Auto Insurance Fraud

Problem Definition

Aim of the project

The aim of the project is to find the Auto Insurance Fraud Reported. Insurance fraud is a huge problem in the industry. It's difficult to identify fraud claims. Machine Learning is in a unique position to help the Auto Insurance industry with this problem. As per industry specific identifying a fraudulent report of auto insurance claim is a very high task from its customer data.

Why Auto Insurance Fraud Detection: Insurance fraud occurs when a customer attempts to earn profit by exploiting the terms and conditions of the insurance agreement. The Auto Insurance fraud takes place when the vehicle collision occurs or vehicle theft. These days investigating a single case being very time consuming also it becomes very complex to identifying a whole number of fraudulent. **To identifying fraudulent through dataset will be very grateful to the industry specific**.

Model Aims: The aim of the model to detect the Automobile Insurance customer's fraud reported or not. The model built to predict the fraud reported will be aiming to obtain the highest **Accuracy score**.

Accuracy simply measures how often the classifier correctly predicts. We can define accuracy as the ratio of the number of correct predictions and the total number of predictions.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

The model will also aim towards obtaining the **highest ROC AUC score, Precision score and Recall**.

Precision for a label is defined as the number of true positives divided by the number of predicted positives.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

Recall (Sensitivity) for a label is defined as the number of true positives divided by the total number of actual positives.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

AUC-ROC — The Receiver Operator Characteristic (ROC) is a probability curve that plots the TPR(True Positive Rate) against the FPR(False Positive Rate) at various threshold values and separates the 'signal' from the 'noise'.

Overview of the dataset

The dataset is obtained from: https://github.com/dsrscientist/Data-Science-ML-Capstone-Projects/blob/master/Automobile_insurance_fraud.csv

- Dataset having 10000 rows and 40 columns.
- Dataset having 19 numerical and 21 object type columns.

Features of the dataset

'months_as_customer'	Number of months of the customer retain
'age'	Age of the customer
'policy_number'	Identity of the customer
'policy_bind_date'	Date in policy coverage get into place
'policy_state'	Policy State location details
'policy_deductable'	Deductible amount of the policy
'policy_annual_premium'	Annual premium amount of the policy
'umbrella_limit'	Umbrella limit amount of the policy
'insured_zip'	Zip code of the customers
'insured_sex'	Gender
'insured_education_level'	Education level of the customers
'insured_occupation'	Occupation of the customers
'insured_hobbies'	Hobbies of the customers
'insured_relationship'	Relationship of the customers
'capital-gains'	Capital gain of the customers
'capital-loss'	Capital loss of the customers
'incident_date'	Incident date of the accident
'incident_type'	Incident type of the accident
'collision_type'	Collision type of the accident
'incident_severity'	Severity of the accident auto mobile
'authorities_contacted'	Authorities contacted when incident occurred
'incident_state'	Incident state of the customers
'incident_city'	Incident city of the customers
'incident_location'	Incident location of the customers
'incident_hour_of_the_day'	Hour when the incident occurred
'number_of_vehicles_involved'	The number of vehicles involved in the accident
'property_damage'	Property damage during the incident
'bodily_injuries'	Bodily injuries occurred during the incident
'witnesses'	Witnesses during the incident
'police_report_available'	Police report
'total_claim_amount'	The total amount to be claimed by the customers
'injury_claim'	The injury claim to be claimed by the customers
'property_claim'	The property claim to be claimed by the customers
'vehicle_claim'	The vehicle claim to be claimed by the customers
'auto_make'	Automobile company of the customers
'auto_model'	Automobile models of the customers
'auto_year'	Automobile purchase year

'fraud_reported'	Fraud reported by the customers
'policy_csl'	Predetermined limit for the combined total of the Bodily Injury Liability coverage and Property Damage Liability coverage per occurrence or accident

Model Building

Data Analysis

Used Libraries:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

Importing Dataset

```
df = pd.read_csv("https://raw.githubusercontent.com/arjun0200/Datasets/main/Automobile_insurance_fraud.csv")
pd.set_option("max_rows", None)
pd.set_option("max_columns", None)
```

	months_as_customer	age	policy_number	policy_bind_date	policy_state	policy_csl	policy_deductable	policy_annual_premium	umbrella_limit	insured_zip
0	328	48	521585	17-10-2014	OH	250/500	1000	1406.91	0	466132
1	228	42	342868	27-06-2006	IN	250/500	2000	1197.22	5000000	468176
2	134	29	687698	06-09-2000	OH	100/300	2000	1413.14	5000000	430632
3	256	41	227811	25-05-1990	IL	250/500	2000	1415.74	6000000	608117
4	228	44	367455	06-06-2014	IL	500/1000	1000	1583.91	6000000	610706

insured_sex	insured_education_level	insured_occupation	insured_hobbies	insured_relationship	capital-gains	capital-loss	incident_date	incident_type	collision_type
MALE	MD	craft-repair	sleeping	husband	53300	0	25-01-2015	Single Vehicle Collision	Side Collision
MALE	MD	machine-op-inspct	reading	other-relative	0	0	21-01-2015	Vehicle Theft	?
FEMALE	PhD	sales	board-games	own-child	35100	0	22-02-2015	Multi-vehicle Collision	Rear Collision
FEMALE	PhD	armed-forces	board-games	unmarried	48900	-62400	10-01-2015	Single Vehicle Collision	Front Collision
MALE	Associate	sales	board-games	unmarried	66000	-46000	17-02-2015	Vehicle Theft	?

incident_severity	authorities_contacted	incident_state	incident_city	incident_location	incident_hour_of_the_day	number_of_vehicles_involved	property_damage
Major Damage	Police	SC	Columbus	9935 4th Drive	5	1	YES
Minor Damage	Police	VA	Riverwood	6608 MLK Hwy	8	1	?
Minor Damage	Police	NY	Columbus	7121 Francis Lane	7	3	NO
Major Damage	Police	OH	Arlington	6956 Maple Drive	5	1	?
Minor Damage	None	NY	Arlington	3041 3rd Ave	20	1	NO

bodily_injuries	witnesses	police_report_available	total_claim_amount	injury_claim	property_claim	vehicle_claim	auto_make	auto_model	auto_year	fraud_rep
1	2	YES	71610	6510	13020	52080	Saab	92x	2004	
0	0	?	5070	780	780	3510	Mercedes	E400	2007	
2	3	NO	34650	7700	3850	23100	Dodge	RAM	2007	
1	2	NO	63400	6340	6340	50720	Chevrolet	Tahoe	2014	
0	1	NO	6500	1300	650	4550	Accura	RSX	2009	

fraud_reported	_c39
Y	NaN
Y	NaN
N	NaN
Y	NaN
N	NaN

Checking data types of the dataset

```
df.dtypes  # checking datatypes of the dataset
```

```
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
```

Observations:

- From above code we find that 'months_as_customer', 'age', 'policy_number', 'policy_deductable', 'policy_annual_premium', 'umbrella_limit', 'insured_zip', 'capital-gains', 'capital-loss', 'incident_hour_of_the_day', 'number_of_vehicles_involved', 'bodily_injuries', 'witnesses', 'total_claim_amount', 'injury_claim', 'property_claim', 'vehicle_claim', 'auto_year', '_c39' these columns are of numerical types.

- And these are object tpyes 'policy_bind_date', 'policy_state', 'policy_csl', 'insured_sex', 'insured_education_level', 'insured_occupation', 'insured_hobbies', 'insured_relationship', 'incident_date', 'incident_type', 'collision_type', 'incident_severity', 'authorities_contacted', 'incident_state', 'incident_city', 'incident_location', 'property_damage', 'police_report_available', 'auto_make', 'auto_model', 'fraud_reported'.

Summary of the dataset:

```
df.describe()  # checking overall summary of the dataset
```

	months_as_customer	age	policy_number	policy_deductable	policy_annual_premium	umbrella_limit	insured_zip	capital-gains	capital-lo
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1.000000e+03	1000.000000	1000.000000	1000.0000
mean	203.954000	38.948000	546238.648000	1136.000000	1256.406150	1.101000e+06	501214.488000	25126.100000	-26793.7000
std	115.113174	9.140287	257063.005276	611.864673	244.167395	2.297407e+06	71701.610941	27872.187708	28104.0966
min	0.000000	19.000000	100804.000000	500.000000	433.330000	-1.000000e+06	430104.000000	0.000000	-111100.0000
25%	115.750000	32.000000	335980.250000	500.000000	1089.607500	0.000000e+00	448404.500000	0.000000	-51500.0000
50%	199.500000	38.000000	533135.000000	1000.000000	1257.200000	0.000000e+00	466445.500000	0.000000	-23250.0000
75%	276.250000	44.000000	759099.750000	2000.000000	1415.695000	0.000000e+00	603251.000000	51025.000000	0.0000
max	479.000000	64.000000	999435.000000	2000.000000	2047.590000	1.000000e+07	620962.000000	100500.000000	0.0000

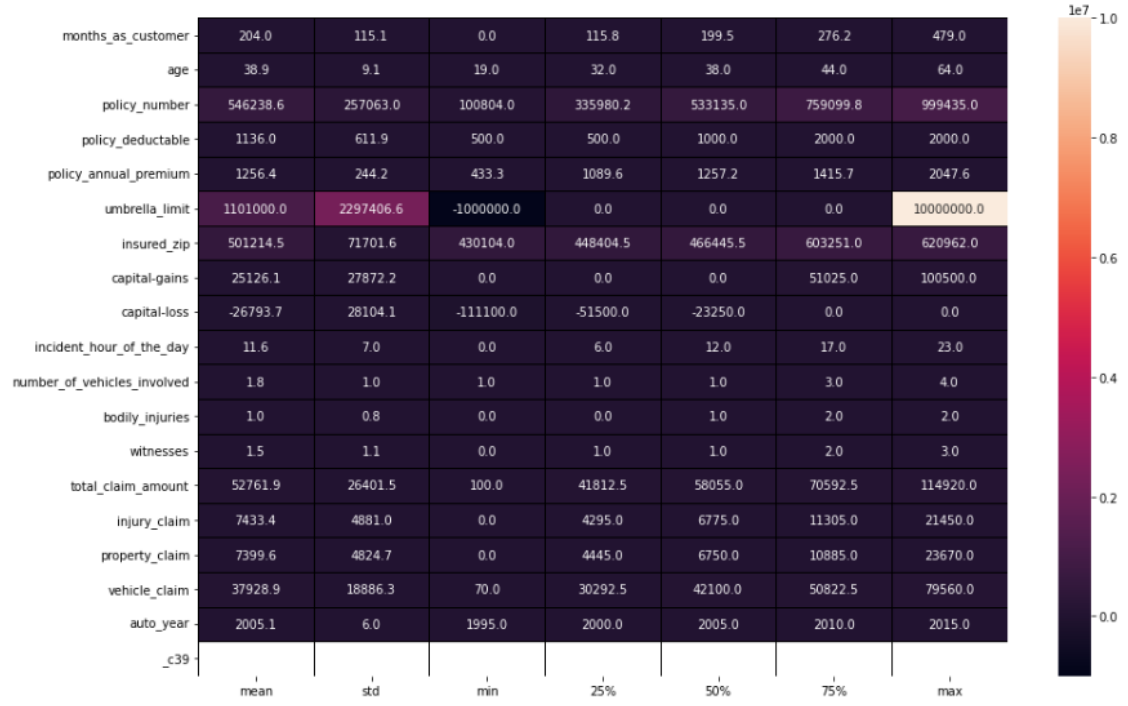
capital-loss	incident_hour_of_the_day	number_of_vehicles_involved	bodily_injuries	witnesses	total_claim_amount	injury_claim	property_claim	vehicle_c
1000.000000	1000.000000	1000.00000	1000.000000	1000.000000	1000.00000	1000.000000	1000.000000	1000.00
-26793.700000	11.644000	1.83900	0.992000	1.487000	52761.94000	7433.420000	7399.570000	37928.95
28104.096686	6.951373	1.01888	0.820127	1.111335	26401.53319	4880.951853	4824.726179	18886.25
-111100.000000	0.000000	1.00000	0.000000	0.000000	100.00000	0.000000	0.000000	70.00
-51500.000000	6.000000	1.00000	0.000000	1.000000	41812.50000	4295.000000	4445.000000	30292.50
-23250.000000	12.000000	1.00000	1.000000	1.000000	58055.00000	6775.000000	6750.000000	42100.00
0.000000	17.000000	3.00000	2.000000	2.000000	70592.50000	11305.000000	10885.000000	50822.50
0.000000	23.000000	4.00000	2.000000	3.000000	114920.00000	21450.000000	23670.000000	79560.00

vehicle_claim	auto_year	_c39
1000.000000	1000.000000	0.0
37928.950000	2005.103000	NaN
18886.252893	6.015861	NaN
70.000000	1995.000000	NaN
30292.500000	2000.000000	NaN
42100.000000	2005.000000	NaN
50822.500000	2010.000000	NaN
79560.000000	2015.000000	NaN

Checking Summary Visually for Ease in Understandings

```
plt.figure(figsize = (15,10))
sns.heatmap(df.describe()[1:].transpose(), linewidths = 1, linecolor = 'black', annot = True, fmt = '.1f' )
```

```
<AxesSubplot:>
```



From above visualization summary we find that:-

1. Umbrella_limit maximum values are very much higher than its overall dataset hence this column dataset probably having outliers.

2. Total_claim_amount having also very high value of maximum which is more far away then its mean as per above graph hence we can say that some of outliers are in the dataset.

Quick Summary of the dataset:

```
df.info()  # checking basic summary of the dataset
```

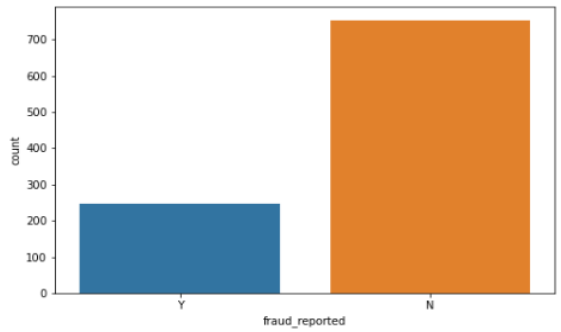
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 40 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   months_as_customer           1000 non-null   int64  
 1   age                          1000 non-null   int64  
 2   policy_number                1000 non-null   int64  
 3   policy_bind_date             1000 non-null   object 
 4   policy_state                 1000 non-null   object 
 5   policy_csl                   1000 non-null   object 
 6   policy_deductable            1000 non-null   int64  
 7   policy_annual_premium        1000 non-null   float64
 8   umbrella_limit               1000 non-null   int64  
 9   insured_zip                  1000 non-null   int64  
 10  insured_sex                  1000 non-null   object 
 11  insured_education_level      1000 non-null   object 
 12  insured_occupation           1000 non-null   object 
 13  insured_hobbies              1000 non-null   object 
 14  insured_relationship         1000 non-null   object 
 15  capital-gains                1000 non-null   int64  
 16  capital-loss                 1000 non-null   int64  
 17  incident_date                1000 non-null   object 
 18  incident_type                1000 non-null   object 
 19  collision_type               1000 non-null   object 
 20  incident_severity            1000 non-null   object 
 21  authorities_contacted        1000 non-null   object 
 22  incident_state               1000 non-null   object 
 23  incident_city                1000 non-null   object 
 24  incident_location            1000 non-null   object 
 25  incident_hour_of_the_day     1000 non-null   int64  
 26  number_of_vehicles_involved  1000 non-null   int64  
 27  property_damage              1000 non-null   object 
 28  bodily_injuries              1000 non-null   int64  
 29  witnesses                    1000 non-null   int64  
 30  police_report_available      1000 non-null   object 
 31  total_claim_amount           1000 non-null   int64  
 32  injury_claim                 1000 non-null   int64  
 33  property_claim               1000 non-null   int64  
 34  vehicle_claim                1000 non-null   int64  
 35  auto_make                    1000 non-null   object 
 36  auto_model                   1000 non-null   object 
 37  auto_year                    1000 non-null   int64  
 38  fraud_reported               1000 non-null   object 
 39  _c39                         0 non-null      float64
dtypes: float64(2), int64(17), object(21)
memory usage: 312.6+ KB
```

Checking Target Variable

Checking fraud_reported

```
plt.figure(figsize = (8,5))
target = df.fraud_reported.value_counts()
print(target)
target = sns.countplot(df.fraud_reported)
```

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



Observations of the target variable

- From above fraud_reported output we can see that dataset are in imbalance count hence we need to fix it further for better performance model later.

Checking Null Values in the dataset

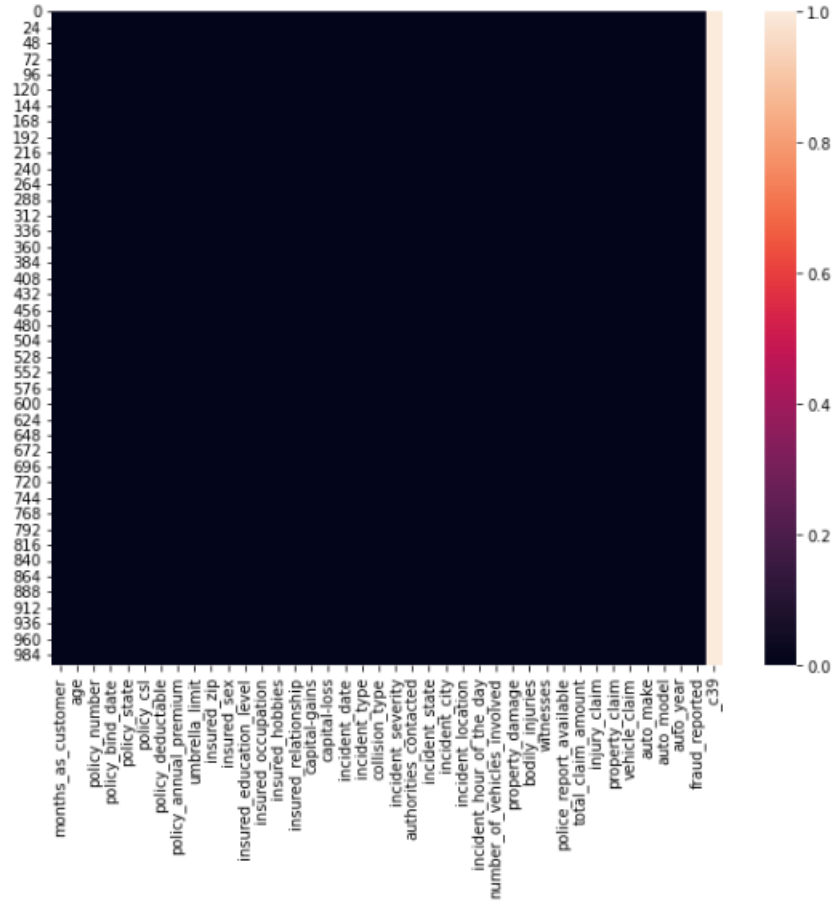
```
df.isnull().sum()/len(df)*100  # checking null values with percentage
```

```
months_as_customer            0.0
age                           0.0
policy_number                 0.0
policy_bind_date              0.0
policy_state                  0.0
policy_csl                    0.0
policy_deductable             0.0
policy_annual_premium         0.0
umbrella_limit                0.0
insured_zip                   0.0
insured_sex                   0.0
insured_education_level       0.0
insured_occupation            0.0
insured_hobbies               0.0
insured_relationship          0.0
capital-gains                 0.0
capital-loss                  0.0
incident_date                 0.0
incident_type                 0.0
collision_type                0.0
incident_severity             0.0
authorities_contacted         0.0
incident_state                0.0
incident_city                 0.0
incident_location             0.0
incident_hour_of_the_day      0.0
number_of_vehicles_involved   0.0
property_damage               0.0
bodily_injuries               0.0
witnesses                     0.0
police_report_available       0.0
total_claim_amount            0.0
injury_claim                  0.0
property_claim                0.0
vehicle_claim                 0.0
auto_make                     0.0
auto_model                    0.0
auto_year                     0.0
fraud_reported                0.0
_c39                        100.0
dtype: float64
```

Checking Null Values of the dataset visually

```
plt.figure(figsize = (10,8))
sns.heatmap(df.isnull())
```

```
<AxesSubplot:>
```



We can see that column _c39 having all rows with NaN values hence we need to drop it.

