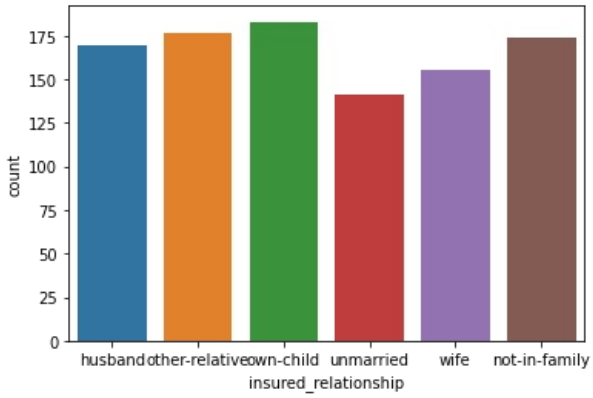


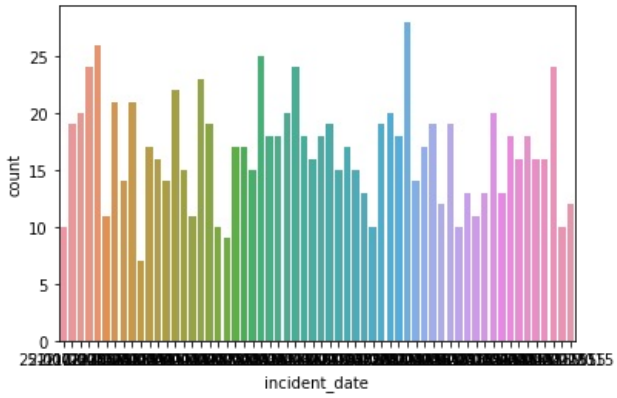


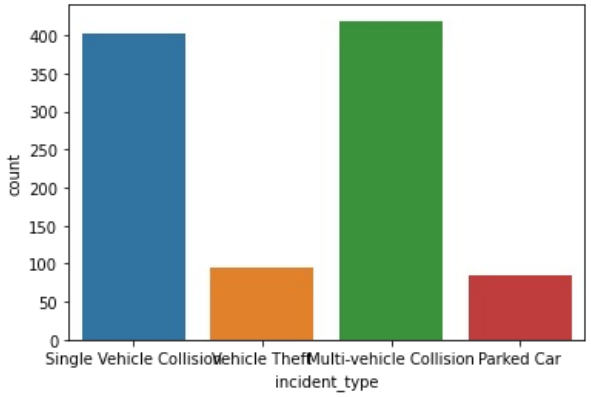


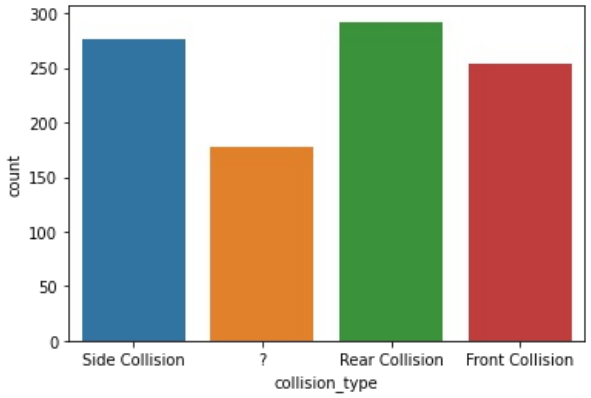
insured_occupation

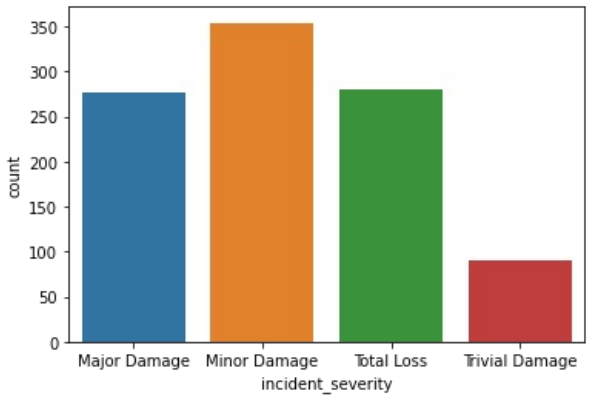


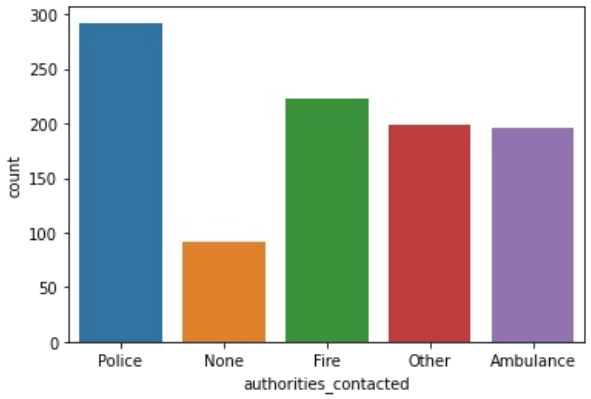


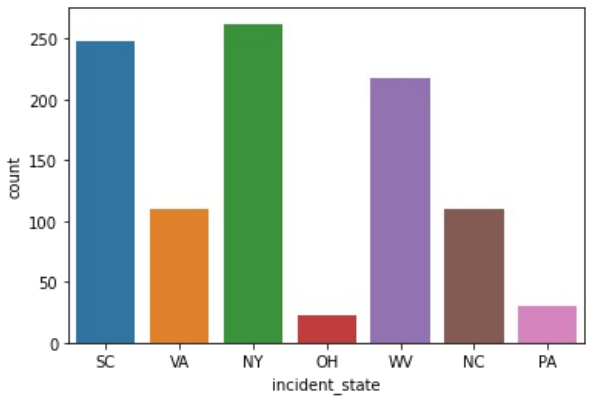


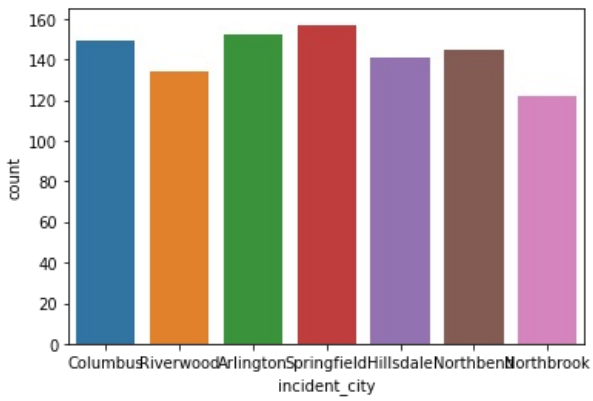


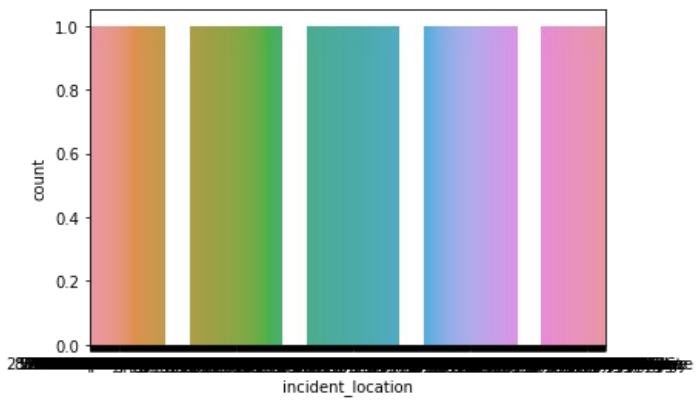


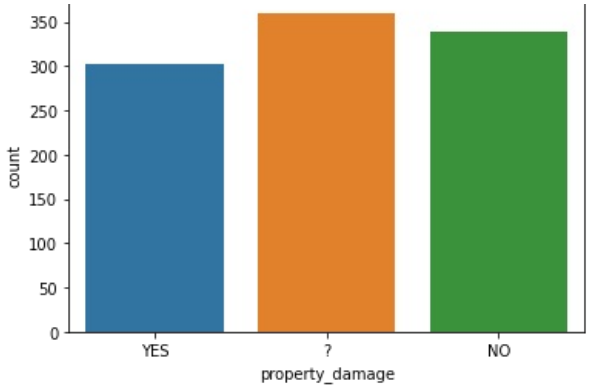

count
incident_severity
Major Damage
Minor Damage
Total Loss
Trivial Damage

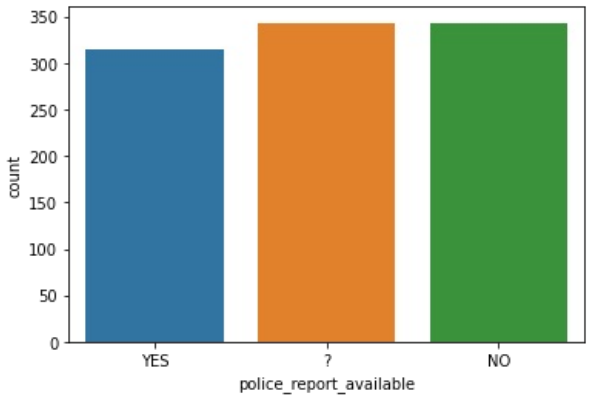

count
authorities_contacted
Police
None
Fire
Other
Ambulance

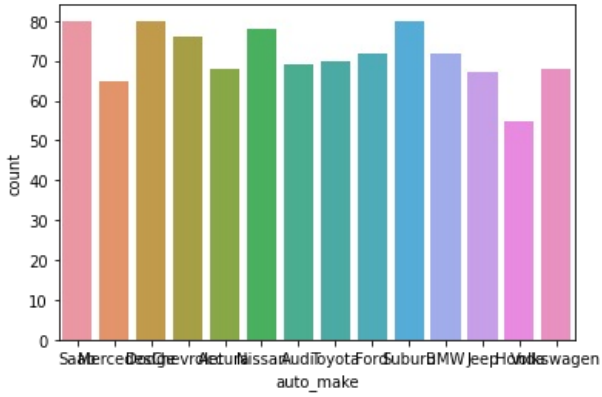

count
incident_state
SC
VA
NY
OH
WV
NC
PA

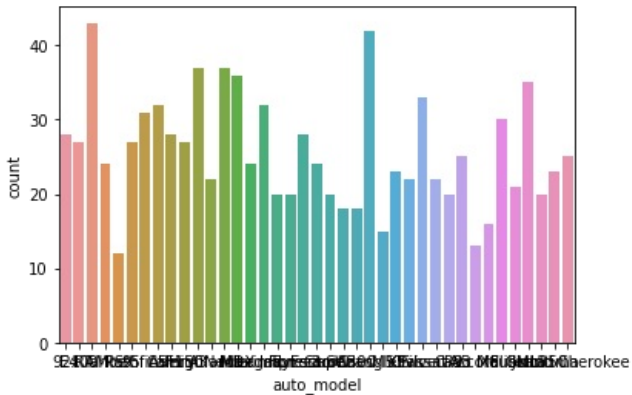

count
incident_city
Columbus
Riverwood
Arlington
Springfield
Hillsdale
Northbend
Northbrook