```
df.drop('_c39', inplace = True, axis = 1)
```

Dropped _c39 from the dataset.

From dataset analysis we find that dataset still having NaN values with ? symbol which means unknown hence we are going to replace with NaN and making assumptions to clean the dataset.

```
df.replace('?', np.NaN, inplace = True)
```

Code to check NaN values columns

```
df.isna().any()[lambda x: x]
```

```
collision_type             True
property_damage            True
police_report_available    True
dtype: bool
```

We find that collision_type, property_damage and police_report_available having NaN values.

Dealing with NaN values by imputation

```
df['collision_type'] = df['collision_type'].fillna(df_mode['collision_type'].mode()[0])
```

```
df['property_damage'] = df['property_damage'].fillna(df['property_damage'].mode()[0])
```

```
df['police_report_available'] = df['police_report_available'].fillna(df['police_report_available'].mode()[0])
```

We are going to fill NaN values with most frequent values of the dataset as NaN columns are of object types.

Converting date columns dataset to date time format

```
df['policy_bind_date'] = pd.to_datetime(df['policy_bind_date'])
df['incident_date'] = pd.to_datetime(df['incident_date'])
```

Extracting important details

```
df['policy_bind_month'] = df['policy_bind_date'].dt.month   # extracting important date things
df['policy_bind_year'] = df['policy_bind_date'].dt.year
```

```
df['incident_date_month'] = df['incident_date'].dt.month   # extracting important date things
df['incident_date_year'] = df['incident_date'].dt.year
```

```
df.incident_date_year.unique()
```

```
array([2015], dtype=int64)
```

We are going to drop this column because we didn't find any machine learning information in it.

Dropping Columns which are not liable in model learning

We find that policy_number, incident location, insured_zip are the unique values of the location and identification of the customer hence these dataset are liable in prediction hence we need to drop it.

We extracted all important details from incident date and policy bind date hence we can drop it.

df.drop(['policy_number', 'policy_bind_date', 'incident_location', 'incident_date_year','incident_date','insured_zip'], inplace = True, axis = 1)

```
df.drop(['policy_number', 'policy_bind_date', 'incident_location', 'incident_date_year','incident_date','insured_zip'], inplace
```

Checking Unique value counts

```
for i in df.columns:
    print(i, "\n", df[i].nunique())
```

```
months_as_customer
 391
age
 46
policy_state
 3
policy_csl
 3
policy_deductable
 3
policy_annual_premium
 991
umbrella_limit
 11
insured_sex
 2
insured_education_level
 7
insured_occupation
 14
insured_hobbies
 20
insured_relationship
 6
capital-gains
 338
capital-loss
 354
incident_type
 4
collision_type
 3
incident_severity
 4
```

```
authorities_contacted
 5
incident_state
 7
incident_city
 7
incident_hour_of_the_day
 24
number_of_vehicles_involved
 4
property_damage
 2
bodily_injuries
 3
witnesses
 4
police_report_available
 2
total_claim_amount
 763
injury_claim
 638
property_claim
 626
vehicle_claim
 726
auto_make
 14
auto_model
 39
auto_year
 21
fraud_reported
 2
policy_bind_month
 12
policy_bind_year
 26
incident_date_month
 12
```

We find that umbrella limit shows very higher values in dataset, this dataset has unique values in a few amounts hence we can convert it into object data type.

```
df['umbrella_limit'] = df['umbrella_limit'].astype('object')
```

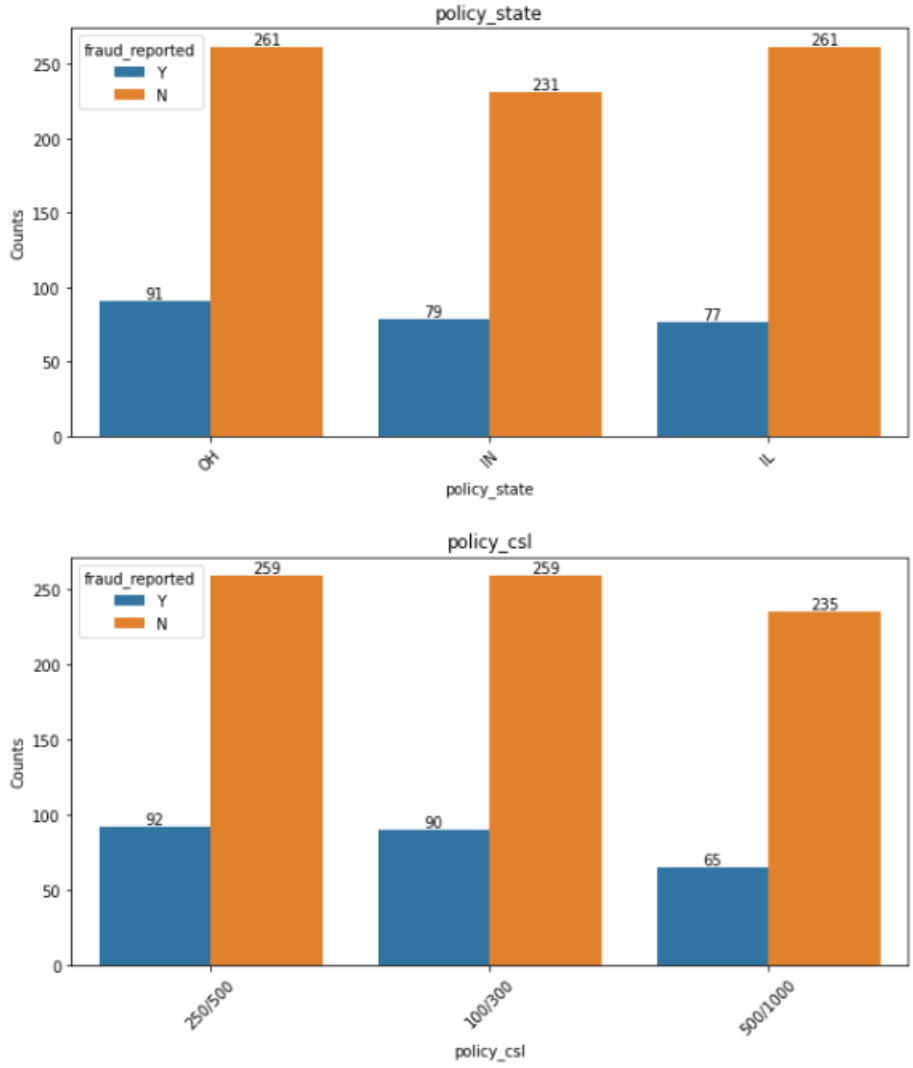
Dividing columns into object and numerical categories for easy visualization understandings

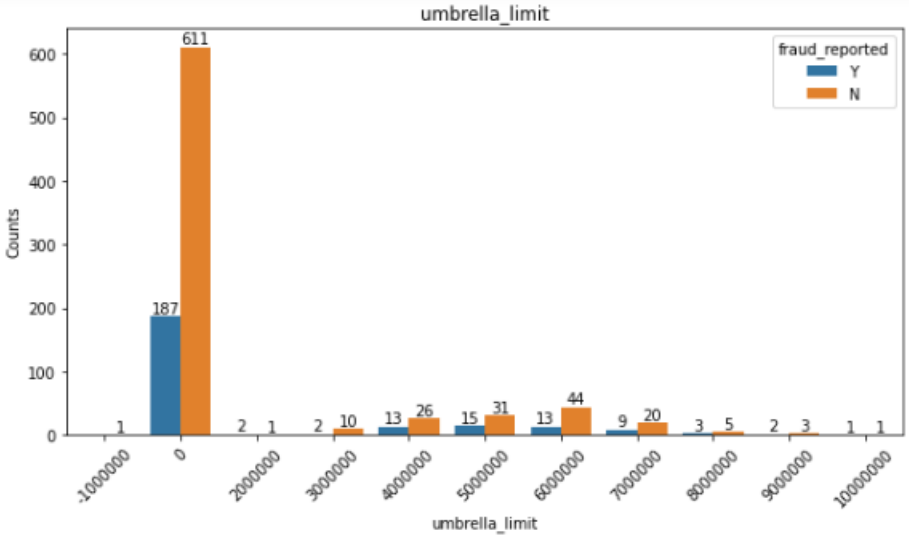
```
df_object = df.select_dtypes(include = 'object').columns  # object dataset columns
df_num = df.select_dtypes(exclude = 'object').columns  # numerical dataset columns
```

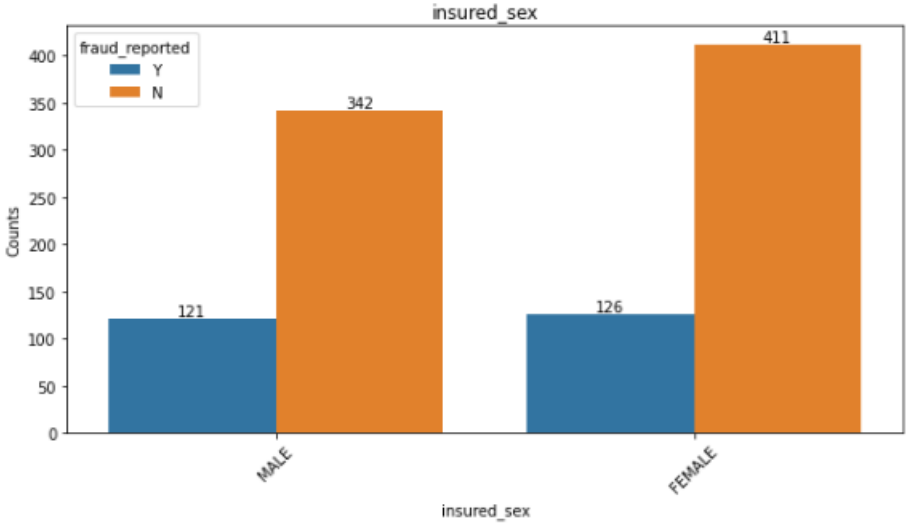
Exploratory Data Analysis Concluding Remarks

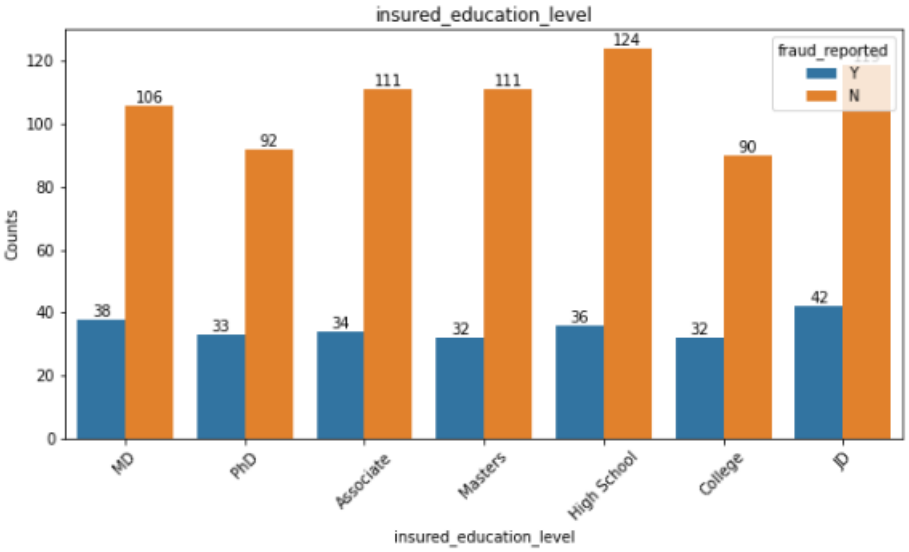
Bar Plot

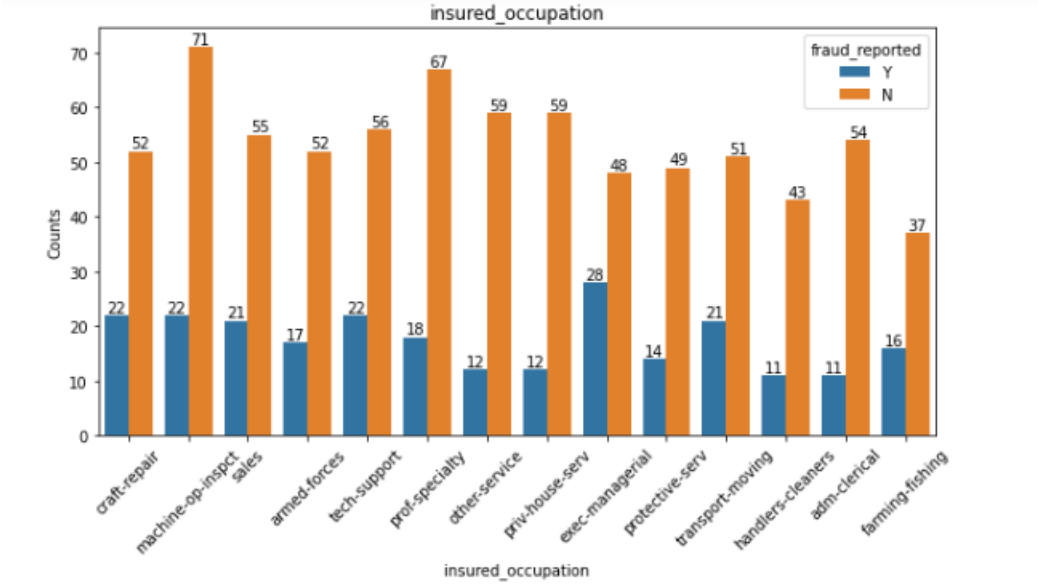
```
for i in df_object:
    graph = df[i].value_counts()
    plt.figure(figsize = (10,5))
    #plt.subplot(9,2,)
    df_graph = sns.countplot(x= i,hue = 'fraud_reported',  data = df)
    df_graph.set_title(i)
    df_graph.set_xlabel(i)
    df_graph.set_ylabel('Counts')
    plt.xticks(rotation = 45)
    for i in df_graph.containers:
            df_graph.bar_label(i)
```

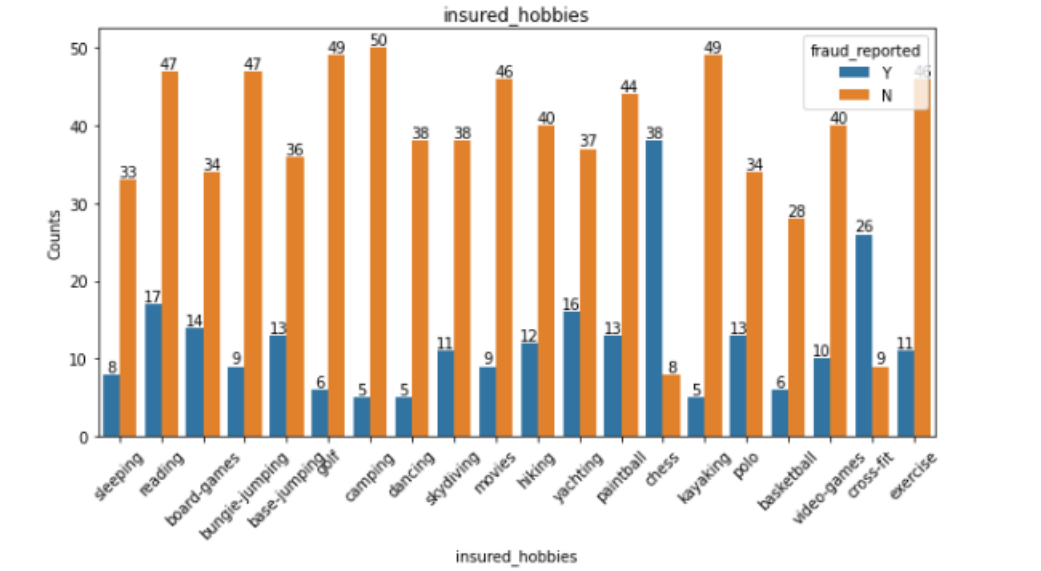


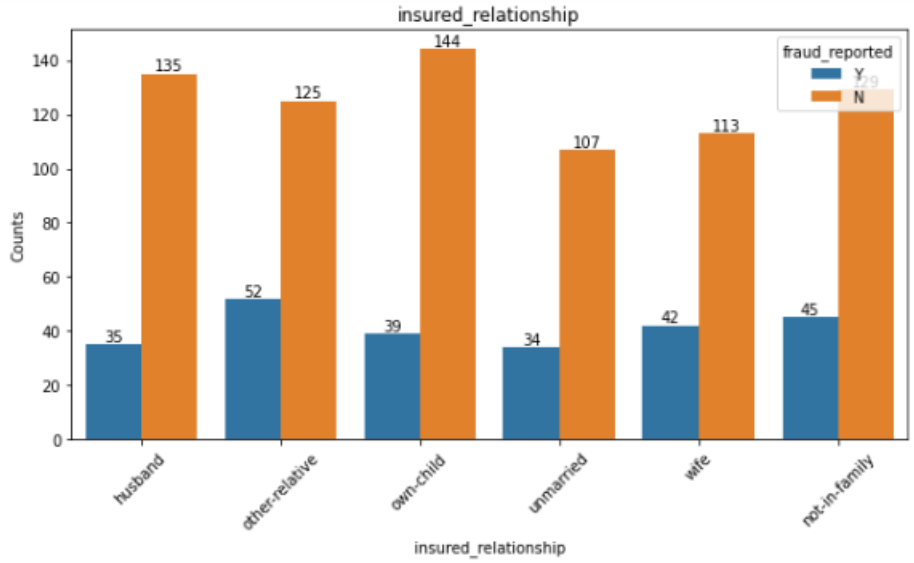

umbrella_limit
fraud_reported
Y
N
Counts
umbrella_limit
-1000000
0
2000000
3000000
4000000
5000000
6000000
7000000
8000000
9000000
10000000
1
187
611
2
1
2
10
13
26
15
31
13
44
9
20
3
5
2
3
1
1

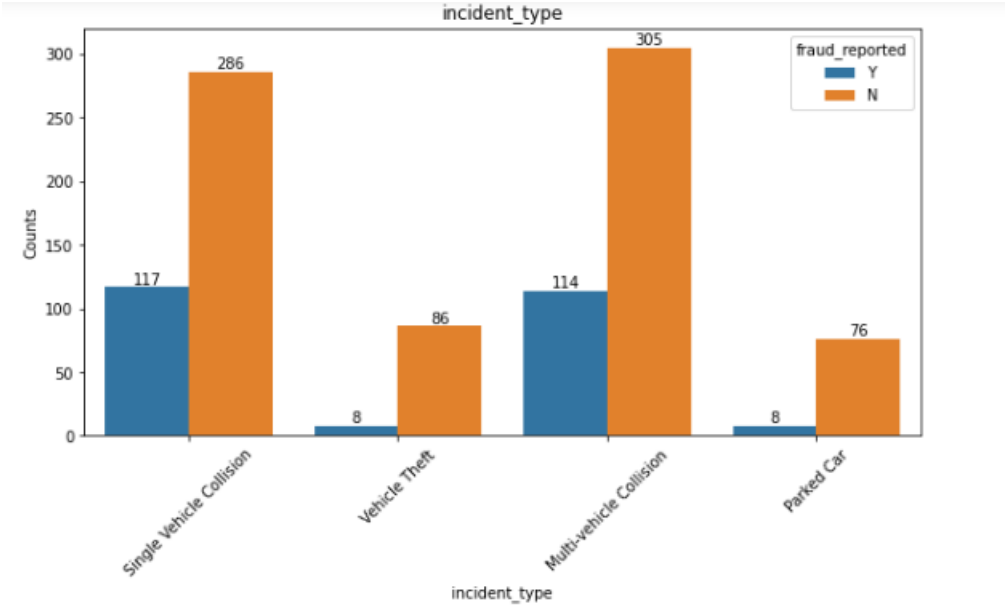

insured_sex
fraud_reported
Y
N
Counts
insured_sex
MALE
FEMALE
121
342
126
411

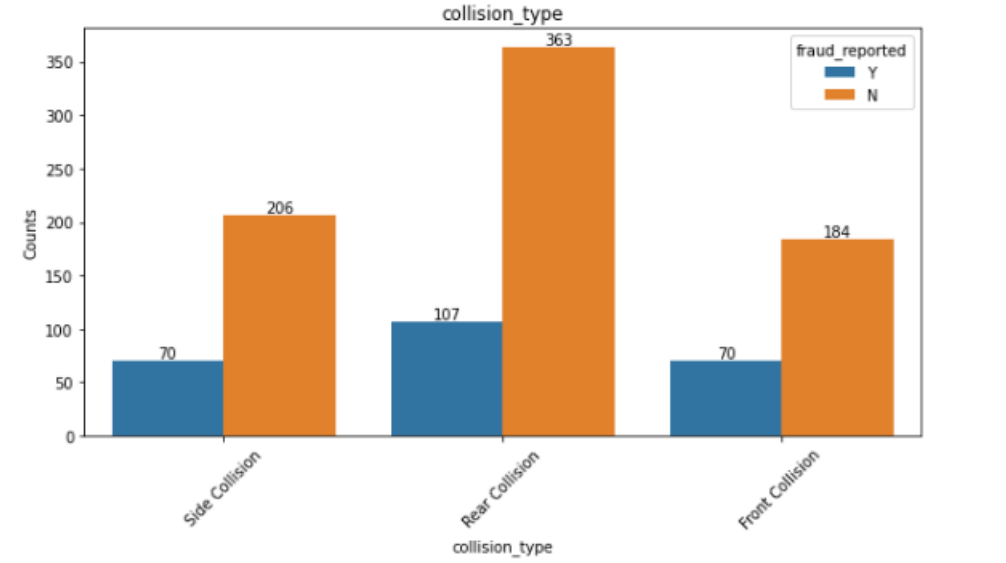

insured_education_level
fraud_reported
Y
N
Counts
insured_education_level
MD
PhD
Associate
Masters
High School
College
JD
38
106
33
92
34
111
32
111
36
124
32
90
42
111

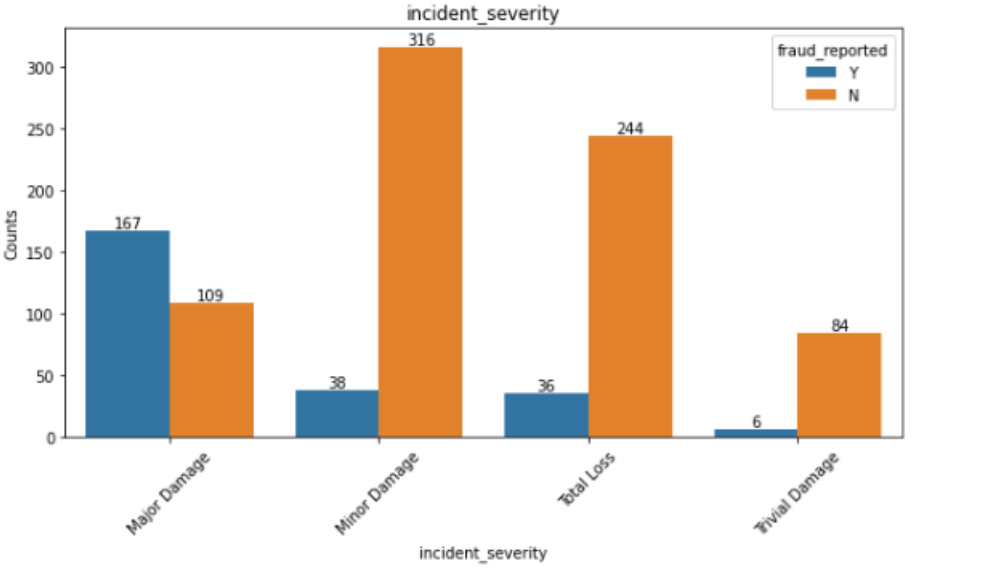

insured_occupation
fraud_reported
Y
N
Counts
craft-repair
machine-op-inspct
sales
armed-forces
tech-support
prof-specialty
other-service
priv-house-serv
exec-managerial
protective-serv
transport-moving
handlers-cleaners
adm-clerical
farming-fishing
insured_occupation

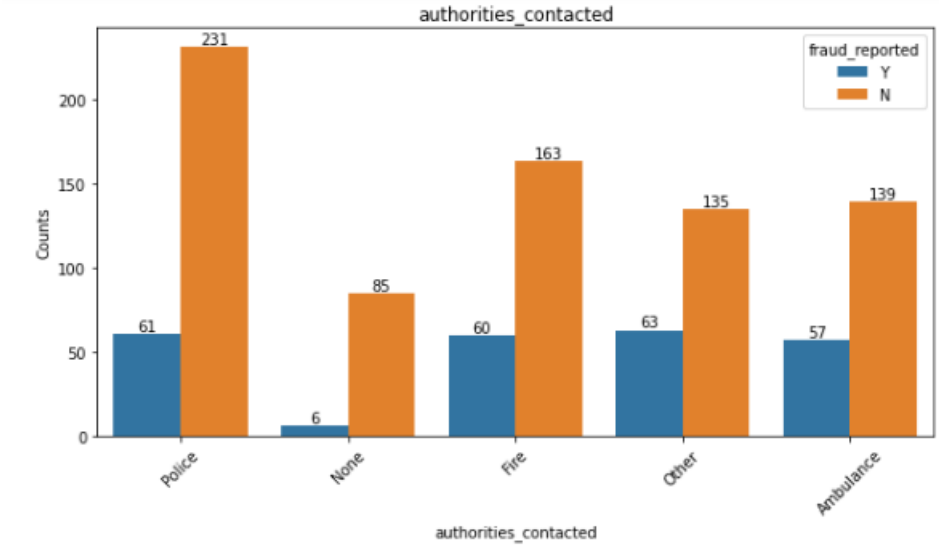

insured_hobbies
fraud_reported
Y
N
Counts
sleeping
reading
board-games
bungie-jumping
base-jumping
golf
camping
dancing
skydiving
movies
hiking
yachting
paintball
chess
kayaking
polo
basketball
video-games
cross-fit
exercise
insured_hobbies

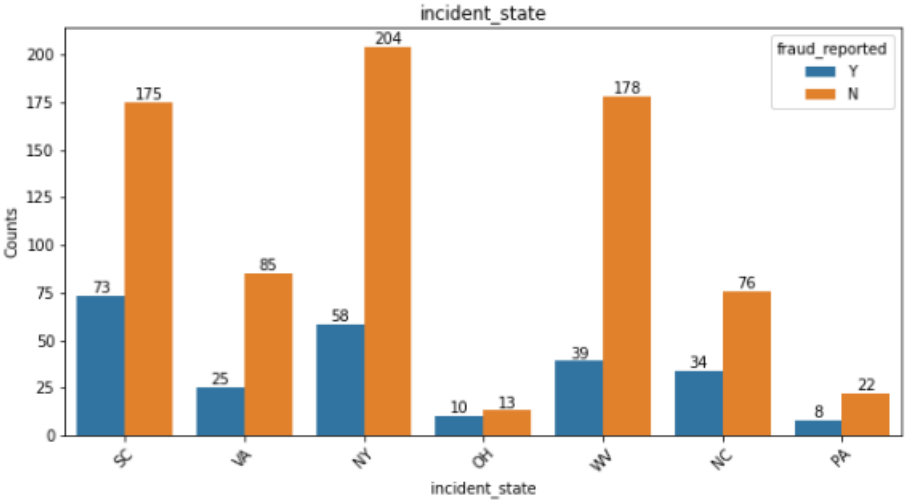

insured_relationship
fraud_reported
Y
N
Counts
husband
other-relative
own-child
unmarried
wife
not-in-family
insured_relationship

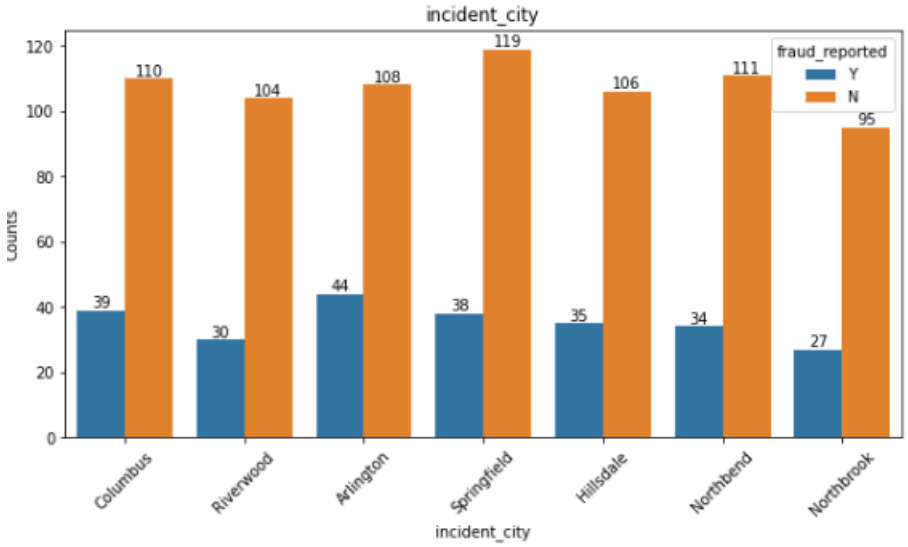

incident_type
fraud_reported
Y
N
Counts
incident_type
Single Vehicle Collision
Vehicle Theft
Multi-vehicle Collision
Parked Car
117
286
8
86
114
305
8
76

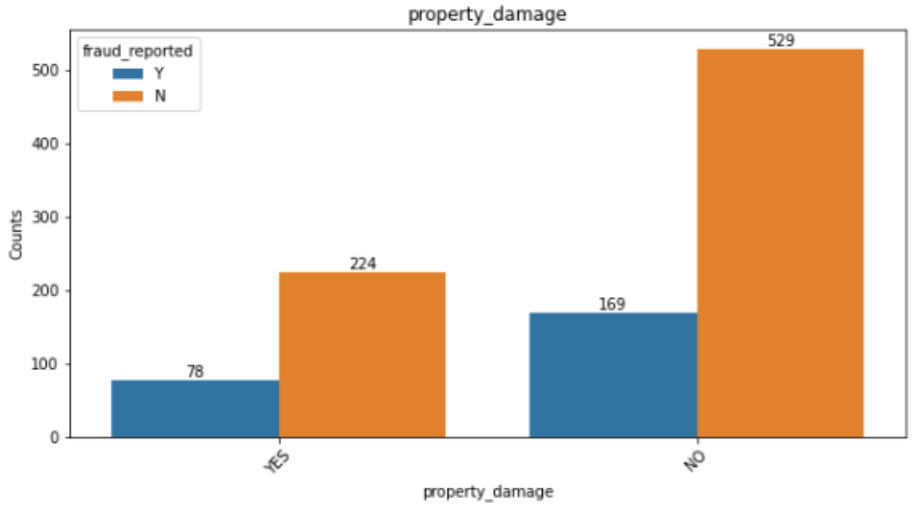

collision_type
fraud_reported
Y
N
Counts
collision_type
Side Collision
Rear Collision
Front Collision
70
206
107
363
70
184

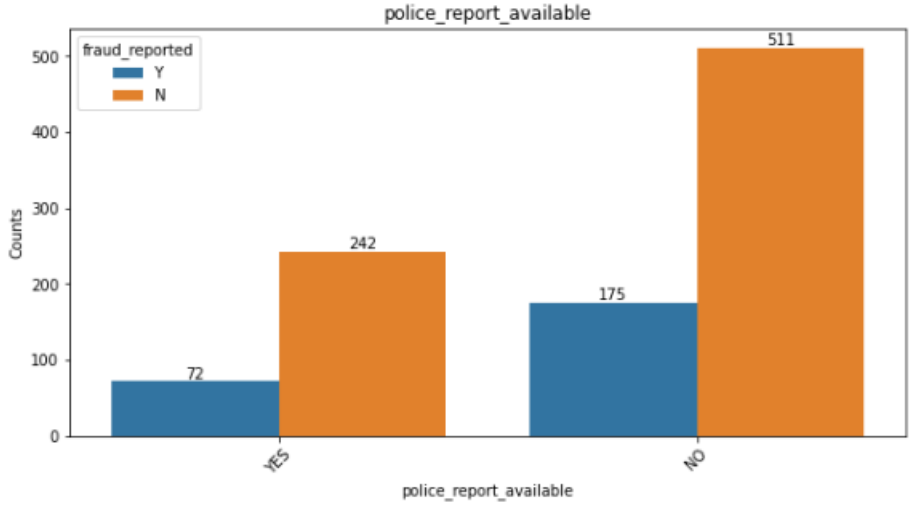

incident_severity
fraud_reported
Y
N
Counts
incident_severity
Major Damage
Minor Damage
Total Loss
Trivial Damage
167
109
38
316
36
244
6
84

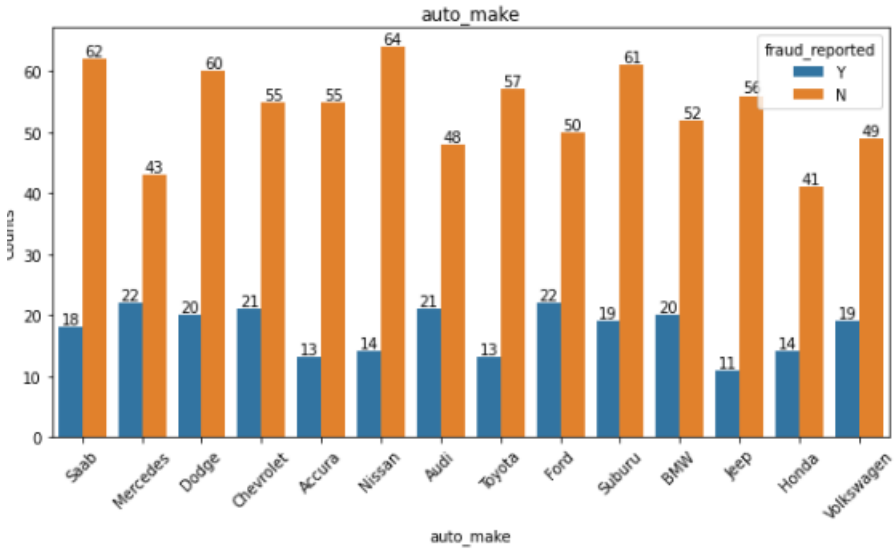

authorities_contacted
fraud_reported
Y
N
Counts
231
61
6
85
60
163
63
135
57
139
Police
None
Fire
Other
Ambulance
authorities_contacted

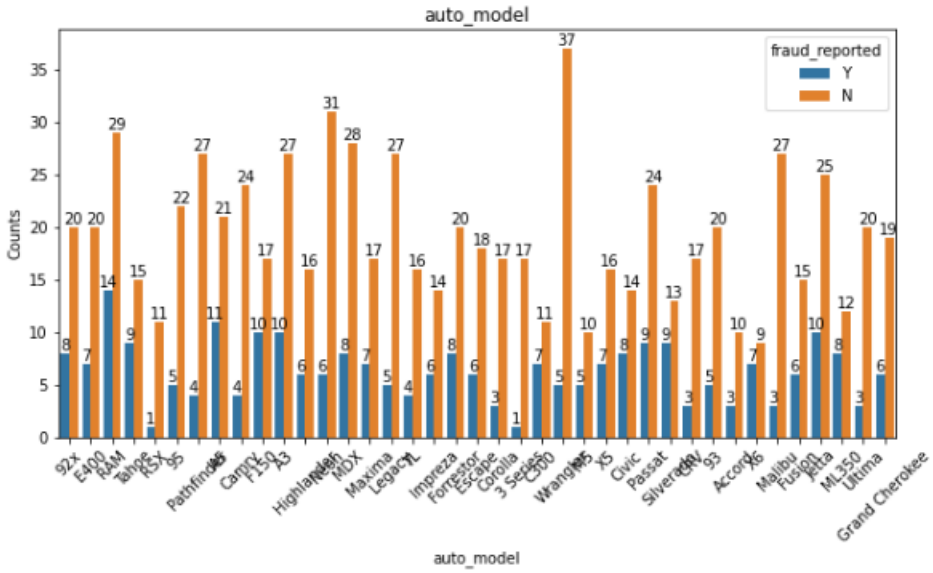

incident_state
fraud_reported
Y
N
Counts
73
175
25
85
58
204
10
13
39
178
34
76
8
22
SC
VA
NY
OH
WV
NC
PA
incident_state

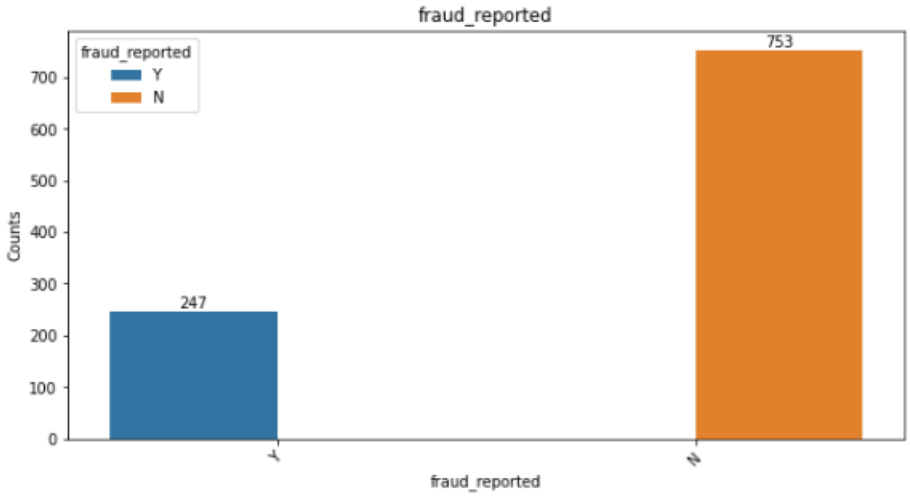

incident_city
fraud_reported
Y
N
Counts
39
110
30
104
44
108
38
119
35
106
34
111
27
95
Columbus
Riverwood
Arlington
Springfield
Hillsdale
Northbend
Northbrook
incident_city


property_damage
fraud_reported
Y
N
Counts
property_damage
YES
NO
78
224
169
529


police_report_available
fraud_reported
Y
N
Counts
police_report_available
YES
NO
72
242
175
511


auto_make
fraud_reported
Y
N
auto_make
Saab
Mercedes
Dodge
Chevrolet
Accura
Nissan
Audi
Toyota
Ford
Suburu
BMW
Jeep
Honda
Volkswagen
62
43
60
55
55
64
48
57
50
61
52
56
41
49
18
22
20
21
13
14
21
13
22
19
20
11
14
19





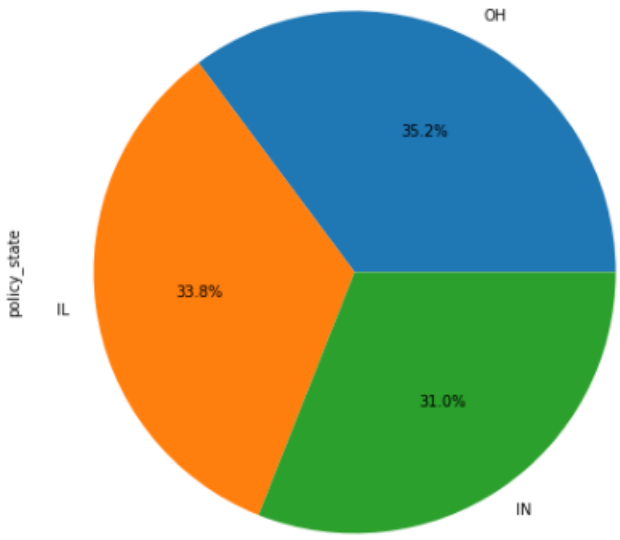
Observations :

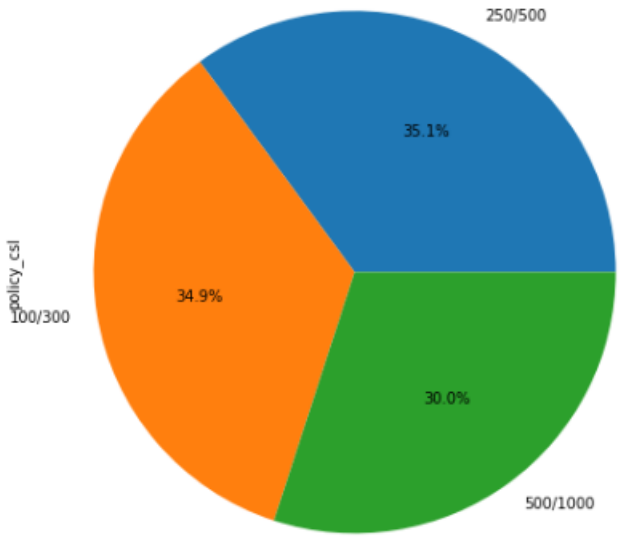
1. policy_state with OH are morely going to fraud as per dataset.
2. 250/500 policy_csl are going to fraud as per dataset.
3. insured_sex : Female candidates are more fraudster than males as per given dataset.
4. insured_education_level : JD education people are more likely to do fraud as per dataset.