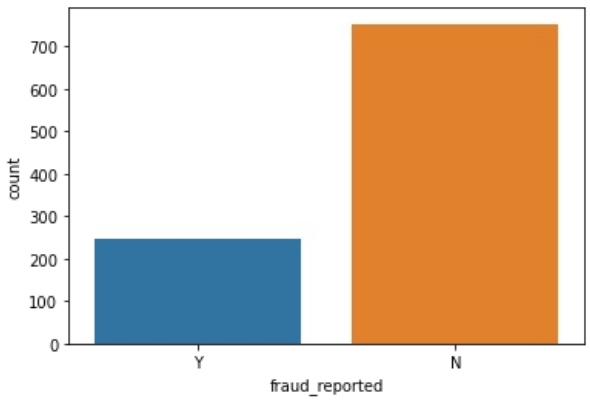

count
incident_location


count
property_damage
YES
?
NO


count
police_report_available
YES
?
NO


count
auto_make


count
auto_model


count
fraud_reported
Y
N

In [101…

```
# Count values of different values for each variables.
for i in cat_vars:
    print(cat_vars[i].value_counts(), end = '\n---------\n\n')
```

```
OH    352
IL    338
IN    310
Name: policy_state, dtype: int64
---------

250/500     351
100/300     349
500/1000    300
Name: policy_csl, dtype: int64
---------

FEMALE    537
MALE      463
Name: insured_sex, dtype: int64
---------

JD             161
High School    160
Associate      145
MD             144
Masters        143
PhD            125
College        122
Name: insured_education_level, dtype: int64
---------

machine-op-inspct    93
prof-specialty       85
tech-support         78
exec-managerial      76
sales                76
craft-repair         74
transport-moving     72
priv-house-serv      71
other-service        71
armed-forces         69
adm-clerical         65
protective-serv      63
handlers-cleaners    54
farming-fishing      53
Name: insured_occupation, dtype: int64
---------

reading           64
paintball         57
exercise          57
bungie-jumping    56
camping           55
golf              55
movies            55
kayaking          54
yachting          53
hiking            52
video-games       50
skydiving         49
base-jumping      49
board-games       48
polo              47
chess             46
dancing           43
sleeping          41
cross-fit         35
basketball        34
Name: insured_hobbies, dtype: int64
---------

own-child         183
other-relative    177
not-in-family     174
husband           170
wife              155
unmarried         141
Name: insured_relationship, dtype: int64
---------

02-02-2015    28
17-02-2015    26
07-01-2015    25
```

```
10-01-2015    24
04-02-2015    24
24-01-2015    24
19-01-2015    23
08-01-2015    22
13-01-2015    21
30-01-2015    21
22-02-2015    20
31-01-2015    20
06-02-2015    20
12-02-2015    20
14-01-2015    19
23-02-2015    19
21-02-2015    19
01-01-2015    19
12-01-2015    19
21-01-2015    19
01-02-2015    18
25-02-2015    18
20-01-2015    18
18-01-2015    18
28-02-2015    18
03-01-2015    18
14-02-2015    18
09-01-2015    17
26-02-2015    17
06-01-2015    17
08-02-2015    17
24-02-2015    17
13-02-2015    16
16-01-2015    16
16-02-2015    16
15-02-2015    16
05-02-2015    16
15-01-2015    15
18-02-2015    15
28-01-2015    15
17-01-2015    15
27-02-2015    14
20-02-2015    14
22-01-2015    14
03-02-2015    13
09-02-2015    13
23-01-2015    13
27-01-2015    13
01-03-2015    12
04-01-2015    12
02-01-2015    11
26-01-2015    11
29-01-2015    11
10-02-2015    10
25-01-2015    10
07-02-2015    10
19-02-2015    10
11-02-2015    10
11-01-2015     9
05-01-2015     7
Name: incident_date, dtype: int64
---------

Multi-vehicle Collision     419
Single Vehicle Collision    403
Vehicle Theft                94
Parked Car                   84
Name: incident_type, dtype: int64
---------

Rear Collision     292
Side Collision     276
Front Collision    254
?                  178
Name: collision_type, dtype: int64
---------

Minor Damage      354
Total Loss        280
Major Damage      276
Trivial Damage     90
Name: incident_severity, dtype: int64
---------

Police       292
Fire         223
```

```
Other        198
Ambulance    196
None          91
Name: authorities_contacted, dtype: int64
---------

NY    262
SC    248
WV    217
VA    110
NC    110
PA     30
OH     23
Name: incident_state, dtype: int64
---------

Springfield    157
Arlington      152
Columbus       149
Northbend      145
Hillsdale      141
Riverwood      134
Northbrook     122
Name: incident_city, dtype: int64
---------

4910 1st Lane       1
3805 Lincoln Hwy    1
6608 MLK Hwy        1
1916 Elm St         1
7571 Elm Ridge      1
                   ..
6484 Tree Drive     1
8906 Elm Lane       1
1956 Apache St      1
8917 Tree Ridge     1
5639 1st Ridge      1
Name: incident_location, Length: 1000, dtype: int64
---------

?      360
NO     338
YES    302
Name: property_damage, dtype: int64
---------

?      343
NO     343
YES    314
Name: police_report_available, dtype: int64
---------

Saab          80
Dodge         80
Suburu        80
Nissan        78
Chevrolet     76
BMW           72
Ford          72
Toyota        70
Audi          69
Accura        68
Volkswagen    68
Jeep          67
Mercedes      65
Honda         55
Name: auto_make, dtype: int64
---------

RAM              43
Wrangler         42
Neon             37
A3               37
MDX              36
Jetta            35
Passat           33
A5               32
Legacy           32
Pathfinder       31
Malibu           30
Camry            28
92x              28
Forrestor        28
```

```
F150              27
E400              27
95                27
Grand Cherokee    25
93                25
Maxima            24
Escape            24
Tahoe             24
Ultima            23
X5                23
Highlander        22
Silverado         22
Civic             22
Fusion            21
TL                20
CRV               20
ML350             20
Impreza           20
Corolla           20
3 Series          18
C300              18
X6                16
M5                15
Accord            13
RSX               12
Name: auto_model, dtype: int64
---------

N    753
Y    247
Name: fraud_reported, dtype: int64
---------
```

In [102…

```
cat_vars=cat_vars.drop(['incident_date'],axis=1)
cat_vars
```

Out[102…

	policy_state	policy_csl	insured_sex	insured_education_level	insured_occupation	insured_hobbies	insured_relationship	incident_type	co
0	OH	250/500	MALE	MD	craft-repair	sleeping	husband	Single Vehicle Collision	S
1	IN	250/500	MALE	MD	machine-op-inspct	reading	other-relative	Vehicle Theft	
2	OH	100/300	FEMALE	PhD	sales	board-games	own-child	Multi-vehicle Collision	R
3	IL	250/500	FEMALE	PhD	armed-forces	board-games	unmarried	Single Vehicle Collision	Fr
4	IL	500/1000	MALE	Associate	sales	board-games	unmarried	Vehicle Theft	
...	...	...	...	...	...	...	...	...	
995	OH	500/1000	FEMALE	Masters	craft-repair	paintball	unmarried	Single Vehicle Collision	Fr
996	IL	100/300	FEMALE	PhD	prof-specialty	sleeping	wife	Single Vehicle Collision	R
997	OH	250/500	FEMALE	Masters	armed-forces	bungie-jumping	other-relative	Multi-vehicle Collision	S
998	IL	500/1000	MALE	Associate	handlers-cleaners	base-jumping	wife	Single Vehicle Collision	R
999	OH	250/500	FEMALE	Associate	sales	kayaking	husband	Parked Car	

1000 rows × 19 columns

In [103…

```
# Count values of different values for each variables.
for i in cat_vars:
    print(cat_vars[i].value_counts(), end = '\n---------\n\n')
```

```
OH    352
IL    338
IN    310
Name: policy_state, dtype: int64
---------

250/500     351
100/300     349
500/1000    300
```

```
Name: policy_csl, dtype: int64
---------

FEMALE    537
MALE      463
Name: insured_sex, dtype: int64
---------

JD             161
High School    160
Associate      145
MD             144
Masters        143
PhD            125
College        122
Name: insured_education_level, dtype: int64
---------

machine-op-inspct    93
prof-specialty       85
tech-support         78
exec-managerial      76
sales                76
craft-repair         74
transport-moving     72
priv-house-serv      71
other-service        71
armed-forces         69
adm-clerical         65
protective-serv      63
handlers-cleaners    54
farming-fishing      53
Name: insured_occupation, dtype: int64
---------

reading           64
paintball         57
exercise          57
bungie-jumping    56
camping           55
golf              55
movies            55
kayaking          54
yachting          53
hiking            52
video-games       50
skydiving         49
base-jumping      49
board-games       48
polo              47
chess             46
dancing           43
sleeping          41
cross-fit         35
basketball        34
Name: insured_hobbies, dtype: int64
---------

own-child         183
other-relative    177
not-in-family     174
husband           170
wife              155
unmarried         141
Name: insured_relationship, dtype: int64
---------

Multi-vehicle Collision     419
Single Vehicle Collision    403
Vehicle Theft                94
Parked Car                   84
Name: incident_type, dtype: int64
---------

Rear Collision     292
Side Collision     276
Front Collision    254
?                  178
Name: collision_type, dtype: int64
---------

Minor Damage      354
Total Loss        280
```

```
Major Damage      276
Trivial Damage     90
Name: incident_severity, dtype: int64
---------

Police       292
Fire         223
Other        198
Ambulance    196
None          91
Name: authorities_contacted, dtype: int64
---------

NY    262
SC    248
WV    217
VA    110
NC    110
PA     30
OH     23
Name: incident_state, dtype: int64
---------

Springfield    157
Arlington      152
Columbus       149
Northbend      145
Hillsdale      141
Riverwood      134
Northbrook     122
Name: incident_city, dtype: int64
---------

4910 1st Lane       1
3805 Lincoln Hwy    1
6608 MLK Hwy        1
1916 Elm St         1
7571 Elm Ridge      1
                   ..
6484 Tree Drive     1
8906 Elm Lane       1
1956 Apache St      1
8917 Tree Ridge     1
5639 1st Ridge      1
Name: incident_location, Length: 1000, dtype: int64
---------

?      360
NO     338
YES    302
Name: property_damage, dtype: int64
---------

?      343
NO     343
YES    314
Name: police_report_available, dtype: int64
---------

Saab          80
Dodge         80
Suburu        80
Nissan        78
Chevrolet     76
BMW           72
Ford          72
Toyota        70
Audi          69
Accura        68
Volkswagen    68
Jeep          67
Mercedes      65
Honda         55
Name: auto_make, dtype: int64
---------

RAM              43
Wrangler         42
Neon             37
A3               37
MDX              36
Jetta            35
Passat           33
```

```
A5                32
Legacy            32
Pathfinder        31
Malibu            30
Camry             28
92x               28
Forrestor         28
F150              27
E400              27
95                27
Grand Cherokee    25
93                25
Maxima            24
Escape            24
Tahoe             24
Ultima            23
X5                23
Highlander        22
Silverado         22
Civic             22
Fusion            21
TL                20
CRV               20
ML350             20
Impreza           20
Corolla           20
3 Series          18
C300              18
X6                16
M5                15
Accord            13
RSX               12
Name: auto_model, dtype: int64
---------

N    753
Y    247
Name: fraud_reported, dtype: int64
---------
```