5. insured_occupation : exec-managerial are going to fraud more.
6. insured_hobbies : People with hobby chess are going to fraud more as per dataset.
7. insured_relationship: other-relative relationship are fraud more as per dataset.
8. incident_type: Single Vehicle Collision are going to fraud more.
9. collision_type: Rear Collision are going to fraud more as per dataset.
10. incident_severity: Major Damage severity people going to fraud more as per dataset.
11. authorities_contacted: Most of fraudster contact Other authorities as per dataset.
12. incident_state: Most of the fraudster are of SC state.
13. incident_city: Arlington city having more chances of fraudster.
14. property_damage: No property damage people are going to fraud more as per dataset.

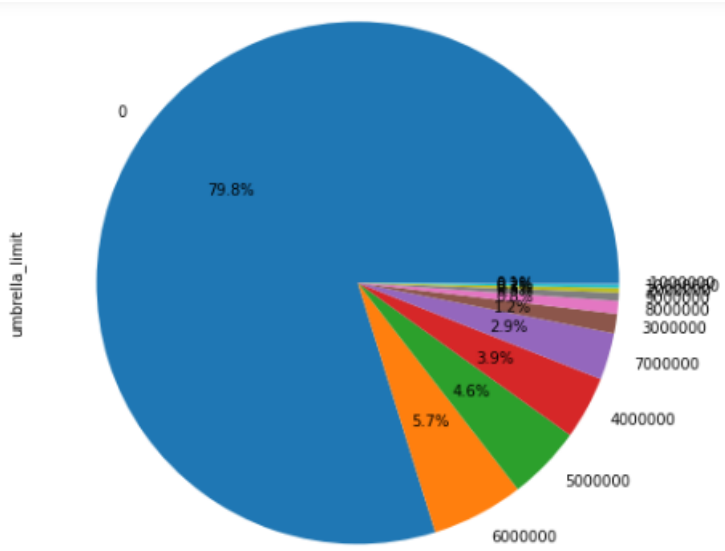
15. police_report_available: No police report people having chances to fraud more as per dataset.
16. auto_make : Mercedes, Ford people are morely to report as fraud as per dataset.
17. auto_model: RAM having more count of fraud as per dataset.

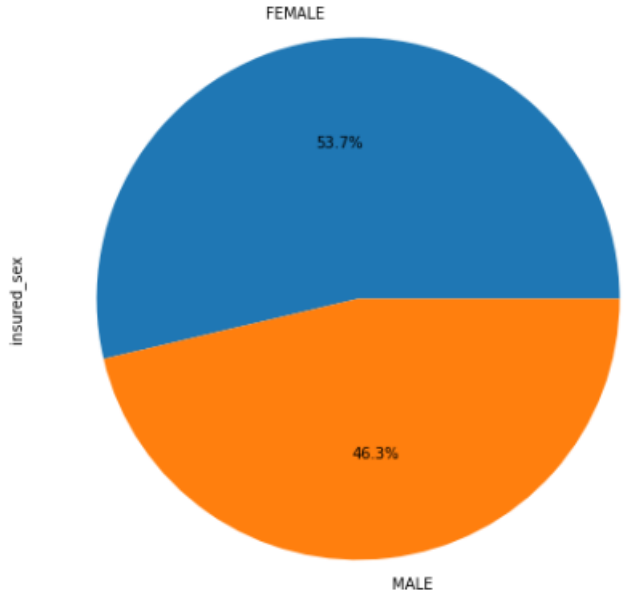
Pie Chart

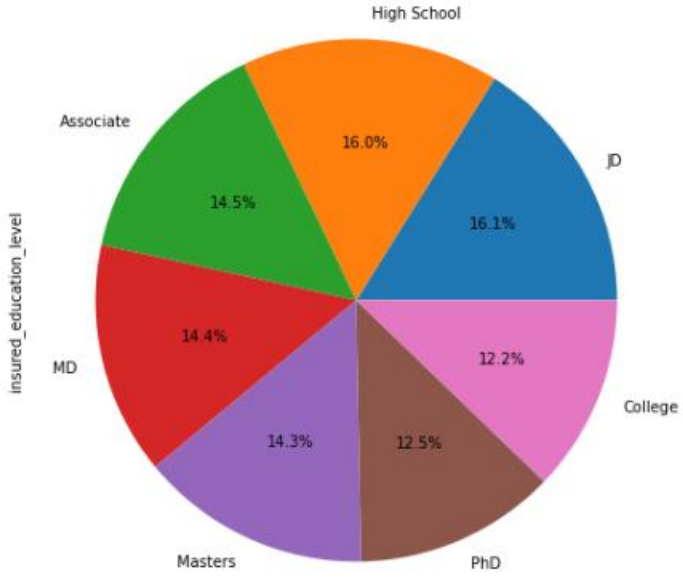
```
for i in df_object:
    graph = df[i].value_counts()
    plt.figure(figsize = (10,8))
    graph.plot(kind = 'pie', autopct = '%1.1f%%')
```

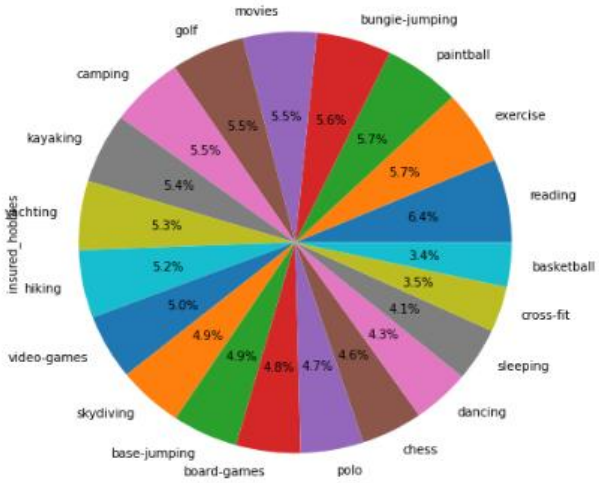


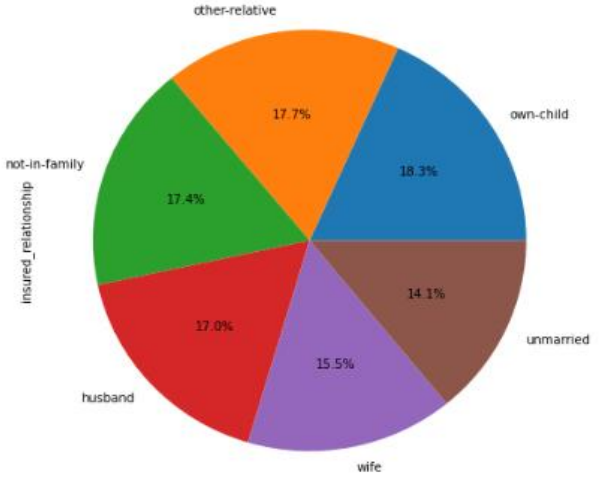


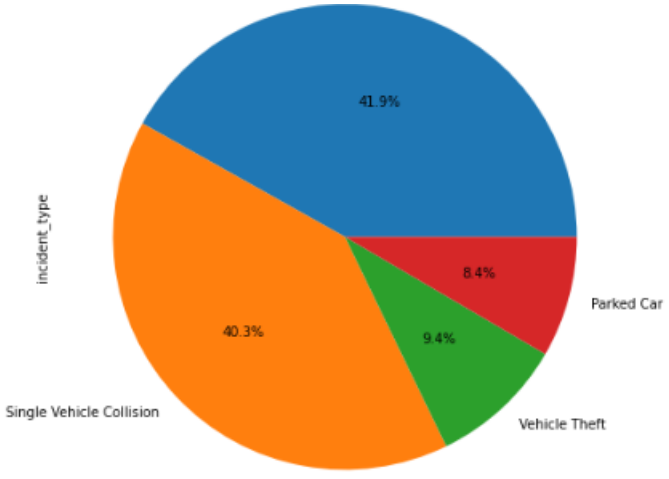

umbrella_limit
0
79.8%
1.2%
2.9%
3.9%
4.6%
5.7%
8000000
3000000
7000000
4000000
5000000
6000000

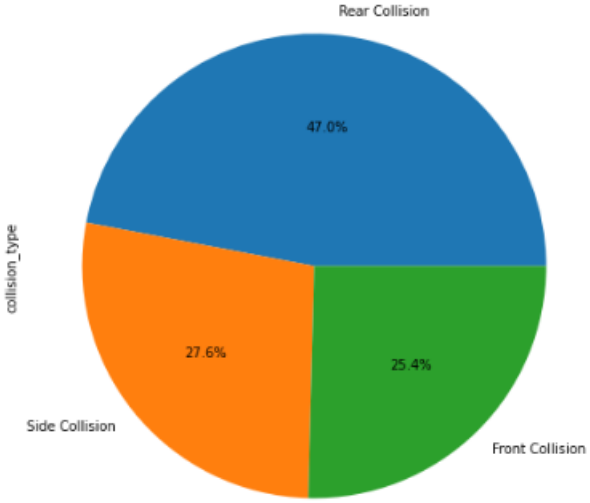

insured_sex
FEMALE
53.7%
46.3%
MALE

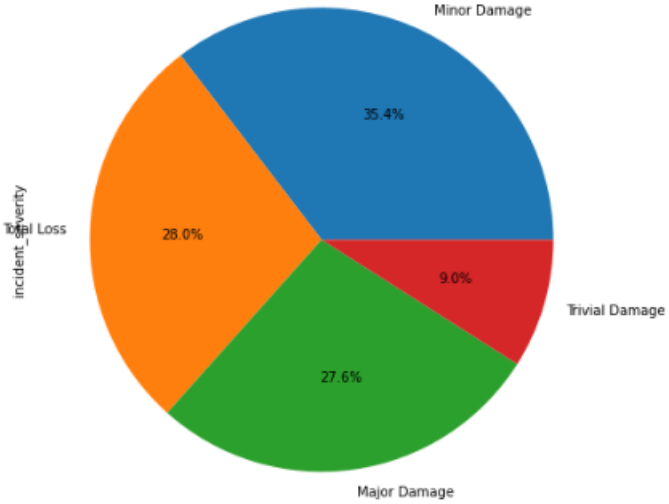

insured_education_level
High School
16.0%
JD
16.1%
Associate
14.5%
MD
14.4%
College
12.2%
Masters
14.3%
PhD
12.5%

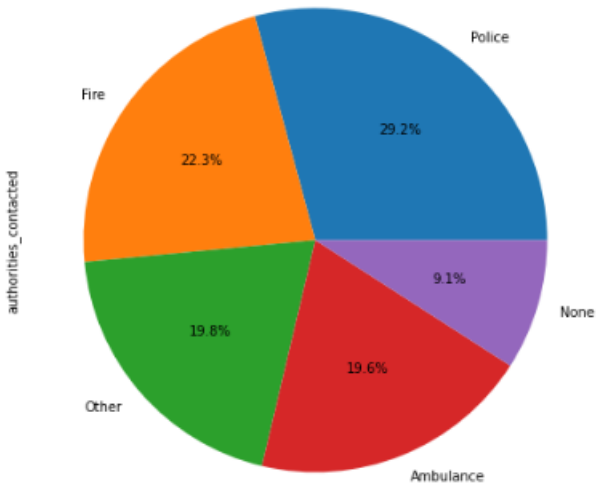

insured_hobbies
movies
bungie-jumping
golf
paintball
camping
exercise
kayaking
reading
yachting
basketball
hiking
cross-fit
video-games
sleeping
skydiving
dancing
base-jumping
chess
board-games
polo
5.5%
5.6%
5.5%
5.7%
5.5%
5.7%
5.4%
6.4%
5.3%
3.4%
5.2%
3.5%
5.0%
4.1%
4.9%
4.3%
4.9%
4.8%
4.7%
4.6%

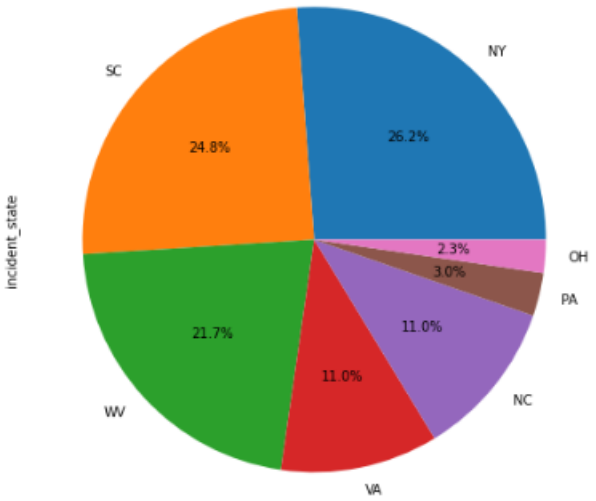

insured_relationship
other-relative
own-child
not-in-family
unmarried
husband
wife
17.7%
18.3%
17.4%
14.1%
17.0%
15.5%

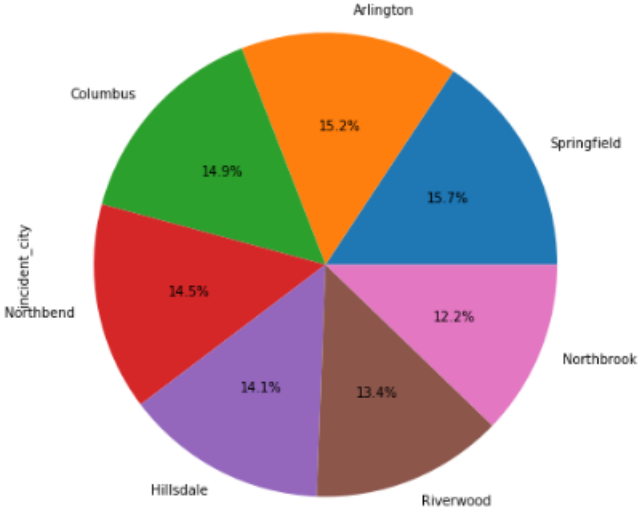

incident_type
Parked Car
Single Vehicle Collision
Vehicle Theft
41.9%
8.4%
40.3%
9.4%

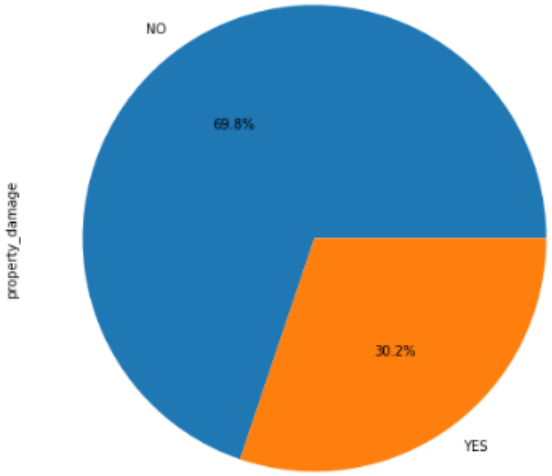

Rear Collision
47.0%
collision_type
27.6%
25.4%
Side Collision
Front Collision

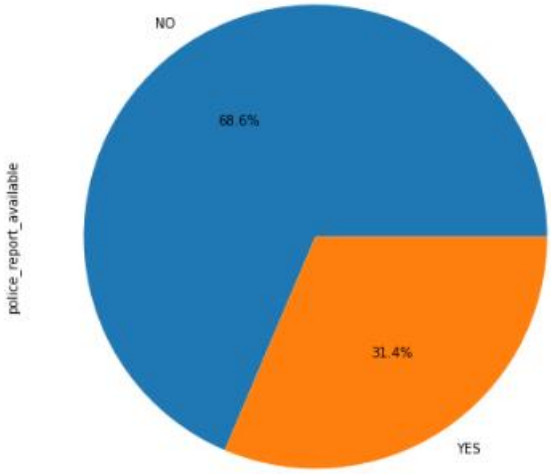

Minor Damage
35.4%
incident_severity
Total Loss
28.0%
9.0%
Trivial Damage
27.6%
Major Damage

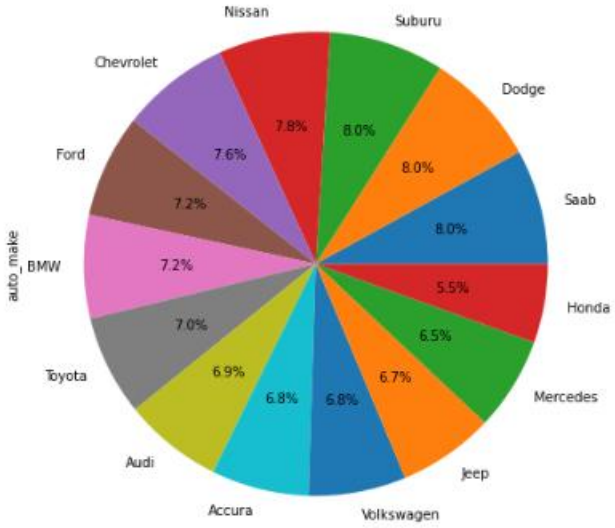

Police
Fire
29.2%
22.3%
authorities_contacted
9.1%
None
19.8%
19.6%
Other
Ambulance

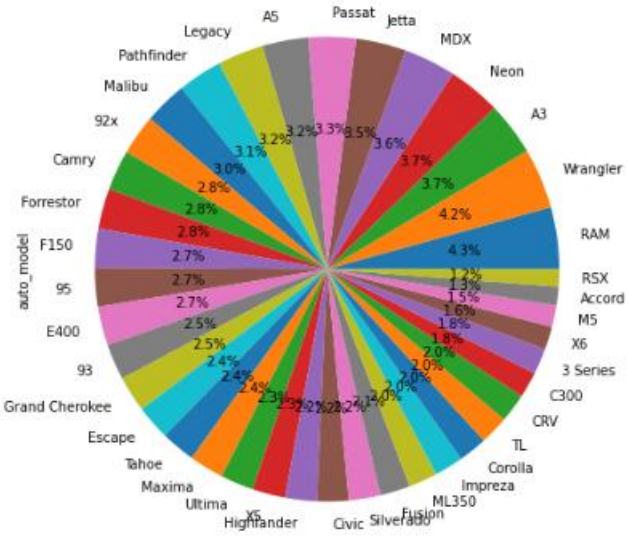

incident_state
NY
SC
OH
PA
NC
WV
VA
26.2%
24.8%
2.3%
3.0%
11.0%
21.7%
11.0%

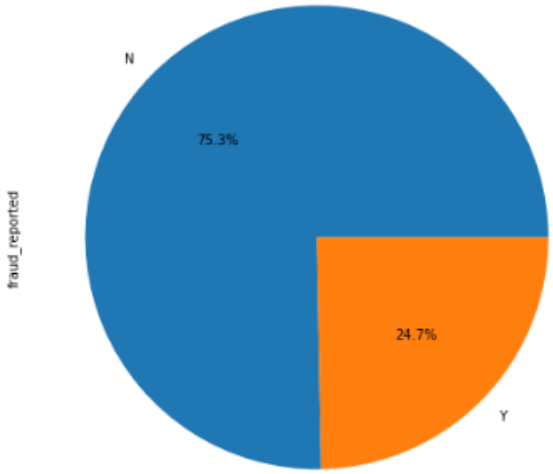

incident_city
Arlington
Columbus
Springfield
Northbend
Northbrook
Hillsdale
Riverwood
15.2%
14.9%
15.7%
14.5%
12.2%
14.1%
13.4%


property_damage
NO
YES
69.8%
30.2%


police_report_available
NO
68.6%
31.4%
YES


auto_make
Nissan
Suburu
Dodge
Saab
Honda
Mercedes
Jeep
Volkswagen
Accura
Audi
Toyota
BMW
Ford
Chevrolet
7.8%
8.0%
8.0%
8.0%
5.5%
6.5%
6.7%
6.8%
6.8%
6.9%
7.0%
7.2%
7.2%
7.6%


auto_model
A5
Passat
Jetta
MDX
Neon
A3
Wrangler
RAM
RSX
Accord
M5
X6
3 Series
C300
CRV
TL
Corolla
Impreza
ML350
Fusion
Silverado
Civic
Highlander
XS
Ultima
Maxima
Tahoe
Escape
Grand Cherokee
93
E400
95
F150
Forrestor
Camry
92x
Malibu
Pathfinder
Legacy
4.3%
4.2%
3.7%
3.7%
3.6%
3.5%
3.3%
3.2%
3.2%
3.1%
3.0%
2.8%
2.8%
2.8%
2.7%
2.7%
2.7%
2.5%
2.5%
2.4%
2.4%



Observations we find:

1. policy_state 35.2 percent of the people select OH policy as per dataset.
2. policy_csl: 35.1 percent of the people select OH policy as per dataset.
3. umbrella limit - Most of the people doesn't having umbrella limit only 20.2 percent ratio of people having umbrella limit.
4. insured_sex : Female candidates are more than males as per given dataset.
5. insured_education_level : JD education people are more people rather than others as per dataset.
6. insured_occupation : 8.5 ratio are of prof-specialty procession people which are higher from among all occupation.