In [105…
```
##
```

In [107…
```
cat_vars['policy_state'].value_counts()
```

Out[107…
```
OH    352
IL    338
IN    310
Name: policy_state, dtype: int64
```

In [108…
```
## we can use the ordinal encoding
```

In [109…
```
from sklearn.preprocessing import LabelEncoder
```

In [110…
```
lab_enc=LabelEncoder()
```

In [111…
```
df2=lab_enc.fit_transform(cat_vars['policy_state'])
```

In [197…
```
pd.Series(df2)
cat_vars['policy_state']=df2
df2
```

Out[197…
```
array([2, 1, 2, 0, 0, 2, 1, 0, 0, 0, 2, 2, 2, 2, 2, 1, 2, 1, 2, 1, 1, 0,
       0, 1, 1, 2, 1, 1, 2, 0, 0, 0, 0, 0, 0, 2, 0, 0, 1, 1, 0, 1, 0, 2,
       0, 0, 1, 2, 1, 0, 2, 0, 1, 1, 2, 2, 1, 1, 0, 2, 1, 0, 0, 2, 2, 1,
       1, 2, 2, 1, 2, 2, 2, 1, 0, 2, 2, 1, 2, 2, 2, 2, 0, 1, 1, 2, 0, 0,
       0, 1, 2, 0, 0, 1, 0, 0, 1, 0, 2, 0, 1, 0, 0, 2, 2, 0, 2, 1, 0, 1,
       2, 1, 0, 1, 2, 1, 0, 2, 0, 2, 0, 0, 0, 2, 0, 1, 0, 1, 2, 0, 1, 1,
       2, 0, 2, 1, 0, 0, 0, 0, 2, 0, 1, 1, 0, 2, 0, 2, 0, 2, 2, 0, 1, 2,
       2, 2, 2, 1, 1, 2, 1, 1, 2, 2, 0, 1, 2, 0, 0, 1, 0, 2, 0, 1, 0, 1,
       1, 0, 1, 0, 2, 0, 1, 2, 2, 2, 0, 1, 1, 2, 2, 1, 2, 2, 0, 0, 1, 2,
```

```
       0, 2, 0, 2, 1, 0, 0, 2, 2, 2, 1, 1, 2, 1, 2, 2, 1, 2, 0, 2, 0, 2,
       2, 2, 0, 0, 1, 1, 0, 0, 0, 0, 2, 2, 1, 0, 1, 1, 1, 0, 2, 1, 0, 0,
       2, 0, 2, 1, 0, 2, 0, 1, 2, 2, 2, 2, 2, 0, 0, 2, 0, 2, 2, 0, 2, 1,
       0, 0, 0, 1, 2, 1, 2, 0, 0, 2, 0, 1, 1, 2, 0, 0, 2, 1, 2, 2, 2, 0,
       1, 2, 1, 1, 2, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 2, 0, 1,
       2, 0, 2, 1, 2, 2, 0, 0, 2, 1, 0, 1, 0, 1, 0, 2, 0, 0, 0, 1, 1, 1,
       2, 0, 1, 0, 2, 0, 2, 0, 1, 1, 0, 2, 0, 0, 1, 1, 2, 1, 0, 0, 2, 0,
       1, 2, 1, 2, 0, 2, 0, 2, 1, 1, 1, 2, 2, 1, 0, 2, 1, 0, 1, 2, 0, 2,
       0, 0, 2, 2, 1, 2, 1, 0, 2, 0, 0, 2, 2, 1, 2, 2, 0, 1, 2, 2, 1, 2,
       1, 1, 2, 1, 2, 0, 1, 1, 2, 1, 2, 1, 2, 2, 1, 2, 2, 0, 2, 2, 2, 2,
       1, 0, 1, 1, 1, 1, 1, 2, 0, 1, 0, 1, 1, 2, 2, 0, 2, 2, 0, 2, 0, 1,
       1, 2, 1, 1, 1, 1, 1, 0, 2, 0, 2, 1, 0, 0, 0, 0, 1, 0, 2, 1, 2, 0,
       0, 0, 2, 0, 1, 0, 2, 0, 2, 0, 0, 0, 0, 2, 0, 2, 0, 1, 0, 2, 1, 0,
       2, 1, 2, 1, 0, 2, 2, 0, 2, 1, 2, 1, 0, 2, 0, 1, 2, 1, 0, 0, 2, 0,
       0, 0, 1, 0, 2, 2, 0, 0, 0, 2, 1, 0, 0, 0, 1, 0, 1, 2, 2, 0, 0, 1,
       0, 1, 0, 1, 0, 1, 2, 2, 0, 0, 0, 1, 2, 2, 0, 2, 1, 1, 1, 2, 1, 2,
       1, 2, 0, 0, 0, 0, 1, 0, 0, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 0, 2, 2,
       2, 2, 2, 1, 2, 2, 1, 2, 0, 0, 2, 2, 0, 0, 1, 0, 1, 0, 2, 0, 1, 1,
       2, 1, 0, 0, 0, 2, 2, 1, 0, 1, 2, 2, 2, 1, 2, 0, 1, 0, 0, 1, 0, 0,
       1, 0, 2, 0, 2, 2, 2, 0, 2, 1, 0, 1, 1, 0, 1, 2, 2, 2, 0, 2, 2, 0,
       2, 0, 2, 2, 0, 0, 0, 2, 1, 0, 2, 0, 0, 1, 2, 2, 1, 2, 2, 0, 2, 0,
       1, 2, 2, 0, 2, 2, 0, 2, 1, 2, 1, 0, 1, 0, 2, 0, 2, 1, 0, 2, 1, 0,
       2, 2, 1, 2, 1, 0, 1, 2, 0, 0, 1, 2, 2, 2, 2, 2, 1, 2, 0, 2, 0, 0,
       0, 1, 1, 2, 2, 2, 1, 0, 2, 2, 0, 1, 2, 2, 2, 1, 1, 2, 2, 2, 0, 2,
       2, 1, 0, 1, 1, 1, 2, 1, 1, 0, 2, 0, 2, 1, 1, 2, 1, 1, 1, 0, 2, 0,
       0, 0, 2, 0, 0, 1, 1, 0, 0, 2, 1, 2, 2, 0, 0, 1, 1, 2, 0, 0, 0, 0,
       0, 1, 2, 1, 0, 1, 1, 1, 0, 1, 1, 1, 2, 0, 0, 1, 0, 1, 0, 1, 2, 0,
       2, 1, 0, 1, 1, 2, 2, 1, 0, 1, 1, 1, 2, 1, 1, 1, 0, 1, 0, 2, 1, 2,
       0, 1, 0, 1, 0, 1, 0, 1, 2, 2, 1, 0, 2, 2, 2, 2, 0, 0, 1, 1, 2, 1,
       1, 0, 2, 2, 2, 2, 1, 0, 2, 1, 1, 2, 1, 0, 1, 2, 2, 2, 1, 1, 2, 2,
       2, 0, 1, 0, 2, 0, 0, 0, 1, 2, 0, 2, 1, 1, 2, 0, 1, 0, 1, 1, 2, 0,
       1, 2, 0, 2, 0, 0, 0, 1, 2, 0, 1, 0, 0, 2, 1, 0, 1, 0, 2, 2, 2, 1,
       0, 1, 2, 1, 2, 2, 2, 1, 0, 1, 2, 0, 2, 2, 2, 2, 0, 0, 2, 0, 0, 2,
       1, 0, 1, 0, 1, 1, 1, 2, 0, 0, 2, 2, 2, 1, 1, 1, 2, 0, 1, 1, 0, 0,
       0, 0, 2, 2, 0, 1, 1, 1, 1, 2, 1, 0, 2, 2, 0, 1, 2, 2, 1, 0, 2, 2,
       2, 2, 2, 2, 2, 0, 1, 1, 2, 1, 1, 1, 0, 2, 1, 0, 1, 2, 1, 0, 1, 1,
       0, 2, 1, 2, 1, 2, 0, 2, 0, 2])
```

In [116…
```
cat_vars['policy_csl'].value_counts()
```

Out[116…
```
250/500     351
100/300     349
500/1000    300
Name: policy_csl, dtype: int64
```

In [117…
```
from sklearn.preprocessing import LabelEncoder
```

In [118…
```
lab_enc=LabelEncoder()
```

In [119…
```
df3=lab_enc.fit_transform(cat_vars['policy_csl'])
```

In [198…
```
pd.Series(df3)
cat_vars['policy_state']=df3
df3
```

Out[198…
```
array([2, 3, 0, 2, 3, 0, 0, 0, 2, 2, 2, 0, 2, 1, 2, 0, 0, 2, 2, 0, 0, 0,
       0, 2, 2, 0, 1, 3, 2, 2, 0, 0, 0, 0, 2, 2, 2, 1, 0, 2, 2, 2, 0, 2,
       2, 0, 0, 0, 3, 2, 0, 3, 3, 0, 1, 2, 0, 1, 2, 0, 0, 0, 2, 0, 0, 0,
       2, 0, 2, 1, 0, 2, 2, 0, 0, 0, 0, 0, 3, 2, 0, 3, 3, 3, 0, 0, 0, 2,
       1, 0, 0, 2, 1, 0, 0, 3, 2, 0, 1, 3, 2, 2, 0, 1, 0, 3, 0, 0, 2, 0,
       2, 2, 0, 0, 3, 0, 2, 0, 2, 0, 2, 2, 2, 0, 2, 0, 2, 1, 2, 0, 0, 0,
       0, 2, 2, 2, 3, 0, 0, 0, 2, 3, 1, 2, 0, 2, 0, 2, 2, 2, 2, 0, 0, 0,
       0, 2, 0, 1, 0, 3, 1, 2, 0, 2, 0, 0, 2, 0, 1, 1, 0, 2, 0, 0, 3, 2,
       2, 2, 2, 3, 0, 2, 2, 2, 2, 2, 0, 3, 0, 2, 1, 2, 2, 3, 0, 0, 3, 1,
       2, 3, 3, 2, 1, 0, 0, 2, 2, 0, 0, 1, 3, 3, 0, 0, 2, 0, 2, 3, 2, 0,
       2, 2, 2, 0, 0, 2, 0, 2, 0, 2, 2, 2, 0, 0, 0, 0, 2, 0, 2, 2, 2, 2,
       1, 0, 3, 2, 2, 0, 1, 2, 0, 0, 3, 2, 0, 0, 0, 2, 3, 0, 1, 2, 0, 0,
       2, 3, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 2, 0, 0, 0, 3, 3, 2, 0, 2,
       0, 3, 2, 1, 2, 0, 0, 2, 0, 2, 0, 1, 1, 0, 3, 0, 0, 1, 0, 2, 2, 0,
       0, 0, 0, 0, 2, 0, 0, 2, 0, 0, 0, 2, 2, 0, 2, 0, 0, 0, 0, 0, 0, 2,
       2, 0, 1, 1, 2, 2, 0, 2, 2, 0, 0, 2, 2, 3, 2, 2, 2, 2, 0, 2, 2, 0,
       2, 2, 0, 2, 0, 0, 2, 2, 0, 0, 1, 0, 1, 1, 3, 2, 2, 0, 0, 0, 2, 1,
       0, 0, 2, 2, 0, 0, 0, 2, 0, 2, 2, 0, 2, 2, 0, 2, 2, 0, 0, 0, 0, 3,
       2, 2, 3, 2, 0, 2, 0, 2, 2, 0, 0, 0, 2, 3, 1, 1, 0, 0, 3, 2, 2, 2,
```

```
3, 2, 2, 2, 2, 2, 0, 0, 2, 2, 0, 2, 0, 0, 2, 2, 2, 2, 2, 1, 3, 1,
2, 0, 2, 1, 1, 2, 2, 2, 2, 0, 2, 2, 1, 0, 2, 0, 3, 2, 2, 2, 2, 2,
0, 0, 0, 2, 0, 0, 0, 0, 2, 0, 0, 3, 1, 0, 2, 2, 3, 2, 0, 2, 0, 0,
2, 2, 2, 0, 2, 0, 3, 0, 0, 2, 0, 2, 3, 0, 2, 2, 2, 2, 2, 2, 0, 2,
0, 2, 0, 0, 2, 1, 2, 0, 2, 2, 2, 0, 0, 3, 2, 2, 0, 0, 1, 3, 0, 3,
2, 2, 2, 2, 0, 3, 0, 2, 0, 0, 1, 2, 2, 1, 0, 0, 2, 2, 2, 0, 3, 0,
0, 2, 3, 1, 0, 2, 3, 3, 0, 0, 0, 2, 2, 2, 0, 2, 2, 0, 0, 2, 0, 0,
0, 2, 2, 2, 2, 0, 0, 2, 2, 0, 0, 0, 1, 3, 3, 0, 2, 0, 0, 2, 2, 2,
0, 2, 1, 1, 0, 0, 2, 2, 2, 3, 0, 2, 2, 1, 0, 2, 0, 0, 2, 0, 1, 0,
0, 0, 2, 0, 2, 2, 1, 2, 1, 2, 0, 1, 2, 0, 0, 0, 2, 2, 2, 1, 3, 3,
0, 0, 2, 0, 2, 2, 2, 0, 0, 2, 1, 0, 0, 0, 0, 0, 2, 2, 3, 0, 2, 2,
0, 2, 0, 0, 2, 2, 0, 2, 2, 0, 0, 2, 0, 2, 3, 3, 0, 1, 0, 0, 2, 1,
1, 0, 2, 2, 1, 0, 0, 2, 2, 2, 1, 0, 0, 3, 2, 2, 3, 0, 0, 0, 1, 0,
0, 2, 2, 2, 0, 2, 2, 1, 0, 2, 2, 0, 2, 0, 2, 3, 2, 1, 2, 1, 0, 0,
1, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 0, 2, 3, 0, 2, 0, 0, 0, 0, 0, 0,
0, 2, 3, 2, 0, 0, 0, 0, 2, 2, 2, 2, 0, 0, 0, 0, 0, 2, 0, 0, 2, 0,
0, 2, 2, 0, 0, 1, 2, 0, 0, 0, 2, 2, 3, 1, 0, 0, 0, 0, 2, 2, 1, 2,
0, 2, 0, 0, 2, 2, 2, 1, 2, 0, 2, 2, 1, 2, 2, 0, 2, 2, 0, 3, 2, 3,
0, 2, 0, 2, 3, 2, 3, 0, 2, 2, 2, 2, 0, 0, 0, 2, 0, 3, 0, 3, 3,
2, 3, 0, 2, 1, 0, 3, 0, 0, 0, 2, 0, 2, 1, 0, 2, 0, 2, 0, 0, 0, 2,
0, 0, 0, 0, 2, 0, 2, 2, 2, 0, 0, 0, 2, 1, 0, 2, 2, 2, 3, 0, 0, 2,
2, 2, 2, 0, 2, 0, 2, 1, 2, 0, 2, 0, 3, 3, 1, 0, 1, 0, 2, 3, 2, 2,
0, 2, 0, 0, 0, 0, 3, 0, 2, 0, 0, 2, 0, 2, 3, 2, 0, 0, 2, 0, 3, 2,
0, 3, 2, 2, 3, 2, 0, 0, 0, 0, 0, 0, 2, 2, 2, 0, 1, 0, 1, 0, 2, 0,
2, 2, 2, 0, 1, 2, 0, 3, 2, 2, 0, 2, 0, 3, 0, 3, 2, 3, 3, 2, 0, 2,
0, 3, 2, 2, 2, 2, 0, 0, 0, 0, 2, 2, 2, 0, 0, 1, 2, 2, 2, 2, 2, 0,
2, 2, 0, 0, 1, 2, 2, 0, 2, 1])
```

In [121…
```
cat_vars['insured_sex'].value_counts()
```

Out[121…
```
FEMALE    537
MALE      463
Name: insured_sex, dtype: int64
```

In [122…
```
from sklearn.preprocessing import LabelEncoder
```

In [123…
```
lab_enc=LabelEncoder()
```

In [125…
```
df4=lab_enc.fit_transform(cat_vars['insured_sex'])
```

In [199…
```
pd.Series(df4)
cat_vars['policy_state']=df4
df4
```

Out[199…
```
array([1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0,
       1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0,
       1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0,
       0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1,
       1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1,
       1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1,
       1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1,
       0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0,
       1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1,
       1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
       1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0,
       1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0,
       1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1,
       1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1,
       1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1,
       1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0,
       1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1,
       0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1,
```

```
1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1,
1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1,
0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0,
1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0,
0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1,
0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0,
1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1,
0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1,
1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0,
1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0,
0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1,
0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0,
0, 1, 1, 1, 1, 0, 0, 0, 1, 0])
```

In [129…
```
cat_vars['insured_occupation'].value_counts()
```

Out[129…
```
machine-op-inspct    93
prof-specialty       85
tech-support         78
exec-managerial      76
sales                76
craft-repair         74
transport-moving     72
priv-house-serv      71
other-service        71
armed-forces         69
adm-clerical         65
protective-serv      63
handlers-cleaners    54
farming-fishing      53
Name: insured_occupation, dtype: int64
```

In [130…
```
insured_occupation=cat_vars[["insured_occupation"]]
```

In [131…
```
insured_occupation=pd.get_dummies(insured_occupation,drop_first=True)
```

In [132…
```
insured_occupation.head()
```

Out[132…

	insured_occupation_armed-forces	insured_occupation_craft-repair	insured_occupation_exec-managerial	insured_occupation_farming-fishing	insured_occupation_handlers-cleaners
0	0	1	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	1	0	0	0	0
4	0	0	0	0	0

In [137…
```
insured_relationship=cat_vars[["insured_relationship"]]
```

In [138…
```
insured_relationship=pd.get_dummies(insured_relationship,drop_first=True)
```

In [139…
```
insured_relationship.head()
```

Out[139…

	insured_relationship_not-in-family	insured_relationship_other-relative	insured_relationship_own-child	insured_relationship_unmarried	insured_relationship_wife
0	0	0	0	0	0
1	0	1	0	0	0
2	0	0	1	0	0
3	0	0	0	1	0
4	0	0	0	1	0