7. insured_hobbies : People with hobby reading are more as per dataset.
8. insured_relationship: own child relationship are more as per dataset upto 18.3 ratio.
9. incident_type: Multi Vehicle Collision are higher in ratio.
10. collision_type: Rear Collision are more as per dataset.
11. incident_severity: Minor Damage severity people are more in auto mobiles.
12. authorities_contacted: Most of people contact police as per dataset.
13. incident_state: Most of the people are of SC state as per graph.
14. incident_city: Arlington city having more chances of fraudster as most of the people are of Arlington.
15. property_damage: No property damage are more as per dataset with ratio 69.8.
16. police_report_available: No police report available for 68.6 percent cases as per dataset.
17. auto_make : suburu and Dodge are more in auto make per dataset with ratio of 8 percent.
18. auto_model: RAM having more count of as per dataset with ratio of 4.2%.

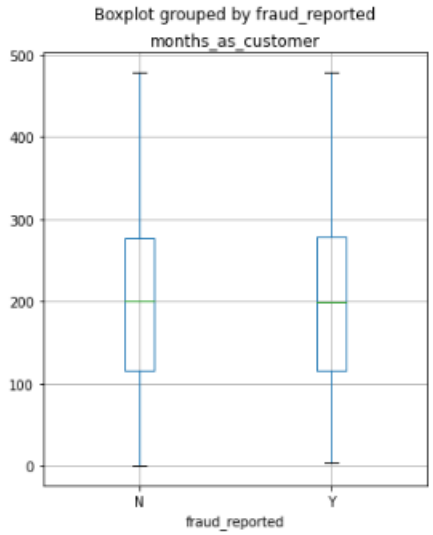
From above all observations we find that Auto_model does not give any informtion about model building because fraudster doesn't show interest in a specific model.
Hence we can drop it.

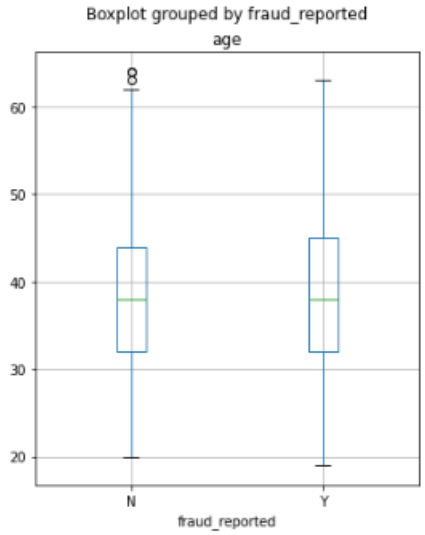
```
df.drop('auto_model', axis = 1, inplace  = True)
```

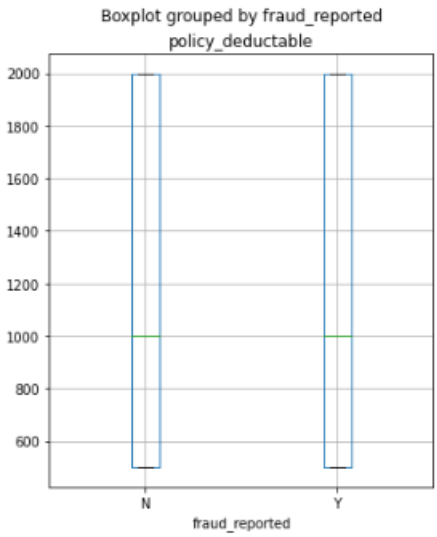
Numerical Columns dataset

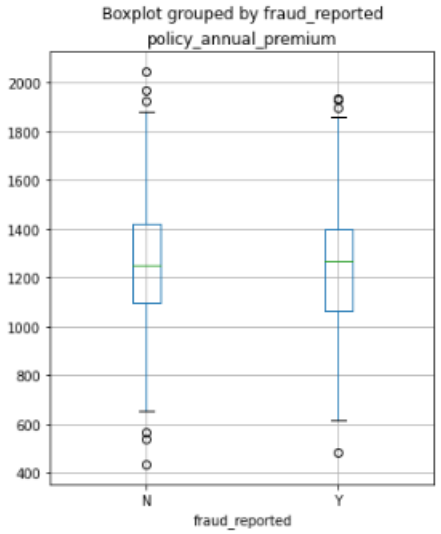
Box Plot

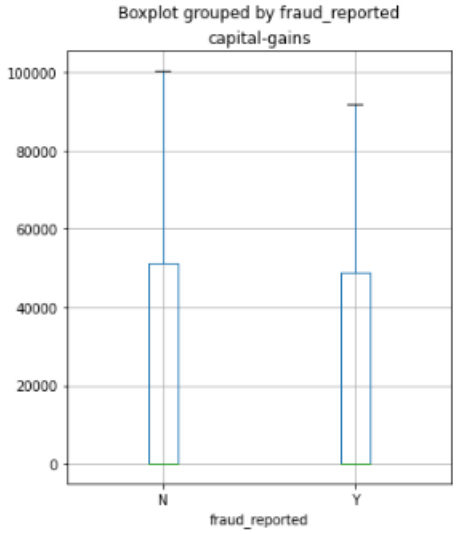
```
for i in df_num:
    df.boxplot(column = i, by = 'fraud_reported', figsize = (5,6))
```

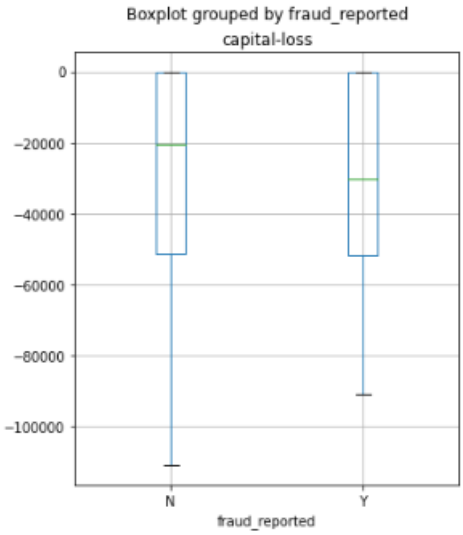


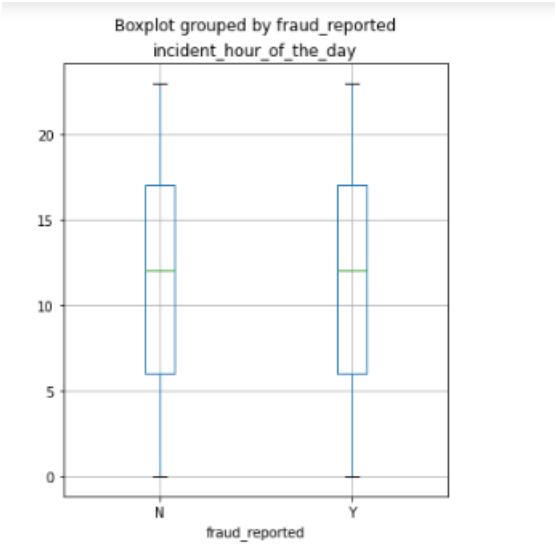


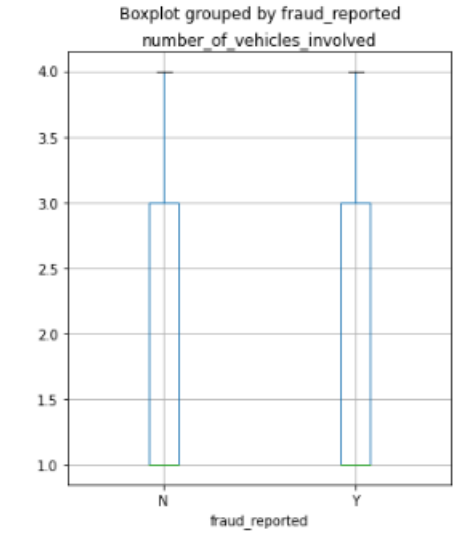


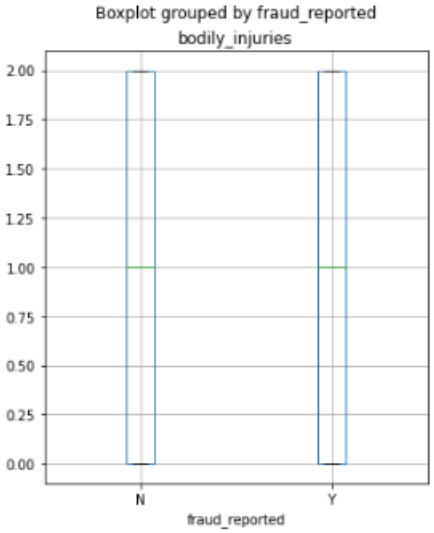

Boxplot grouped by fraud_reported
policy_annual_premium
2000
1800
1600
1400
1200
1000
800
600
400
N
Y
fraud_reported

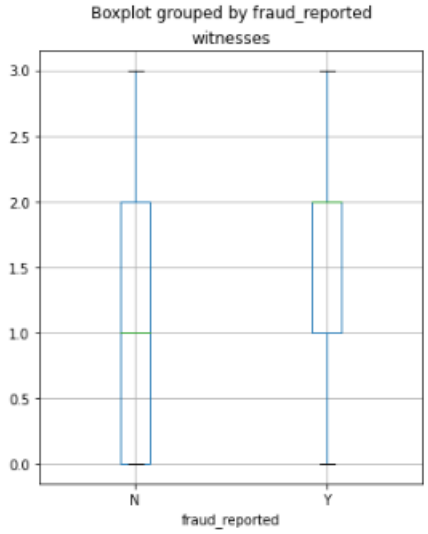

Boxplot grouped by fraud_reported
capital-gains
100000
80000
60000
40000
20000
0
N
Y
fraud_reported

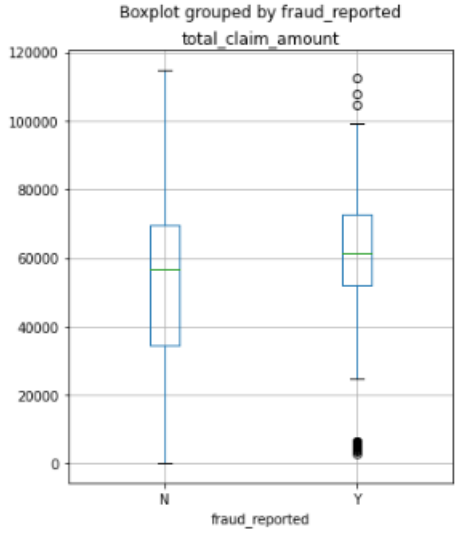

Boxplot grouped by fraud_reported
capital-loss
0
−20000
−40000
−60000
−80000
−100000
N
Y
fraud_reported

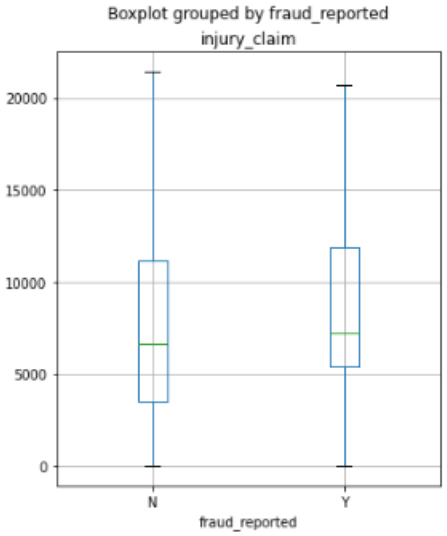

Boxplot grouped by fraud_reported
incident_hour_of_the_day
20
15
10
5
0
N
Y
fraud_reported

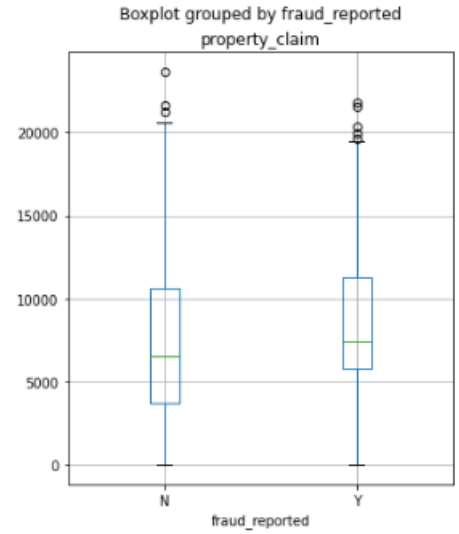

Boxplot grouped by fraud_reported
number_of_vehicles_involved
4.0
3.5
3.0
2.5
2.0
1.5
1.0
N
Y
fraud_reported

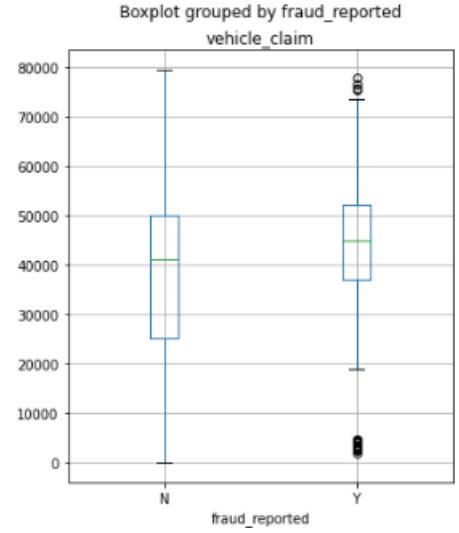

Boxplot grouped by fraud_reported
bodily_injuries
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
N
Y
fraud_reported

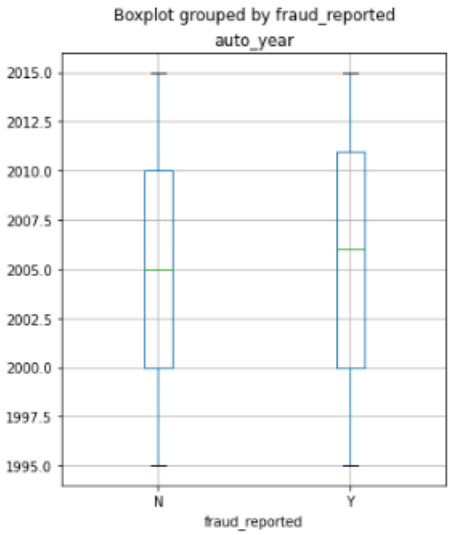

Boxplot grouped by fraud_reported
witnesses
3.0
2.5
2.0
1.5
1.0
0.5
0.0
N
Y
fraud_reported

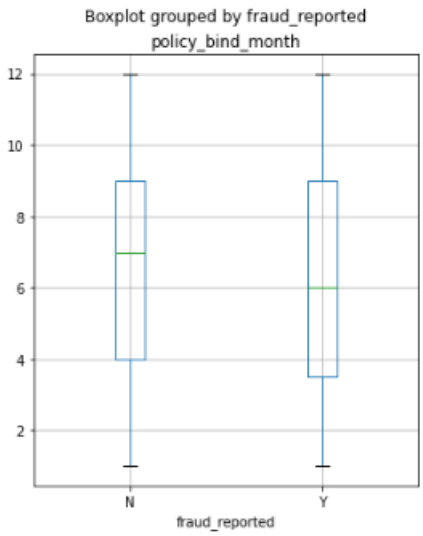

Boxplot grouped by fraud_reported
total_claim_amount
120000
100000
80000
60000
40000
20000
0
N
Y
fraud_reported

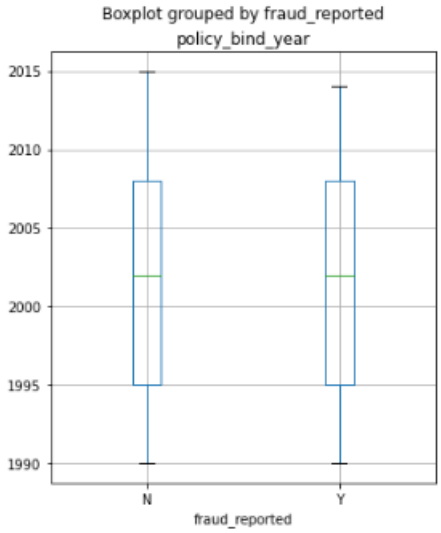

Boxplot grouped by fraud_reported
injury_claim
20000
15000
10000
5000
0
N
Y
fraud_reported

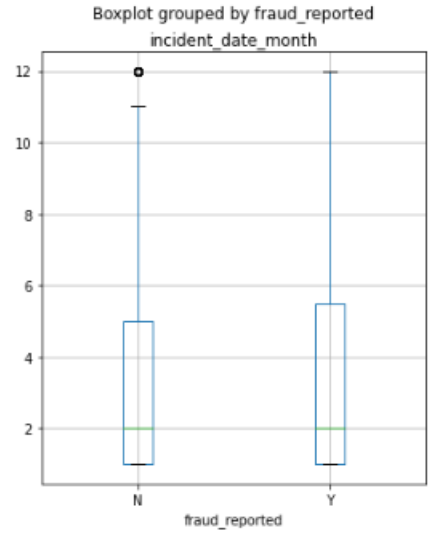

Boxplot grouped by fraud_reported
property_claim
20000
15000
10000
5000
0
N
Y
fraud_reported


Boxplot grouped by fraud_reported
vehicle_claim
80000
70000
60000
50000
40000
30000
20000
10000
0
N
Y
fraud_reported


Boxplot grouped by fraud_reported
auto_year
2015.0
2012.5
2010.0
2007.5
2005.0
2002.5
2000.0
1997.5
1995.0
N
Y
fraud_reported


Boxplot grouped by fraud_reported
policy_bind_month
12
10
8
6
4
2
N
Y
fraud_reported


Boxplot grouped by fraud_reported
policy_bind_year
2015
2010
2005
2000
1995
1990
N
Y
fraud_reported


Boxplot grouped by fraud_reported
incident_date_month
12
10
8
6
4
2
N
Y
fraud_reported

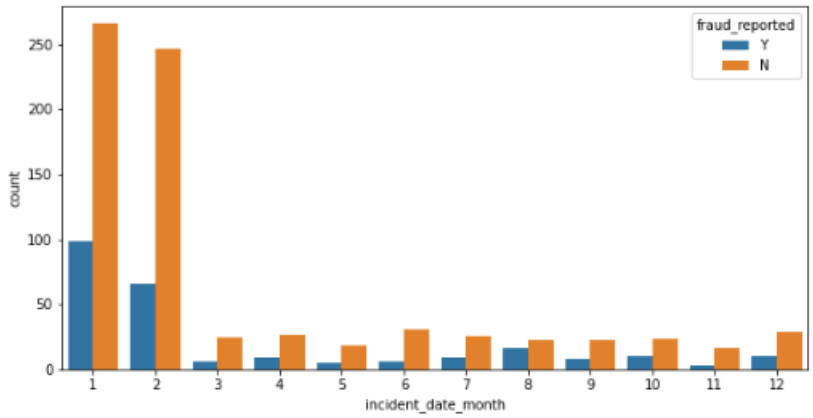
Observations:-

1. age: age column dataset having a neglible amount of outliers as per above graph.
2. policy_annual_premimum having outliers in both criteria with fraud reported yes and no.
3. umrella_limit having too much outliers as per above graph.
4. totol _claim_amount having outliers in the dataset of fraud reported as yes.
5. property_claim having outliers in both criteria.
6. vehicle claim having outliers only in the dataset of Y fraud reported.

Fraud_reported count with respect to incident_data_month