In [140…
```
incident_type=cat_vars[["incident_type"]]
```

In [141…
```
from sklearn.preprocessing import LabelEncoder
```

In [142…
```
lab_enc=LabelEncoder()
```

In [144…
```
df5=lab_enc.fit_transform(cat_vars['incident_type'])
```

In [200…
```
pd.Series(df5)
cat_vars['incident_type']=df5
df5
```

Out[200…
```
array([3, 0, 2, 1, 0, 2, 1, 1, 1, 2, 1, 1, 2, 0, 2, 3, 2, 3, 3, 3, 2, 3,
       2, 1, 2, 2, 0, 0, 3, 2, 3, 3, 1, 1, 3, 1, 2, 0, 2, 1, 2, 3, 3, 2,
       1, 2, 2, 1, 0, 2, 1, 0, 0, 3, 0, 2, 1, 0, 1, 3, 2, 3, 3, 1, 2, 1,
       3, 3, 1, 0, 3, 1, 2, 2, 3, 1, 1, 1, 0, 2, 3, 0, 0, 0, 3, 1, 1, 3,
       0, 3, 1, 3, 0, 1, 2, 0, 3, 2, 0, 0, 2, 3, 1, 0, 3, 0, 2, 1, 1, 2,
       2, 1, 2, 3, 0, 3, 2, 3, 2, 2, 3, 1, 1, 1, 2, 2, 3, 0, 1, 1, 1, 1,
       1, 3, 3, 2, 0, 2, 2, 1, 2, 0, 0, 2, 2, 1, 3, 1, 2, 3, 3, 2, 2, 2,
       2, 1, 2, 0, 1, 0, 0, 1, 3, 1, 2, 3, 2, 1, 0, 0, 2, 2, 2, 1, 0, 2,
       1, 2, 1, 0, 3, 3, 3, 2, 1, 1, 1, 0, 2, 2, 0, 3, 1, 0, 2, 3, 0, 0,
       1, 0, 0, 2, 0, 2, 2, 2, 2, 3, 3, 0, 0, 0, 3, 3, 3, 3, 1, 0, 2, 3,
       1, 1, 3, 3, 3, 1, 2, 1, 1, 2, 3, 3, 3, 2, 1, 3, 1, 2, 3, 1, 1, 1,
       0, 2, 0, 2, 3, 2, 0, 2, 2, 3, 0, 3, 2, 3, 3, 3, 0, 2, 0, 1, 2, 3,
       3, 0, 3, 2, 2, 1, 2, 0, 1, 2, 1, 1, 3, 2, 3, 3, 3, 0, 0, 2, 3, 1,
       2, 0, 2, 0, 3, 1, 2, 3, 2, 3, 1, 0, 0, 2, 0, 3, 2, 0, 2, 1, 3, 2,
       2, 3, 2, 1, 3, 3, 3, 2, 3, 3, 2, 3, 1, 2, 1, 2, 2, 3, 1, 2, 3, 1,
       3, 1, 0, 0, 3, 2, 3, 2, 1, 1, 3, 2, 2, 0, 2, 3, 2, 1, 3, 1, 3, 2,
       2, 1, 1, 2, 3, 3, 3, 3, 2, 2, 0, 3, 0, 0, 0, 1, 3, 2, 1, 3, 1, 0,
       3, 1, 2, 3, 3, 1, 3, 1, 2, 2, 1, 1, 3, 1, 3, 1, 2, 3, 2, 1, 2, 0,
       3, 1, 0, 2, 1, 1, 2, 3, 1, 3, 2, 3, 2, 0, 0, 0, 1, 1, 0, 1, 2, 2,
       0, 3, 2, 1, 1, 3, 1, 3, 1, 3, 2, 1, 1, 3, 1, 1, 2, 3, 3, 0, 0, 0,
       2, 1, 3, 0, 0, 2, 2, 1, 1, 2, 2, 3, 0, 2, 3, 1, 0, 3, 3, 3, 2, 3,
       1, 3, 1, 2, 3, 3, 1, 3, 1, 1, 3, 0, 0, 3, 2, 3, 0, 1, 2, 2, 1, 2,
       2, 1, 2, 2, 3, 1, 0, 1, 3, 2, 2, 2, 0, 2, 1, 3, 3, 3, 3, 1, 2, 2,
       2, 1, 1, 2, 2, 0, 2, 3, 2, 3, 1, 1, 2, 0, 1, 2, 2, 2, 0, 0, 3, 0,
       1, 1, 1, 1, 1, 0, 2, 3, 1, 3, 0, 1, 1, 0, 2, 2, 1, 1, 3, 3, 0, 2,
       3, 2, 0, 0, 3, 2, 0, 0, 3, 2, 3, 1, 1, 2, 2, 3, 2, 1, 2, 1, 2, 3,
       3, 2, 3, 1, 1, 2, 2, 3, 1, 3, 1, 2, 0, 0, 0, 1, 1, 3, 3, 2, 2, 3,
       3, 2, 0, 0, 1, 1, 3, 1, 2, 0, 3, 2, 3, 0, 1, 1, 2, 1, 2, 1, 0, 3,
       3, 1, 2, 1, 3, 3, 0, 2, 0, 1, 3, 0, 1, 3, 1, 2, 1, 2, 3, 0, 0, 0,
       3, 3, 3, 2, 2, 3, 1, 2, 1, 3, 0, 3, 3, 1, 1, 3, 3, 3, 0, 1, 2, 1,
       1, 3, 1, 3, 3, 1, 3, 2, 2, 2, 2, 1, 2, 2, 0, 0, 1, 0, 1, 3, 1, 0,
       0, 2, 1, 2, 0, 2, 3, 3, 1, 1, 0, 3, 2, 0, 2, 2, 0, 1, 1, 2, 0, 2,
       2, 3, 2, 2, 2, 1, 3, 0, 3, 2, 1, 2, 1, 3, 1, 0, 1, 0, 3, 0, 3, 2,
       0, 2, 3, 1, 3, 3, 1, 2, 3, 1, 2, 1, 1, 0, 2, 2, 3, 1, 2, 2, 3, 3,
       3, 3, 0, 3, 2, 2, 3, 3, 2, 2, 1, 1, 3, 2, 2, 1, 2, 2, 2, 1, 2, 3,
       1, 3, 1, 2, 3, 0, 1, 2, 1, 2, 1, 1, 0, 0, 1, 1, 2, 1, 1, 2, 0, 2,
       2, 3, 1, 3, 2, 3, 3, 0, 2, 2, 1, 2, 0, 2, 3, 2, 3, 2, 3, 0, 1, 0,
       1, 2, 3, 2, 0, 3, 0, 3, 1, 1, 2, 1, 1, 3, 2, 3, 3, 2, 0, 2, 0, 0,
       2, 0, 3, 2, 0, 2, 0, 1, 3, 3, 3, 3, 3, 0, 3, 3, 1, 3, 2, 3, 1,
       3, 3, 2, 3, 2, 1, 1, 2, 1, 2, 1, 2, 1, 0, 3, 2, 3, 3, 0, 1, 2, 1,
       1, 2, 2, 3, 1, 2, 1, 0, 2, 3, 3, 2, 0, 0, 0, 1, 0, 1, 2, 0, 3, 3,
       3, 3, 3, 3, 2, 1, 0, 2, 3, 1, 2, 3, 1, 2, 0, 3, 1, 2, 2, 2, 0, 2,
       2, 0, 1, 1, 0, 3, 2, 3, 3, 3, 2, 3, 1, 1, 1, 2, 0, 3, 0, 1, 3, 1,
       3, 3, 1, 3, 0, 3, 1, 0, 2, 3, 3, 2, 1, 0, 2, 0, 2, 0, 0, 1, 2, 2,
       3, 0, 3, 1, 2, 2, 3, 2, 3, 3, 1, 2, 2, 1, 1, 0, 3, 3, 2, 3, 2, 2,
       2, 2, 1, 3, 0, 1, 2, 3, 2, 0])
```

In [148…
```
df5.shape
```

Out[148…
```
(1000,)
```

In [149…
```
cat_vars["collision_type"].value_counts()
```

Out[149…
```
Rear Collision     292
Side Collision     276
Front Collision    254
?                  178
Name: collision_type, dtype: int64
```

In [150…
```
lab_enc=LabelEncoder()
```

In [156…
```
df6=lab_enc.fit_transform(cat_vars['collision_type'])
```

In [201…
```
pd.Series(df6)
cat_vars['collision_type']=df6
df6
```

Out[201…
```
array([3, 0, 2, 1, 0, 2, 1, 1, 1, 2, 1, 1, 2, 0, 2, 3, 2, 3, 3, 3, 2, 3,
       2, 1, 2, 2, 0, 0, 3, 2, 3, 3, 1, 1, 3, 1, 2, 0, 2, 1, 2, 3, 3, 2,
       1, 2, 2, 1, 0, 2, 1, 0, 0, 3, 0, 2, 1, 0, 1, 3, 2, 3, 3, 1, 2, 1,
       3, 3, 1, 0, 3, 1, 2, 2, 3, 1, 1, 1, 0, 2, 3, 0, 0, 0, 3, 1, 1, 3,
       0, 3, 1, 3, 0, 1, 2, 0, 3, 2, 0, 0, 2, 3, 1, 0, 3, 0, 2, 1, 1, 2,
       2, 1, 2, 3, 0, 3, 2, 3, 2, 2, 3, 1, 1, 1, 2, 2, 3, 0, 1, 1, 1, 1,
       1, 3, 3, 2, 0, 2, 2, 1, 2, 0, 0, 2, 2, 1, 3, 1, 2, 3, 3, 2, 2, 2,
       2, 1, 2, 0, 1, 0, 0, 1, 3, 1, 2, 3, 2, 1, 0, 0, 2, 2, 2, 1, 0, 2,
       1, 2, 1, 0, 3, 3, 3, 2, 1, 1, 1, 0, 2, 2, 0, 3, 1, 0, 2, 3, 0, 0,
       1, 0, 0, 2, 0, 2, 2, 2, 2, 3, 3, 0, 0, 0, 3, 3, 3, 3, 1, 0, 2, 3,
       1, 1, 3, 3, 3, 1, 2, 1, 1, 2, 3, 3, 3, 2, 1, 3, 1, 2, 3, 1, 1, 1,
       0, 2, 0, 2, 3, 2, 0, 2, 2, 3, 0, 3, 2, 3, 3, 3, 0, 2, 0, 1, 2, 3,
       3, 0, 3, 2, 2, 1, 2, 0, 1, 2, 1, 1, 3, 2, 3, 3, 3, 0, 0, 2, 3, 1,
       2, 0, 2, 0, 3, 1, 2, 3, 2, 3, 1, 0, 0, 2, 0, 3, 2, 0, 2, 1, 3, 2,
       2, 3, 2, 1, 3, 3, 3, 2, 3, 3, 2, 3, 1, 2, 1, 2, 2, 3, 1, 2, 3, 1,
       3, 1, 0, 0, 3, 2, 3, 2, 1, 1, 3, 2, 2, 0, 2, 3, 2, 1, 3, 1, 3, 2,
       2, 1, 1, 2, 3, 3, 3, 3, 2, 2, 0, 3, 0, 0, 0, 1, 3, 2, 1, 3, 1, 0,
       3, 1, 2, 3, 3, 1, 3, 1, 2, 2, 1, 1, 3, 1, 3, 1, 2, 3, 2, 1, 2, 0,
       3, 1, 0, 2, 1, 1, 2, 3, 1, 3, 2, 3, 2, 0, 0, 0, 1, 1, 0, 1, 2, 2,
       0, 3, 2, 1, 1, 3, 1, 3, 1, 3, 2, 1, 1, 3, 1, 1, 2, 3, 3, 0, 0, 0,
       2, 1, 3, 0, 0, 2, 2, 1, 1, 2, 2, 3, 0, 2, 3, 1, 0, 3, 3, 3, 2, 3,
       1, 3, 1, 2, 3, 3, 1, 3, 1, 1, 3, 0, 0, 3, 2, 3, 0, 1, 2, 2, 1, 2,
       2, 1, 2, 2, 3, 1, 0, 1, 3, 2, 2, 2, 0, 2, 1, 3, 3, 3, 3, 1, 2, 2,
       2, 1, 1, 2, 2, 0, 2, 3, 2, 3, 1, 1, 2, 0, 1, 2, 2, 2, 0, 0, 3, 0,
       1, 1, 1, 1, 1, 0, 2, 3, 1, 3, 0, 1, 1, 0, 2, 2, 1, 1, 3, 3, 0, 2,
       3, 2, 0, 0, 3, 2, 0, 0, 3, 2, 3, 1, 1, 2, 2, 3, 2, 1, 2, 1, 2, 3,
       3, 2, 3, 1, 1, 2, 2, 3, 1, 3, 1, 2, 0, 0, 0, 1, 1, 3, 3, 2, 2, 3,
       3, 2, 0, 0, 1, 1, 3, 1, 2, 0, 3, 2, 3, 0, 1, 1, 2, 1, 2, 1, 0, 3,
       3, 1, 2, 1, 3, 3, 0, 2, 0, 1, 3, 0, 1, 3, 1, 2, 1, 2, 3, 0, 0, 0,
       3, 3, 3, 2, 2, 3, 1, 2, 1, 3, 0, 3, 3, 1, 1, 3, 3, 3, 0, 1, 2, 1,
       1, 3, 1, 3, 3, 1, 3, 2, 2, 2, 1, 2, 2, 0, 0, 1, 0, 1, 3, 1, 0,
       0, 2, 1, 2, 0, 2, 3, 3, 1, 1, 0, 3, 2, 0, 2, 2, 0, 1, 1, 2, 0, 2,
       2, 3, 2, 2, 2, 1, 3, 0, 3, 2, 1, 2, 1, 3, 1, 0, 1, 0, 3, 0, 3, 2,
       0, 2, 3, 1, 3, 3, 1, 2, 3, 1, 2, 1, 1, 0, 2, 2, 3, 1, 2, 2, 3, 3,
       3, 3, 0, 3, 2, 2, 3, 3, 2, 2, 1, 1, 3, 2, 2, 1, 2, 2, 2, 1, 2, 3,
       1, 3, 1, 2, 3, 0, 1, 2, 1, 2, 1, 1, 0, 0, 1, 1, 2, 1, 1, 2, 0, 2,
       2, 3, 1, 3, 2, 3, 3, 0, 2, 2, 1, 2, 0, 2, 3, 2, 3, 2, 3, 0, 1, 0,
       1, 2, 3, 2, 0, 3, 0, 3, 1, 1, 2, 1, 1, 3, 2, 3, 3, 2, 0, 2, 0, 0,
       2, 0, 3, 2, 0, 2, 0, 1, 3, 3, 3, 3, 3, 0, 3, 3, 3, 1, 3, 2, 3, 1,
       3, 3, 2, 3, 2, 1, 1, 2, 1, 2, 1, 2, 1, 0, 3, 2, 3, 3, 0, 1, 2, 1,
       1, 2, 2, 3, 1, 2, 1, 0, 2, 3, 3, 2, 0, 0, 0, 1, 0, 1, 2, 0, 3, 3,
       3, 3, 3, 3, 2, 1, 0, 2, 3, 1, 2, 3, 1, 2, 0, 3, 1, 2, 2, 2, 0, 2,
       2, 0, 1, 1, 0, 3, 2, 3, 3, 3, 2, 3, 1, 1, 1, 2, 0, 3, 0, 1, 3, 1,
       3, 3, 1, 3, 0, 3, 1, 0, 2, 3, 3, 2, 1, 0, 2, 0, 2, 0, 0, 1, 2, 2,
       3, 0, 3, 1, 2, 2, 3, 2, 3, 3, 1, 2, 2, 1, 1, 0, 3, 3, 2, 3, 2, 2,
       2, 2, 1, 3, 0, 1, 2, 3, 2, 0], dtype=int64)
```

In [158…
```
cat_vars["incident_state"].value_counts()
```

Out[158…
```
NY    262
SC    248
WV    217
VA    110
NC    110
PA     30
OH     23
Name: incident_state, dtype: int64
```

In [159…
```
lab_enc=LabelEncoder()
```

In [162…
```
df7=lab_enc.fit_transform(cat_vars['incident_state'])
```

In [202…
```
pd.Series(df7)
cat_vars['incident_state']=df7
df7
```

Out[202…
```
array([4, 5, 1, 2, 1, 4, 1, 5, 6, 0, 1, 4, 4, 4, 4, 6, 1, 6, 1, 5, 1, 4,
       4, 6, 5, 2, 3, 5, 4, 4, 4, 6, 1, 1, 6, 6, 1, 6, 1, 0, 6, 6, 4, 0,
       4, 1, 0, 6, 1, 6, 1, 1, 6, 0, 5, 1, 4, 5, 6, 5, 1, 2, 6, 6, 5, 1,
       4, 6, 6, 0, 4, 6, 0, 4, 5, 4, 1, 6, 4, 4, 0, 0, 1, 5, 1, 1, 4, 5,
       6, 4, 4, 1, 6, 0, 4, 2, 5, 4, 6, 0, 4, 6, 0, 6, 4, 6, 5, 6, 1, 1,
       4, 0, 6, 5, 5, 1, 4, 1, 1, 1, 4, 1, 5, 0, 6, 1, 6, 3, 1, 4, 1, 6,
       0, 6, 6, 1, 5, 4, 0, 5, 1, 4, 4, 4, 6, 0, 0, 6, 4, 4, 1, 0, 1, 6,
       0, 1, 1, 6, 6, 1, 5, 4, 1, 4, 3, 6, 6, 1, 4, 1, 1, 3, 4, 3, 6, 3,
       2, 4, 6, 1, 5, 4, 5, 4, 1, 1, 0, 1, 6, 5, 4, 4, 4, 5, 5, 0, 6, 6,
       1, 4, 1, 6, 1, 4, 6, 4, 4, 5, 4, 6, 0, 2, 1, 4, 0, 0, 1, 0, 6, 4,
       1, 1, 1, 1, 5, 1, 1, 1, 6, 4, 1, 4, 6, 2, 3, 1, 1, 4, 1, 6, 4, 5,
       4, 1, 3, 1, 1, 4, 4, 0, 4, 4, 6, 2, 4, 1, 4, 4, 1, 4, 4, 0, 1, 4,
       4, 0, 3, 4, 3, 4, 5, 1, 6, 1, 1, 4, 6, 3, 6, 4, 3, 1, 4, 6, 6, 5,
       4, 4, 4, 5, 0, 2, 0, 1, 0, 4, 3, 4, 6, 3, 4, 1, 0, 1, 1, 1, 4, 4,
       1, 1, 4, 4, 4, 6, 4, 6, 6, 5, 1, 1, 1, 4, 6, 3, 5, 1, 1, 1, 1, 6,
       1, 1, 4, 5, 0, 6, 4, 1, 1, 1, 5, 4, 1, 6, 5, 5, 6, 5, 4, 4, 5, 0,
       5, 4, 1, 6, 6, 1, 4, 4, 2, 4, 5, 6, 4, 0, 4, 5, 0, 6, 0, 5, 4, 6,
       4, 1, 6, 5, 1, 5, 4, 4, 1, 0, 1, 1, 6, 1, 1, 6, 6, 1, 1, 6, 2, 4,
       1, 5, 6, 1, 6, 1, 1, 0, 4, 6, 6, 5, 2, 0, 6, 4, 1, 3, 0, 6, 1, 1,
       0, 1, 4, 4, 6, 4, 1, 6, 6, 0, 0, 1, 2, 5, 0, 1, 0, 3, 3, 0, 6, 5,
       1, 6, 4, 5, 0, 4, 1, 0, 4, 6, 1, 4, 1, 1, 1, 6, 1, 0, 0, 1, 4, 6,
       4, 1, 4, 5, 4, 5, 6, 0, 1, 6, 6, 4, 3, 0, 4, 5, 0, 6, 4, 3, 6, 1,
       0, 1, 5, 1, 4, 1, 6, 5, 1, 1, 2, 4, 1, 1, 1, 0, 4, 6, 6, 4, 5, 6,
       1, 1, 1, 1, 6, 1, 6, 3, 1, 1, 6, 4, 4, 1, 1, 6, 4, 4, 0, 4, 5, 4,
       6, 6, 4, 4, 5, 4, 6, 1, 6, 1, 5, 4, 5, 0, 0, 4, 5, 4, 5, 2, 5, 4,
       0, 4, 4, 0, 4, 1, 6, 0, 6, 6, 6, 1, 1, 1, 5, 6, 4, 4, 6, 4, 6, 4,
       1, 1, 2, 4, 1, 4, 6, 1, 0, 0, 4, 1, 6, 2, 6, 4, 1, 6, 3, 6, 4, 5,
       4, 0, 6, 3, 1, 6, 5, 0, 4, 6, 4, 1, 4, 1, 5, 1, 1, 6, 1, 4, 1, 6,
       4, 1, 4, 1, 4, 1, 6, 5, 4, 0, 4, 6, 4, 1, 1, 4, 5, 1, 6, 4, 6, 6,
       0, 5, 1, 1, 4, 0, 1, 4, 1, 1, 1, 6, 1, 1, 4, 1, 6, 6, 4, 0, 0, 4,
       4, 5, 6, 1, 1, 6, 1, 4, 5, 4, 3, 4, 6, 4, 5, 4, 1, 1, 4, 0, 1, 0,
       4, 0, 4, 1, 1, 6, 6, 4, 1, 6, 6, 6, 6, 6, 6, 5, 4, 5, 6, 1, 1, 4,
       1, 1, 4, 1, 4, 4, 0, 4, 3, 0, 4, 0, 1, 6, 1, 3, 4, 1, 6, 2, 6, 5,
       0, 4, 5, 6, 4, 5, 4, 6, 1, 6, 4, 0, 6, 6, 6, 0, 6, 4, 4, 0, 6, 6,
       1, 4, 6, 1, 0, 1, 5, 6, 6, 4, 5, 1, 1, 4, 1, 4, 1, 4, 5, 1, 5, 4,
       6, 5, 1, 6, 6, 6, 6, 0, 6, 0, 6, 0, 1, 1, 6, 1, 1, 4, 2, 5, 4, 1,
       5, 1, 6, 6, 1, 0, 1, 5, 4, 1, 1, 0, 6, 6, 4, 1, 0, 5, 1, 6, 5, 6,
       5, 4, 4, 4, 6, 4, 5, 1, 6, 0, 3, 4, 1, 6, 4, 0, 3, 1, 5, 6, 4, 0,
       5, 0, 4, 4, 4, 4, 6, 1, 5, 4, 5, 1, 1, 0, 1, 4, 1, 6, 6, 4, 6, 1,
       4, 4, 1, 1, 6, 4, 6, 4, 1, 1, 4, 2, 1, 6, 1, 6, 0, 4, 6, 1, 4, 0,
       5, 5, 1, 6, 6, 0, 4, 5, 4, 5, 0, 4, 6, 0, 1, 6, 4, 1, 6, 0, 1, 1,
       4, 6, 5, 5, 4, 6, 6, 4, 3, 6, 0, 6, 1, 4, 1, 1, 1, 1, 6, 1, 6, 4,
       6, 5, 1, 1, 5, 4, 4, 3, 0, 1, 5, 6, 1, 5, 1, 4, 1, 1, 4, 0, 4, 4,
       6, 6, 6, 6, 4, 0, 5, 6, 6, 1, 4, 6, 0, 5, 1, 5, 5, 6, 6, 4, 0, 0,
       1, 4, 4, 5, 1, 4, 4, 5, 6, 6, 1, 1, 2, 1, 1, 4, 6, 6, 6, 0, 1, 4,
       1, 6, 2, 2, 4, 0, 4, 0, 1, 6])
```