```
plt.figure(figsize = (10,5))
sns.countplot(x = 'incident_date_month', hue = 'fraud_reported', data = df)
```

```
<AxesSubplot:xlabel='incident_date_month', ylabel='count'>
```

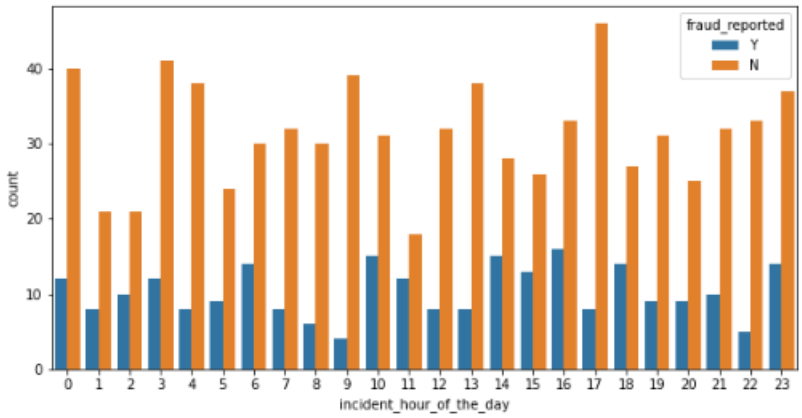


Observations: From above graph we find that in the month of January most of the people being reported as fraud.

Fraud_reported count with respect to incident_hour_of_the_day