In [203…
```
cat_vars['property_damage'].value_counts()
```

Out[203…
```
0    360
1    338
2    302
Name: property_damage, dtype: int64
```

In [166…
```
lab_enc=LabelEncoder()
```

In [167…
```
df8=lab_enc.fit_transform(cat_vars['property_damage'])
```

In [204…
```
pd.Series(df8)
cat_vars['property_damage']=df8
df8
```

Out[204…
```
array([2, 0, 1, 0, 1, 1, 0, 0, 1, 1, 2, 2, 2, 1, 1, 2, 0, 1, 2, 0, 1, 0,
       1, 0, 1, 1, 2, 2, 1, 1, 1, 1, 2, 1, 2, 2, 0, 1, 0, 0, 1, 0, 2, 1,
       2, 1, 2, 1, 0, 1, 0, 0, 0, 2, 1, 2, 0, 1, 2, 1, 1, 2, 1, 2, 1, 2,
       2, 1, 1, 1, 0, 1, 1, 1, 2, 1, 1, 1, 2, 0, 1, 0, 2, 1, 0, 2, 1, 0,
       1, 2, 2, 1, 1, 1, 1, 1, 2, 2, 0, 1, 2, 2, 0, 1, 1, 2, 0, 1, 0, 1,
       0, 2, 0, 0, 2, 1, 1, 0, 1, 0, 1, 1, 0, 2, 2, 1, 1, 0, 2, 0, 2, 0,
       0, 0, 0, 1, 0, 1, 0, 1, 2, 0, 0, 0, 0, 2, 0, 0, 1, 0, 0, 0, 2, 1,
       1, 2, 0, 0, 0, 0, 2, 0, 1, 2, 1, 1, 0, 1, 0, 1, 2, 0, 2, 0, 1, 0,
```

```
1, 2, 2, 0, 0, 2, 2, 0, 0, 2, 0, 2, 0, 1, 2, 0, 1, 2, 0, 2, 0, 2,
2, 0, 1, 0, 1, 2, 1, 1, 0, 0, 1, 1, 1, 0, 2, 0, 2, 0, 1, 0, 0, 1,
2, 1, 2, 0, 0, 2, 2, 0, 2, 0, 1, 1, 0, 0, 0, 1, 2, 1, 0, 2, 0, 2,
0, 0, 2, 2, 2, 0, 0, 0, 2, 1, 2, 0, 1, 1, 2, 2, 2, 0, 0, 1, 0,
1, 1, 0, 1, 2, 2, 0, 1, 2, 0, 1, 0, 2, 1, 0, 2, 1, 1, 2, 0, 0, 2,
2, 0, 2, 1, 2, 0, 0, 2, 0, 0, 2, 0, 2, 1, 1, 2, 2, 0, 2, 0, 0, 1,
0, 1, 1, 0, 0, 0, 2, 1, 1, 2, 0, 2, 2, 2, 0, 1, 2, 2, 2, 2, 0, 1,
0, 0, 1, 0, 0, 1, 1, 2, 0, 0, 2, 2, 0, 0, 0, 2, 2, 0, 2, 0, 2, 0,
1, 0, 2, 1, 2, 2, 0, 1, 2, 2, 2, 1, 1, 0, 0, 1, 0, 2, 2, 0, 1, 2,
0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 1, 1, 1, 0, 2, 2, 2, 0, 1, 0, 1, 2,
1, 2, 0, 1, 2, 2, 2, 0, 1, 0, 0, 0, 0, 1, 0, 2, 1, 0, 2, 1, 0, 2,
1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 2, 2, 0, 2, 0, 0, 1, 0, 0, 1, 2,
1, 1, 2, 1, 1, 2, 2, 2, 1, 0, 2, 1, 2, 0, 0, 2, 0, 0, 2, 2, 1, 2,
1, 1, 2, 0, 1, 0, 0, 1, 0, 2, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 2,
1, 0, 1, 0, 1, 1, 2, 2, 1, 0, 1, 0, 0, 0, 0, 2, 0, 1, 2, 0, 2, 1,
1, 1, 0, 2, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 2, 1, 2, 1, 2, 1, 1,
0, 2, 0, 0, 1, 1, 1, 1, 0, 2, 2, 1, 0, 0, 2, 2, 2, 0, 0, 2, 0, 0,
0, 2, 2, 2, 2, 2, 0, 1, 1, 0, 2, 1, 0, 2, 0, 1, 1, 0, 2, 1, 1, 1,
1, 1, 1, 2, 0, 2, 2, 0, 1, 2, 2, 1, 0, 0, 2, 0, 1, 0, 0, 0, 1, 2,
1, 2, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 2, 1, 1, 1, 0, 0, 1, 0, 0,
0, 1, 0, 1, 0, 0, 0, 0, 2, 2, 2, 0, 1, 0, 0, 1, 1, 2, 2, 0, 2, 1,
0, 0, 0, 1, 0, 1, 0, 2, 1, 1, 0, 0, 1, 2, 1, 1, 1, 2, 0, 0, 0, 0,
1, 2, 2, 1, 0, 1, 0, 2, 1, 1, 0, 1, 2, 0, 1, 1, 2, 2, 0, 1, 0, 2,
1, 1, 2, 1, 1, 0, 2, 1, 2, 0, 2, 1, 0, 1, 0, 0, 2, 2, 1, 1, 2, 0,
2, 2, 2, 2, 1, 1, 1, 1, 0, 1, 2, 2, 1, 1, 2, 1, 2, 0, 2, 1, 1, 0,
1, 2, 0, 0, 1, 0, 2, 0, 1, 1, 1, 0, 0, 2, 0, 1, 1, 2, 0, 1, 2, 1,
2, 1, 1, 1, 2, 0, 2, 0, 0, 2, 1, 0, 0, 2, 0, 0, 2, 0, 2, 2, 2, 0,
1, 0, 0, 2, 1, 2, 1, 0, 2, 0, 0, 1, 1, 1, 1, 2, 2, 1, 2, 0, 2, 2,
2, 2, 1, 0, 0, 2, 1, 2, 2, 0, 0, 2, 2, 1, 2, 1, 0, 0, 1, 1, 1, 0,
1, 0, 0, 0, 2, 2, 0, 0, 0, 1, 2, 0, 0, 1, 1, 2, 1, 2, 2, 1, 2, 0,
1, 1, 2, 1, 1, 0, 1, 0, 1, 0, 1, 2, 2, 1, 2, 1, 1, 1, 1, 0, 0, 1,
0, 1, 1, 2, 2, 0, 2, 0, 1, 1, 1, 2, 0, 1, 2, 0, 2, 1, 2, 0, 2, 2,
1, 0, 0, 2, 1, 2, 1, 1, 2, 1, 2, 0, 0, 2, 0, 0, 1, 1, 1, 2, 2, 2,
1, 2, 2, 1, 1, 0, 2, 0, 1, 0, 2, 0, 2, 2, 0, 1, 2, 2, 1, 1, 1, 2,
1, 0, 0, 2, 2, 2, 2, 1, 0, 2, 1, 0, 0, 0, 0, 0, 0, 0, 1, 2, 2, 1,
0, 1, 1, 0, 1, 0, 1, 1, 2, 2, 2, 1, 1, 1, 1, 0, 1, 1, 0, 0, 2, 2,
1, 0, 2, 1, 0, 2, 1, 2, 0, 0, 0, 1, 2, 0, 0, 0, 2, 0, 2, 0, 2, 2,
0, 1, 2, 0, 0, 2, 2, 0, 0, 0])
```

In [169…
```
cat_vars['police_report_available'].value_counts()
```

Out[169…
```
?      343
NO     343
YES    314
Name: police_report_available, dtype: int64
```

In [170…
```
lab_enc=LabelEncoder()
```

In [171…
```
df9=lab_enc.fit_transform(cat_vars['police_report_available'])
```

In [196…
```
pd.Series(df9)
cat_vars['police_report_available']=df9
df9
```

Out[196…
```
array([1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1,
       1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,
       0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
```

```
1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0,
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1,
0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0,
1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1,
1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0,
1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0]])
```

In [174…
```
cat_vars['auto_make'].value_counts()
```

Out[174…
```
Saab          80
Dodge         80
Suburu        80
Nissan        78
Chevrolet     76
BMW           72
Ford          72
Toyota        70
Audi          69
Accura        68
Volkswagen    68
Jeep          67
Mercedes      65
Honda         55
Name: auto_make, dtype: int64
```

In [175…
```
auto_make=cat_vars['auto_make']
```

In [177…
```
auto_make=pd.get_dummies(auto_make,drop_first=True)
auto_make
```

Out[177…

	Audi	BMW	Chevrolet	Dodge	Ford	Honda	Jeep	Mercedes	Nissan	Saab	Suburu	Toyota	Volkswagen
0	0	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	1	0	0	0	0	0
2	0	0	0	1	0	0	0	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	0	0	0	0	0	1	0	0	0	0	0	0	0
996	0	0	0	0	0	0	0	0	0	0	0	0	1
997	0	0	0	0	0	0	0	0	0	0	1	0	0
998	1	0	0	0	0	0	0	0	0	0	0	0	0
999	0	0	0	0	0	0	0	1	0	0	0	0	0

1000 rows × 13 columns

In [178…
```
auto_model=cat_vars['auto_model']
```

In [179…

```
auto_model=pd.get_dummies(auto_model,drop_first=True)
auto_model
```

Out[179…

	92x	93	95	A3	A5	Accord	C300	CRV	Camry	Civic	...	Pathfinder	RAM	RSX	Silverado	TL	Tahoe	Ultima	Wrangler	X5	X6
0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	1	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	1	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	1	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	0	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
996	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
997	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
998	0	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
999	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

1000 rows × 38 columns

In [180…

```
lab_enc=LabelEncoder()
```

In [183…

```
df_10=lab_enc.fit_transform(cat_vars['fraud_reported'])
```

In [219…

```
pd.Series(df_10)
cat_vars['fraud_reported']=df_10
df_10
```

Out[219…

```
array([1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1,
       1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,
       0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1,
       0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

In [185…
```
## concatenate dataframe
```

In [208…
```
cat_vars_new=pd.concat([insured_occupation,insured_relationship,auto_make,auto_model],axis=1)
cat_vars_new.head()
```

Out[208…

	insured_occupation_armed-forces	insured_occupation_craft-repair	insured_occupation_exec-managerial	insured_occupation_farming-fishing	insured_occupation_handlers-cleaners
0	0	1	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	1	0	0	0	0
4	0	0	0	0	0

5 rows × 69 columns

In [212…
```
# Combining Numerical and Categorical data.
final_data = pd.concat([cont_data, cat_vars_new], axis = 1)
final_data
```

Out[212…

	months_as_customer	age	policy_deductable	policy_annual_premium	umbrella_limit	insured_zip	capital-gains	capital-loss	total_claim_amount	inj
0	328	48	1000	1406.91	0	466132	53300	0	71610.0	
1	228	42	2000	1197.22	0	468176	0	0	5070.0	
2	134	29	2000	1413.14	0	430632	35100	0	34650.0	
3	256	41	2000	1415.74	0	608117	48900	-62400	63400.0	
4	228	44	1000	1583.91	0	610706	66000	-46000	6500.0	
...	...	...	...	...	...	...	...	...	...	
995	3	38	1000	1310.80	0	431289	0	0	87200.0	
996	285	41	1000	1436.79	0	608177	70900	0	108480.0	
997	130	34	500	1383.49	0	442797	35100	0	67500.0	
998	458	62	2000	1356.92	0	441714	0	0	46980.0	
999	456	60	1000	766.19	0	612260	0	0	5060.0	

1000 rows × 82 columns

In [215…
```
x=final_data
```

In [216…
```
x
```

Out[216…

	months_as_customer	age	policy_deductable	policy_annual_premium	umbrella_limit	insured_zip	capital-gains	capital-loss	total_claim_amount	inj
0	328	48	1000	1406.91	0	466132	53300	0	71610.0	
1	228	42	2000	1197.22	0	468176	0	0	5070.0	
2	134	29	2000	1413.14	0	430632	35100	0	34650.0	
3	256	41	2000	1415.74	0	608117	48900	-62400	63400.0	
4	228	44	1000	1583.91	0	610706	66000	-46000	6500.0	
...	...	...	...	...	...	...	...	...	...	
995	3	38	1000	1310.80	0	431289	0	0	87200.0	
996	285	41	1000	1436.79	0	608177	70900	0	108480.0	
997	130	34	500	1383.49	0	442797	35100	0	67500.0	
998	458	62	2000	1356.92	0	441714	0	0	46980.0	
999	456	60	1000	766.19	0	612260	0	0	5060.0	

1000 rows × 82 columns

In [217…
```
y=
```

In [218…
```
y
```

Out[218…
```
0      Y
1      Y
2      N
3      Y
4      N
      ..
995    N
996    N
997    N
998    N
999    N
Name: fraud_reported, Length: 1000, dtype: object
```

In [220…
```
y=df_10
```

In [221…
```
y
```

Out[221…
```
array([1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1,
       1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,
       0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1,
       0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