```
plt.figure(figsize = (10,5))
sns.countplot(x = 'incident_hour_of_the_day', hue = 'fraud_reported', data = df)
```

```
<AxesSubplot:xlabel='incident_hour_of_the_day', ylabel='count'>
```

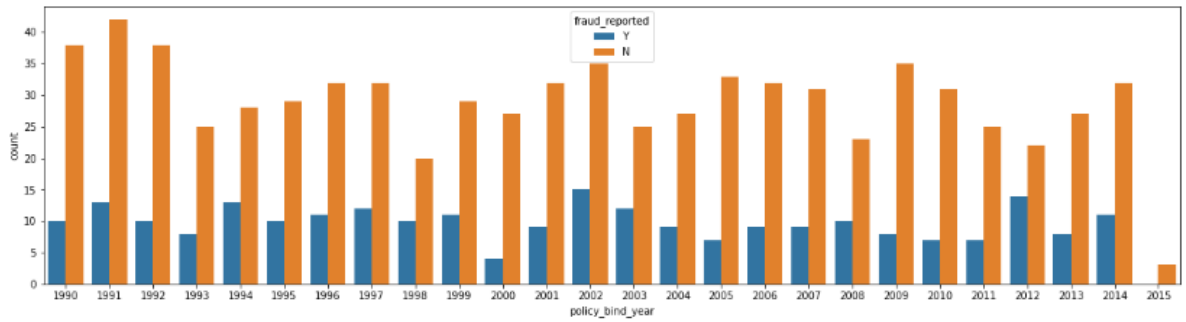


Observations: There is not a common pattern to say that that on this hour claim peopled are going to do fraud in future.

Fraud_reported count with respect to Policy_bind_year