In [222…
```
y.shape
```

Out[222…
```
(1000,)
```

In [223…
```
from sklearn.preprocessing import StandardScaler
```

In [224…
```
st=StandardScaler()
```

In [225…
```
st.fit_transform(x)
```

Out[225…
```
array([[ 1.07813958,  0.99320035, -0.22238259, ..., -0.20938323,
        -0.15343224, -0.12751534],
       [ 0.2089946 ,  0.33530654,  1.41278352, ..., -0.20938323,
        -0.15343224, -0.12751534],
       [-0.60800168, -1.09013003,  1.41278352, ..., -0.20938323,
        -0.15343224, -0.12751534],
       ...,
       [-0.64276748, -0.5418852 , -1.03996564, ..., -0.20938323,
        -0.15343224, -0.12751534],
       [ 2.20802805,  2.52828589,  1.41278352, ..., -0.20938323,
        -0.15343224, -0.12751534],
       [ 2.19064515,  2.30898795, -0.22238259, ..., -0.20938323,
        -0.15343224, -0.12751534]])
```

In [226…
```
from sklearn.model_selection import train_test_split,cross_val_score
#importing models
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier,GradientBoostingClassifier
```

In [228…
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=41)
```

In [229…
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

In [241…
```
kn=KNeighborsClassifier()
```

In [242…
```
kn.fit(x_train,y_train)
```

Out[242…
```
KNeighborsClassifier()
```

In [243…
```
y_pred=kn.predict(x_test)
```

In [244…
```
y_pred
```

Out[244…
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

In [234…
```
accuracy_score(y_pred,y_test)
```

Out[234…
```
0.6366666666666667
```

In [245…
```
confusion_matrix(y_test,y_pred)
```

Out[245… array([[185, 15],
       [94,   6]], dtype=int64)

In [246…
```
classification_report(y_test,y_pred)
```

Out[246…
```
'              precision    recall  f1-score   support\n\n           0       0.66      0.93      0.77       200\n           1       0.29      0.06      0.10       100\n\n    accuracy                           0.64       300\n   macro avg       0.47      0.49      0.44       300\nweighted avg       0.54      0.64      0.55       300\n'
```

In [247…
```
sv=SVC()
```

In [248…
```
sv.fit(x_train,y_train)
```

Out[248… SVC()

In [249…
```
y_pred=sv.predict(x_test)
```

In [250…
```
accuracy_score(y_test,y_pred)
```

Out[250… 0.6666666666666666

In [251…
```
confusion_matrix(y_test,y_pred)
```

Out[251… array([[200,   0],
       [100,   0]], dtype=int64)

In [252…
```
classification_report(y_test,y_pred)
```

```
C:\Users\Rakesh Lodem\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Rakesh Lodem\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Rakesh Lodem\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Out[252…
```
'              precision    recall  f1-score   support\n\n           0       0.67      1.00      0.80       200\n           1       0.00      0.00      0.00       100\n\n    accuracy                           0.67       300\n   macro avg       0.33      0.50      0.40       300\nweighted avg       0.44      0.67      0.53       300\n'
```

In [253…
```
## gradient Boosting classifier
```

In [254…
```
gb=GradientBoostingClassifier()
```

In [255…
```
gb.fit(x_train,y_train)
```

Out[255… GradientBoostingClassifier()

In [256…
```
y_pred=gb.predict(x_test)
```

In [257…
```
accuracy_score(y_test,y_pred)
```

Out[257…
```
0.6466666666666666
```

In [258…
```
confusion_matrix(y_test,y_pred)
```

Out[258…
```
array([[189,  11],
       [ 95,   5]], dtype=int64)
```

In [259…
```
classification_report(y_test,y_pred)
```

Out[259…
```
'              precision    recall  f1-score   support\n\n           0       0.67      0.94      0.78       200\n
1       0.31      0.05      0.09       100\n\n    accuracy                           0.65       300\n   macro avg
0.49      0.50      0.43       300\nweighted avg       0.55      0.65      0.55       300\n'
```

In [260…
```
## randomforest classifier
```

In [261…
```
rf=RandomForestClassifier()
```

In [262…
```
rf.fit(x_train,y_train)
```

Out[262…
```
RandomForestClassifier()
```

In [263…
```
y_pred=rf.predict(x_test)
```

In [264…
```
accuracy_score(y_test,y_pred)
```

Out[264…
```
0.6666666666666666
```

In [265…
```
confusion_matrix(y_test,y_pred)
```

Out[265…
```
array([[199,   1],
       [ 99,   1]], dtype=int64)
```

In [266…
```
classification_report(y_test,y_pred)
```

Out[266…
```
'              precision    recall  f1-score   support\n\n           0       0.67      0.99      0.80       200\n
1       0.50      0.01      0.02       100\n\n    accuracy                           0.67       300\n   macro avg
0.58      0.50      0.41       300\nweighted avg       0.61      0.67      0.54       300\n'
```

In [268…
```
## decision tree classifier
```

In [269…
```
dt=DecisionTreeClassifier()
```

In [270…
```
dt.fit(x_train,y_train)
```

Out[270…
```
DecisionTreeClassifier()
```

In [271…
```
y_pred=dt.predict(x_test)
```

In [272…
```
accuracy_score(y_test,y_pred)
```

Out[272…
```
0.58
```

In [273…
```
confusion_matrix(y_test,y_pred)
```

Out[273…
```
array([[150,  50],
       [ 76,  24]], dtype=int64)
```

In [275…
```
classification_report(y_test,y_pred)
```

Out[275…
```
'              precision    recall  f1-score   support\n\n           0       0.66      0.75      0.70       200\n
1       0.32      0.24      0.28       100\n\n    accuracy                           0.58       300\n   macro avg
0.49      0.49      0.49       300\nweighted avg       0.55      0.58      0.56       300\n'
```

In [276…
```
## ada boost classifier
```

In [277…
```
ad=AdaBoostClassifier()
```

In [278…
```
ad.fit(x_train,y_train)
```

Out[278…
```
AdaBoostClassifier()
```

In [280…
```
y_pred=ad.predict(x_test)
```

In [282…
```
accuracy_score(y_test,y_pred)
```

Out[282…
```
0.6733333333333333
```

In [283…
```
confusion_matrix(y_test,y_pred)
```

Out[283…
```
array([[191,   9],
       [ 89,  11]], dtype=int64)
```

In [284…
```
classification_report(y_test,y_pred)
```

Out[284…
```
'              precision    recall  f1-score   support\n\n           0       0.68      0.95      0.80       200\n
1       0.55      0.11      0.18       100\n\n    accuracy                           0.67       300\n   macro avg
0.62      0.53      0.49       300\nweighted avg       0.64      0.67      0.59       300\n'
```

In [285…
```
## svc and ada boost working well
```

In [286…
```
## hyperparameter tuning
```

In [287…
```
from sklearn.model_selection import RandomizedSearchCV
```

In [289…
```
params={'C':[0.001,1,2,3,4,5,6,7,8,9,10],'gamma':[0.1,0.2,0.3,0.4,0.5,0.6]}
```

In [291…
```
r=RandomizedSearchCV(SVC(),params)
```

In [292…
```
r.fit(x_train,y_train)
```

Out[292…

```
RandomizedSearchCV(estimator=SVC(),
                   param_distributions={'C': [0.001, 1, 2, 3, 4, 5, 6, 7, 8, 9,
                                              10],
                                        'gamma': [0.1, 0.2, 0.3, 0.4, 0.5,
                                                  0.6]})
```

In [293…

```
r.best_params_
```

Out[293…

```
{'gamma': 0.6, 'C': 6}
```

In [294…

```
svc=SVC(C=6,gamma=0.6)
```

In [295…

```
svc.fit(x_train,y_train)
```

Out[295…

```
SVC(C=6, gamma=0.6)
```

In [296…

```
y_pred=svc.predict(x_test)
```

In [297…

```
accuracy_score(y_test,y_pred)
```

Out[297…

```
0.6666666666666666
```

In [298…

```
classification_report(y_test,y_pred)
```

```
C:\Users\Rakesh Lodem\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning
: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_divis
ion` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Rakesh Lodem\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning
: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_divis
ion` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Rakesh Lodem\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning
: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_divis
ion` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Out[298…

```
'              precision    recall  f1-score   support\n\n           0       0.67      1.00      0.80       200\n
1       0.00      0.00      0.00       100\n\n    accuracy                           0.67       300\n   macro avg
0.33      0.50      0.40       300\nweighted avg       0.44      0.67      0.53       300\n'
```

In [299…

```
confusion_matrix(y_test,y_pred)
```

Out[299…

```
array([[200,   0],
       [100,   0]], dtype=int64)
```

In [301…

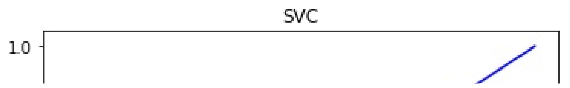
```
## finalizing the SVC model
```

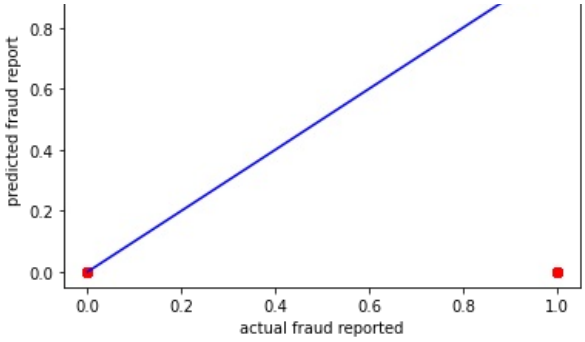
In [303…

```
plt.scatter(x=y_test,y=y_pred,color='r')
plt.plot(y_test,y_test,color='b')
plt.xlabel('actual fraud reported')
plt.ylabel('predicted fraud report')
plt.title('SVC')
```

Out[303…

```
Text(0.5, 1.0, 'SVC')
```





In [304…
```
## evaluting the model
```

In [305…
```
import numpy as np
```

In [306…
```
a=np.array(y_test)
```

In [307…
```
predicted=np.array(sv.predict(x_test))
```

In [310…
```
df_com=pd.DataFrame({'true':a,'predicted':predicted},index=range(len(a)))
df_com.head()
```

Out[310…

	true	predicted
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0

In [311…
```
## saving the model
```

In [312…
```
import pickle
```

In [313…
```
filename='INSURANCE_CLAIM_FRAUD_DETECTION'
```

In [314…
```
pickle.dump(svc,open(filename,'wb'))
```

In [ ]:
```

```