```
plt.figure(figsize = (20,5))
sns.countplot(x = 'policy_bind_year', hue = 'fraud_reported', data = df)
```

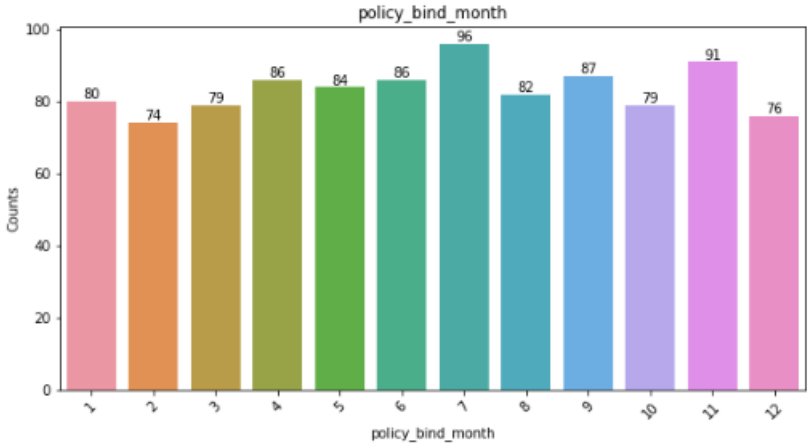
```
<AxesSubplot:xlabel='policy_bind_year', ylabel='count'>
```



Observations: There is not a common pattern to say that that on this hour claim peopled are going to do fraud in future.

Policy_bind_month counts

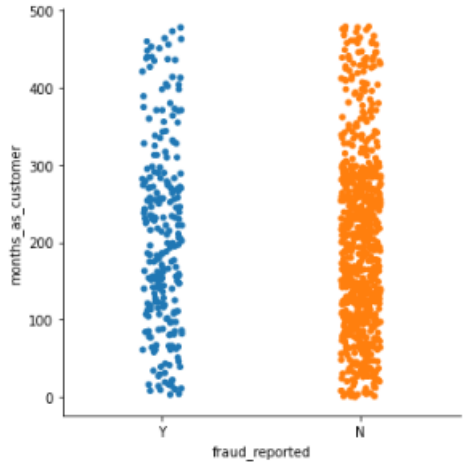
```
plt.figure(figsize = (10,5))
pbm = sns.countplot(x= 'policy_bind_month', data = df)
pbm.set_title('policy_bind_month')
pbm.set_xlabel('policy_bind_month')
pbm.set_ylabel('Counts')
plt.xticks(rotation = 45)
for i in pbm.containers:
    pbm.bar_label(i)
```

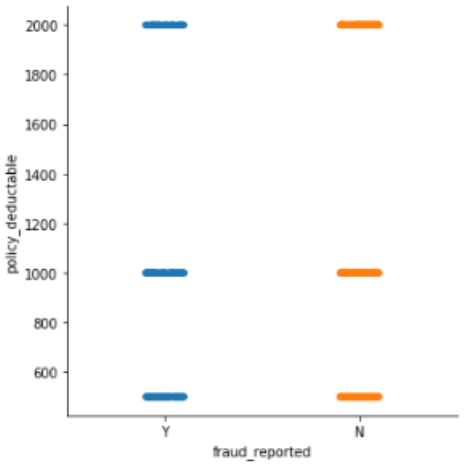


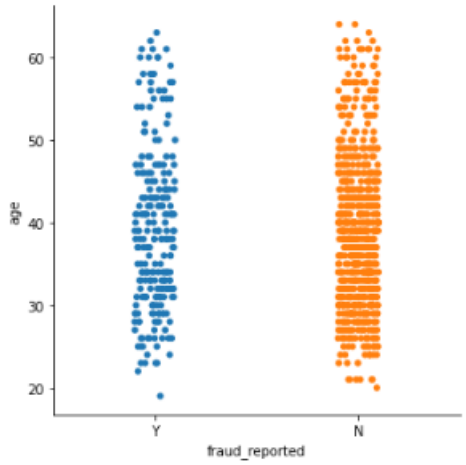
Observations: On 7 month of year maximum people bind with a policy as per graph and very low in February.

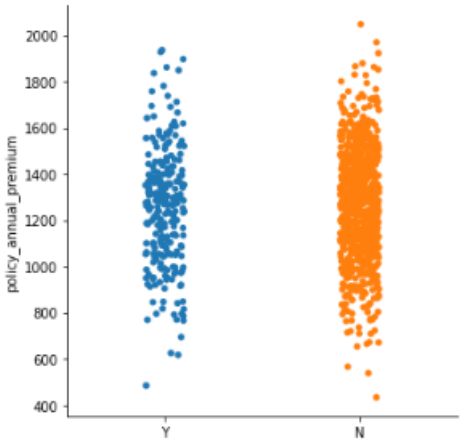
Categorical Plot

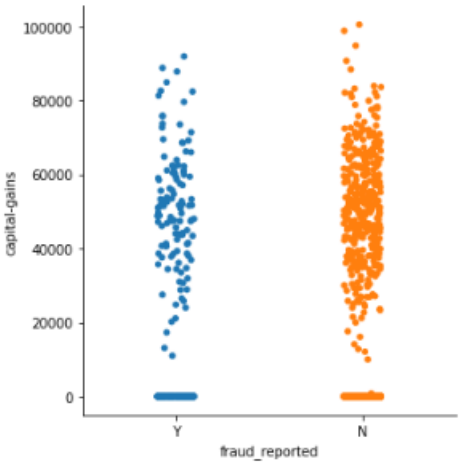
```
for i in df_num:
    sns.catplot(x = 'fraud_reported', y = i, data = df)
```

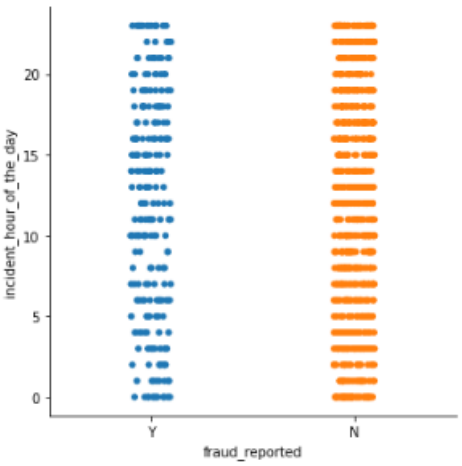


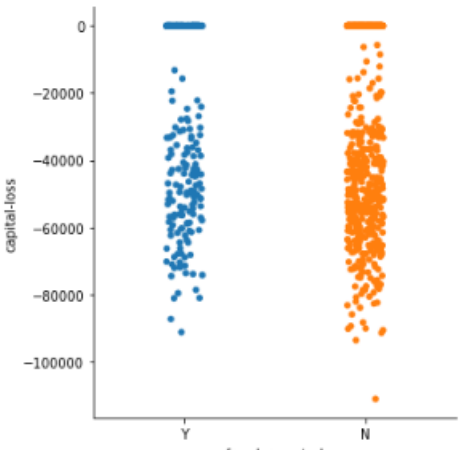


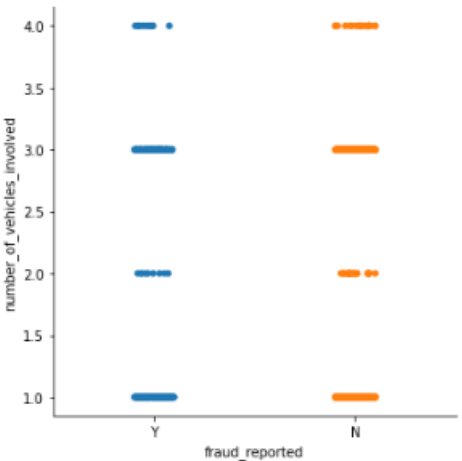


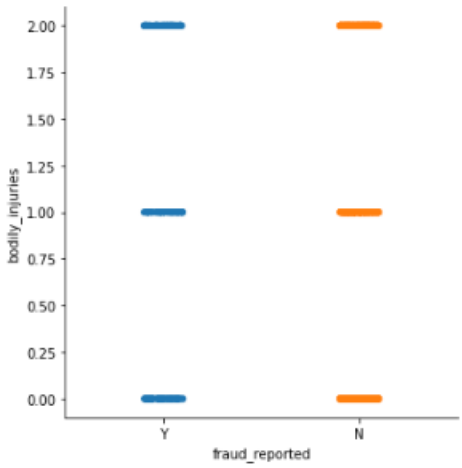


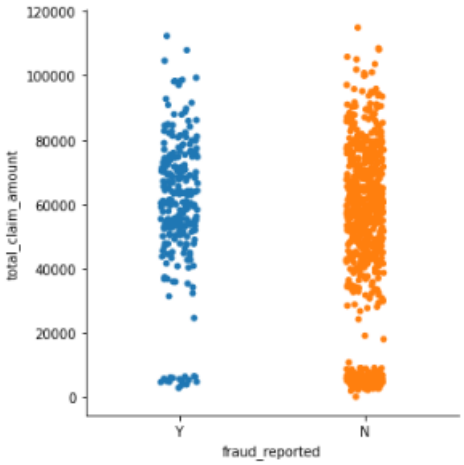


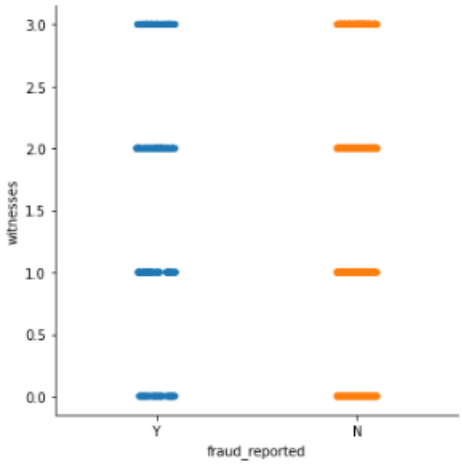


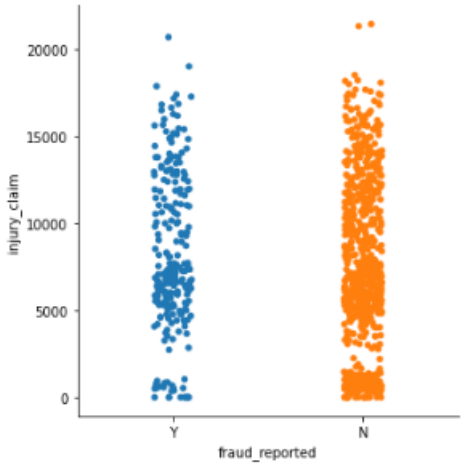


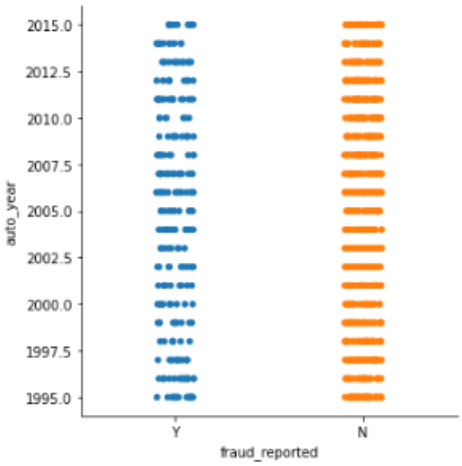


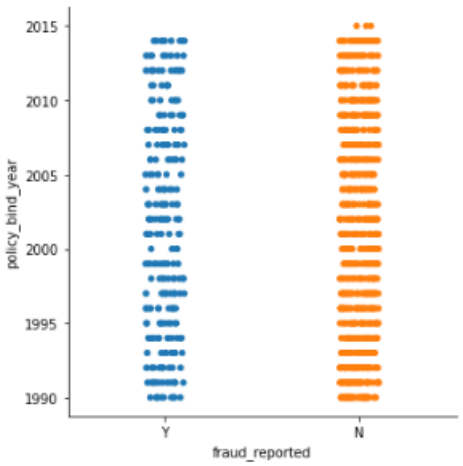

bodily_injuries
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Y
N
fraud_reported

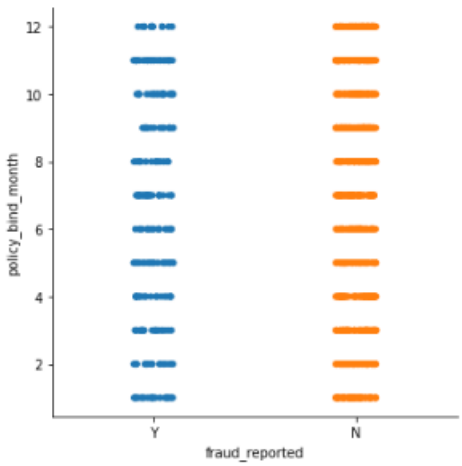

total_claim_amount
120000
100000
80000
60000
40000
20000
0
Y
N
fraud_reported

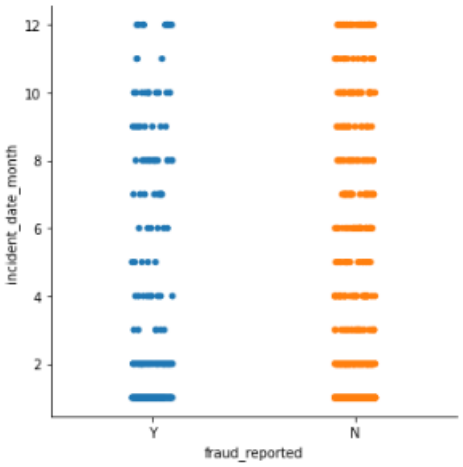

witnesses
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Y
N
fraud_reported


injury_claim
20000
15000
10000
5000
0
Y
N
fraud_reported


auto_year
2015.0
2012.5
2010.0
2007.5
2005.0
2002.5
2000.0
1997.5
1995.0
Y
N
fraud_reported


policy_bind_year
2015
2010
2005
2000
1995
1990
Y
N
fraud_reported


policy_bind_month
12
10
8
6
4
2
Y
N
fraud_reported

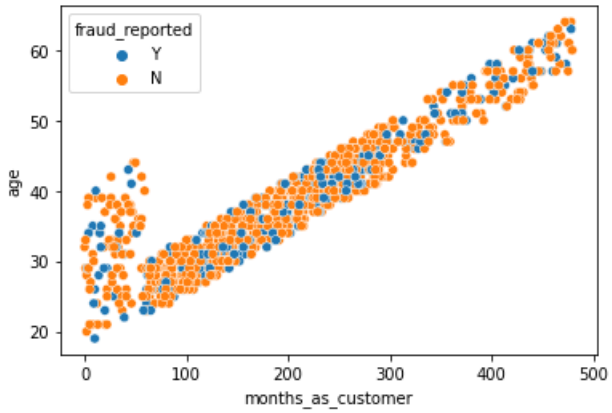

incident_date_month
12
10
8
6
4
2
Y
N
fraud_reported

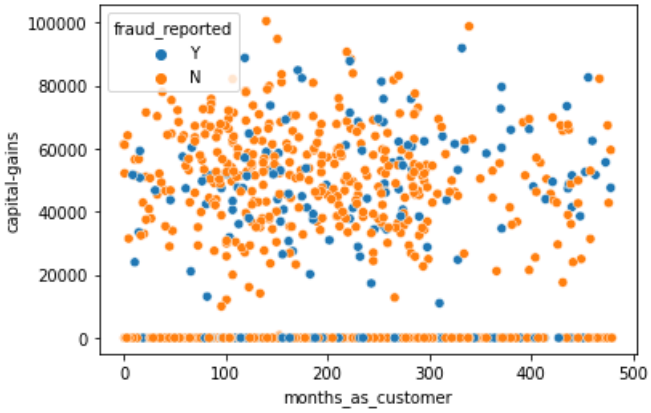
Observations: As per above graphs, we find that most of the dataset having fraud_reports N present in higher amounts.

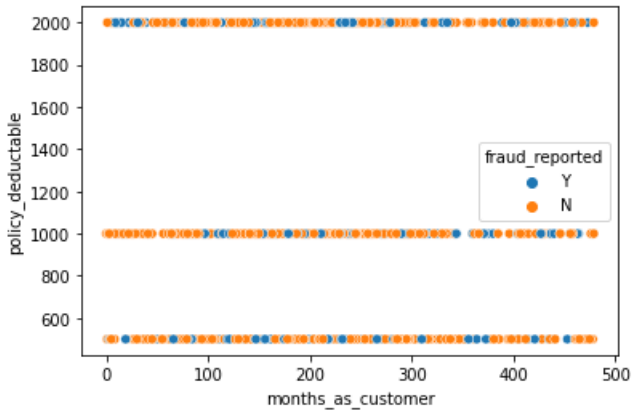
Bivariate Analysis of Numerical Dataset

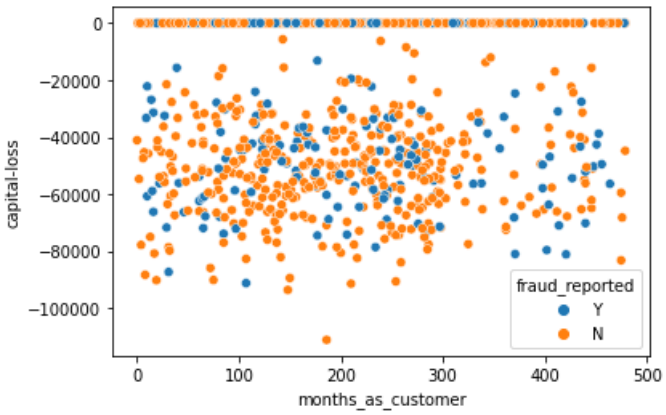
Code

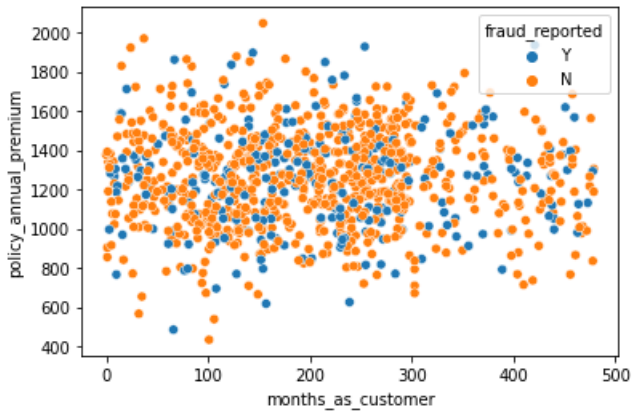
```
for i in range(len(df_num)):
    for j in df_num[i+1:]:
        plt.figure()
        sns.scatterplot(df[df_num[i]], df[j], hue = df['fraud_reported'])
```

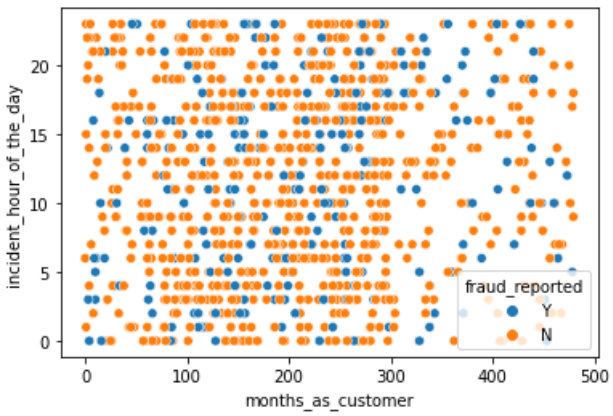


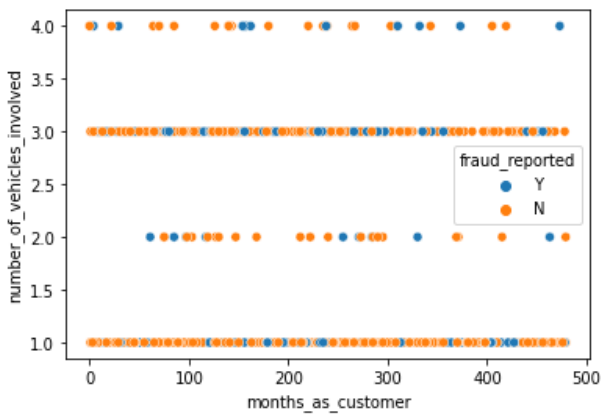


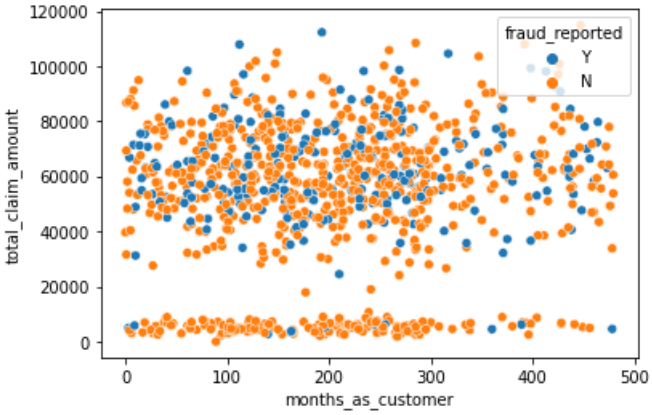


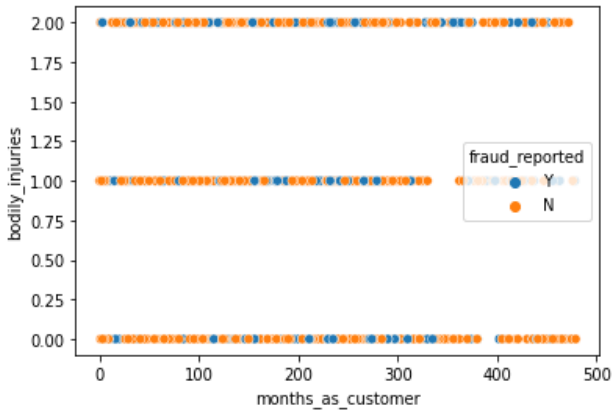


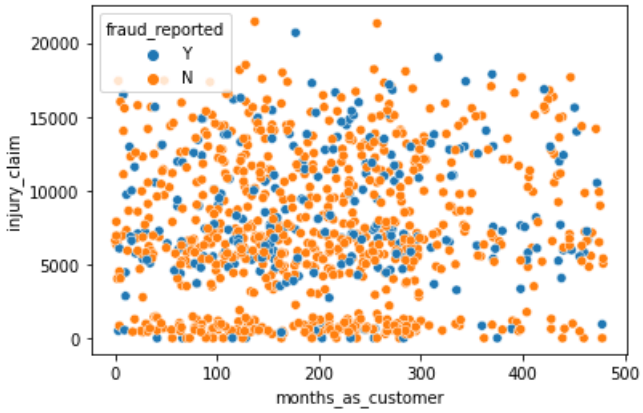


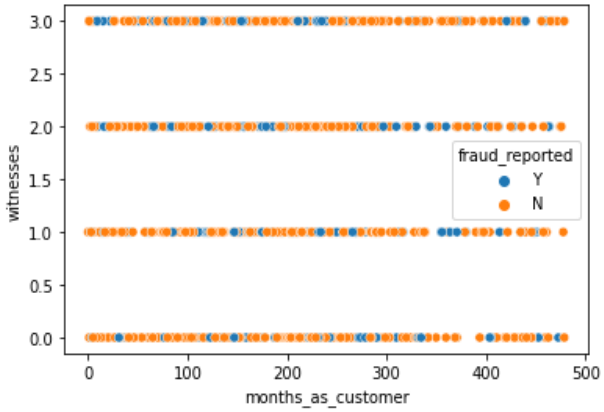


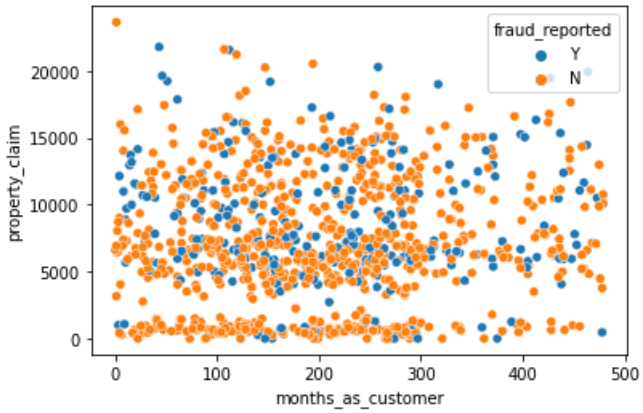

number_of_vehicles_involved
months_as_customer
fraud_reported
Y
N

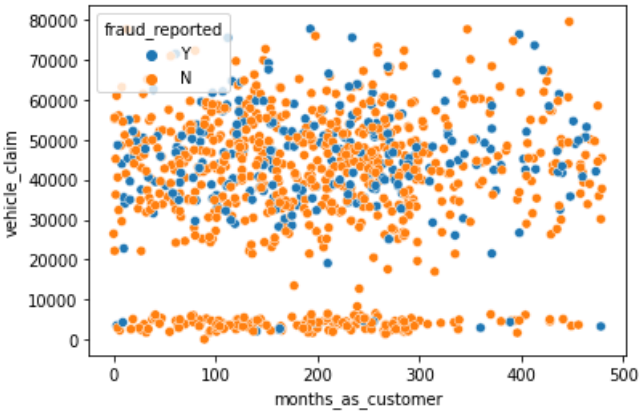

total_claim_amount
months_as_customer
fraud_reported
Y
N

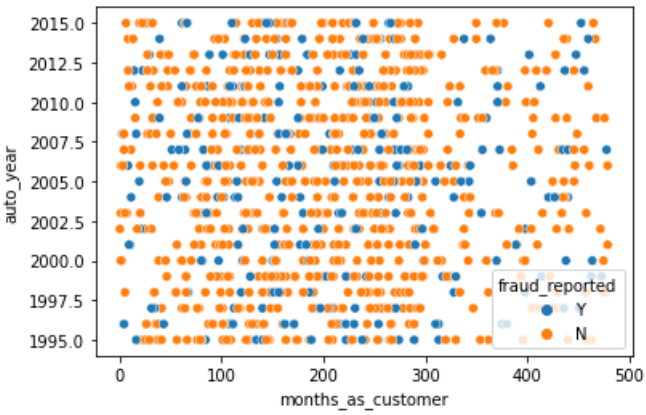

bodily_injuries
months_as_customer
fraud_reported
Y
N

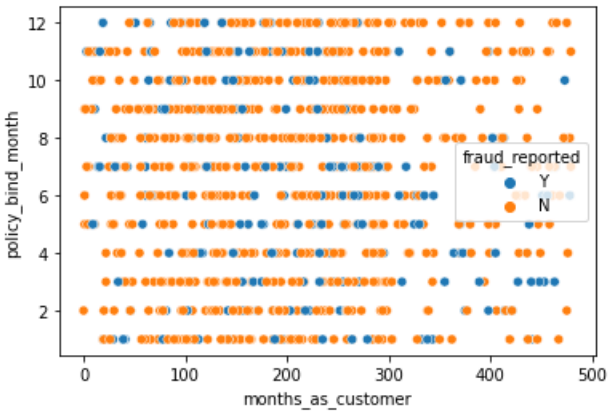

injury_claim
months_as_customer
fraud_reported
Y
N

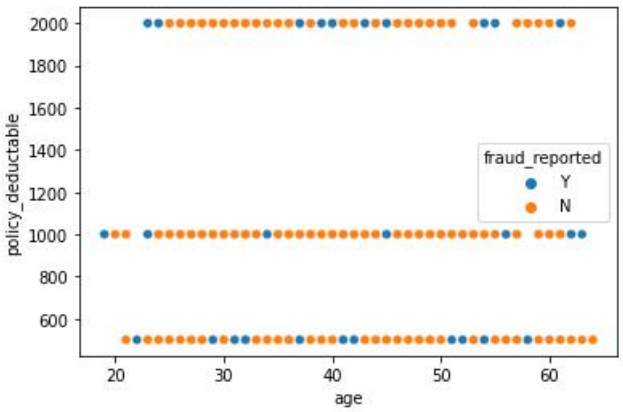

witnesses
months_as_customer
fraud_reported
Y
N


property_claim
months_as_customer
fraud_reported
Y
N

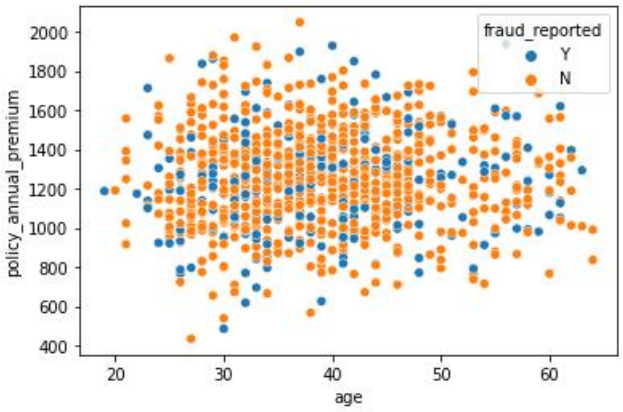

vehicle_claim
months_as_customer
fraud_reported
Y
N

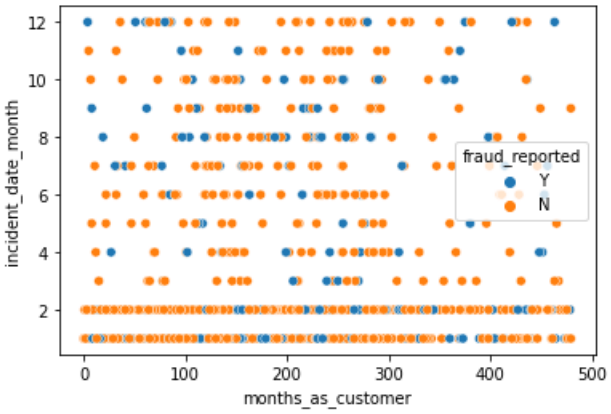

auto_year
months_as_customer
fraud_reported
Y
N

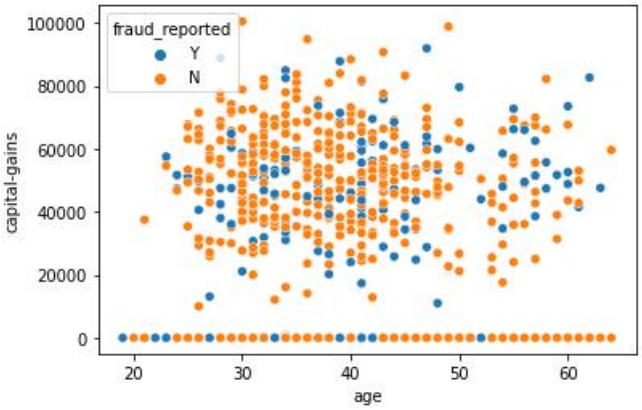

policy_bind_month
months_as_customer
fraud_reported
Y
N


policy_deductable
age
fraud_reported
Y
N

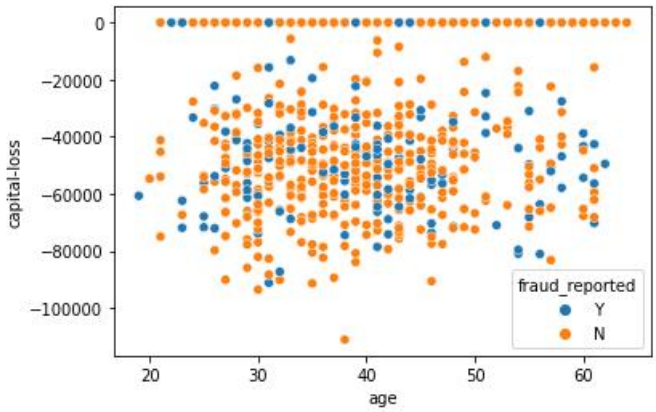

policy_bind_year
months_as_customer
fraud_reported
Y
N

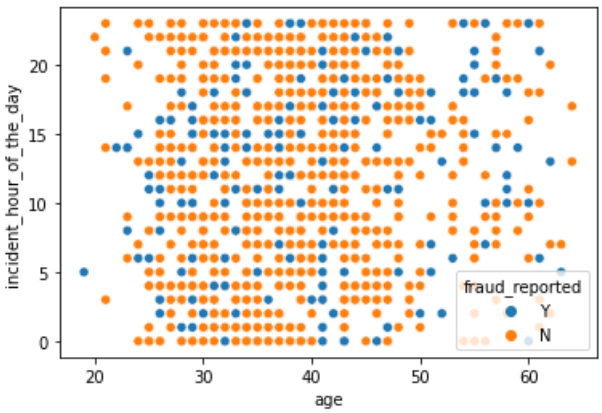

policy_annual_premium
age
fraud_reported
Y
N

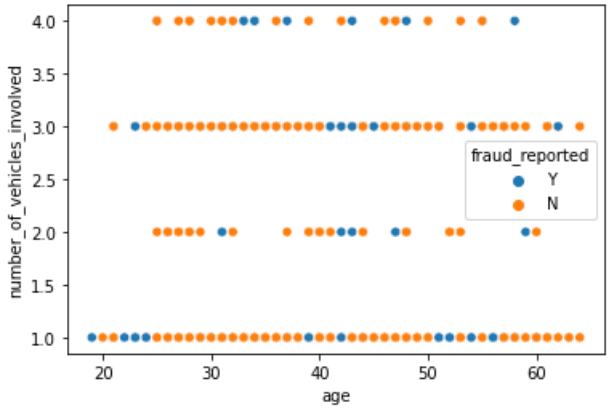

incident_date_month
months_as_customer
fraud_reported
Y
N

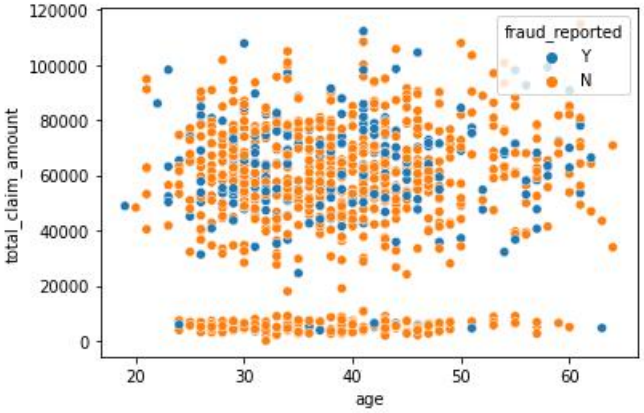

capital-gains
age
fraud_reported
Y
N

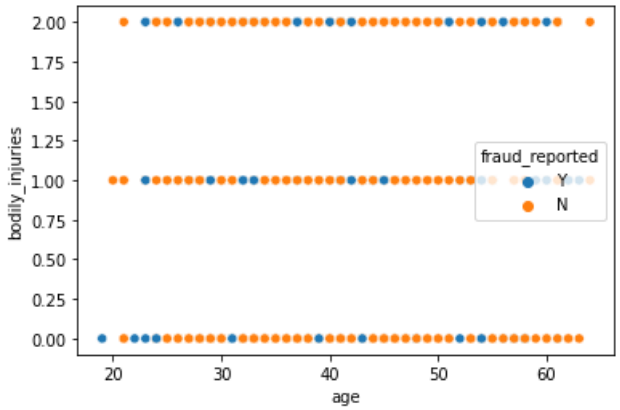
age

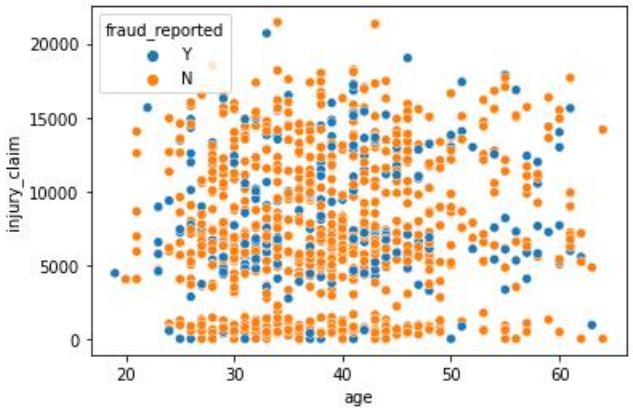

capital-loss
age
fraud_reported
Y
N

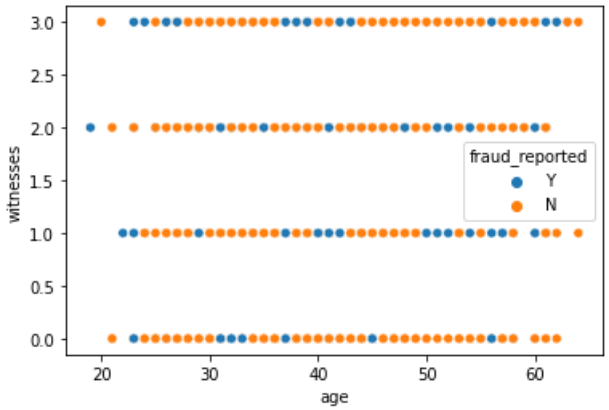

incident_hour_of_the_day
age
fraud_reported
Y
N

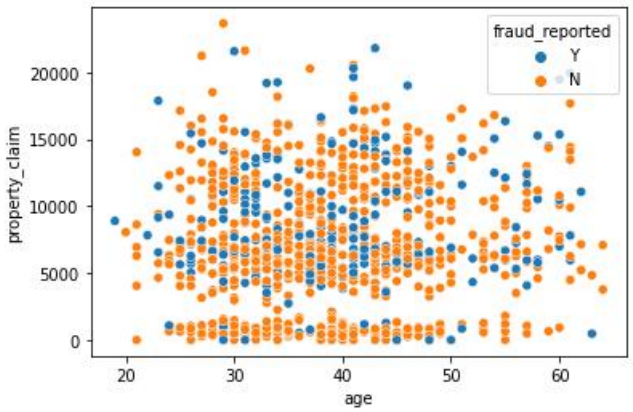

number_of_vehicles_involved
age
fraud_reported
Y
N

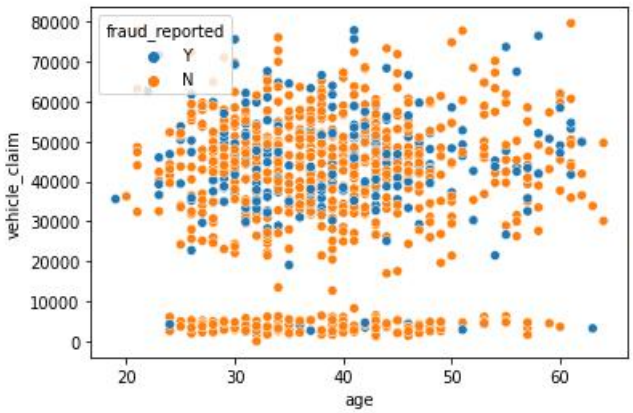

total_claim_amount
age
fraud_reported
Y
N

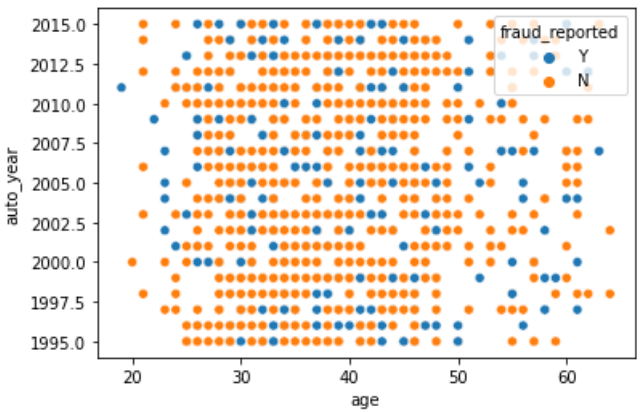

bodily_injuries
age
fraud_reported
Y
N

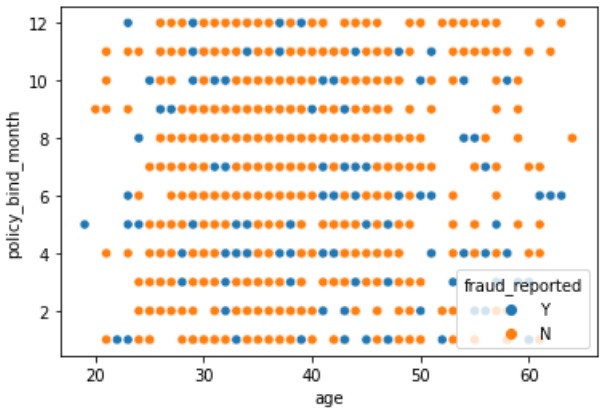

injury_claim
age
fraud_reported
Y
N

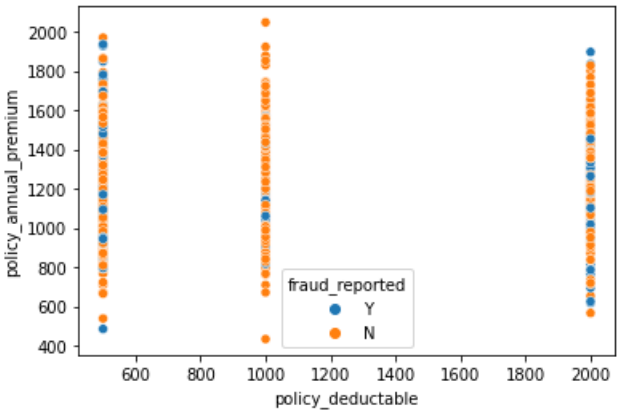

witnesses
age
fraud_reported
Y
N

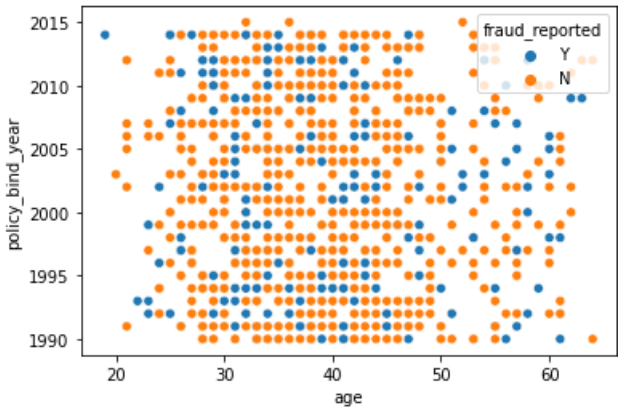

property_claim
age
fraud_reported
Y
N

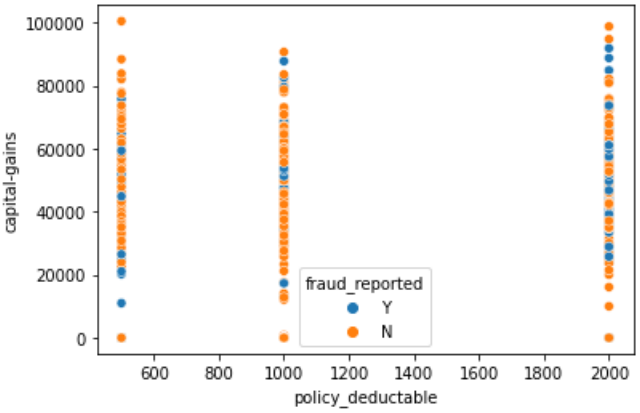

vehicle_claim
age
fraud_reported
Y
N

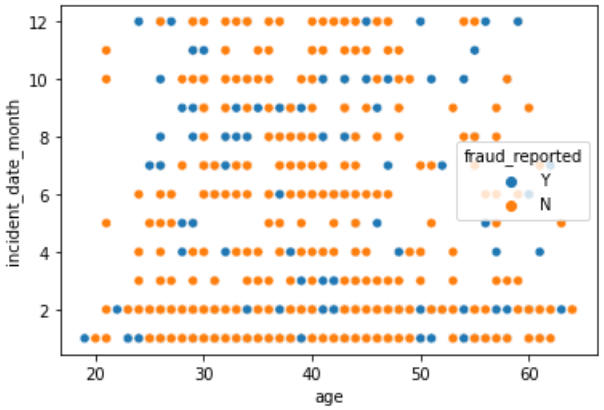

auto_year
age
fraud_reported
Y
N

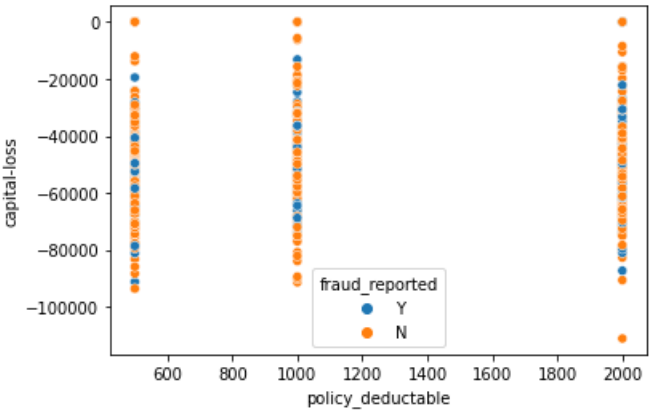

policy_bind_month
age
fraud_reported
Y
N

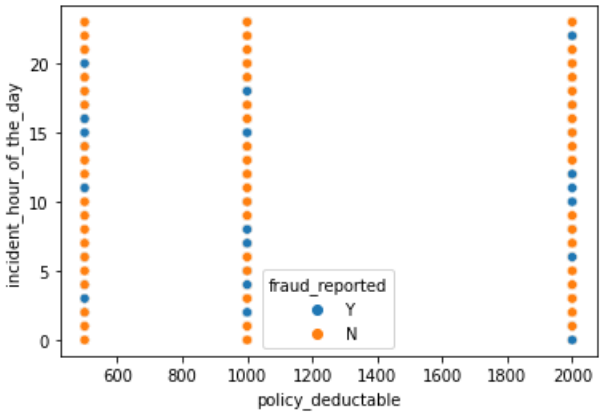

policy_annual_premium
policy_deductable
fraud_reported
Y
N

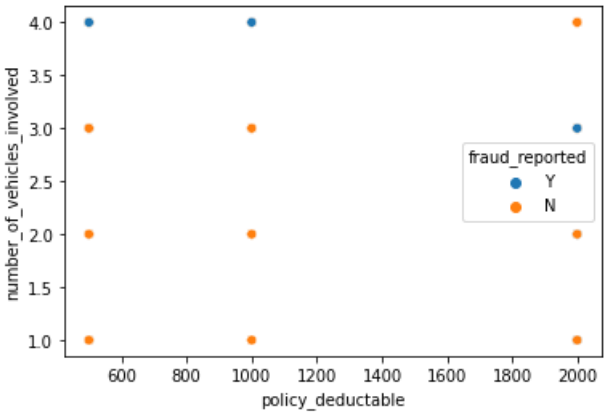

policy_bind_year
age
fraud_reported
Y
N

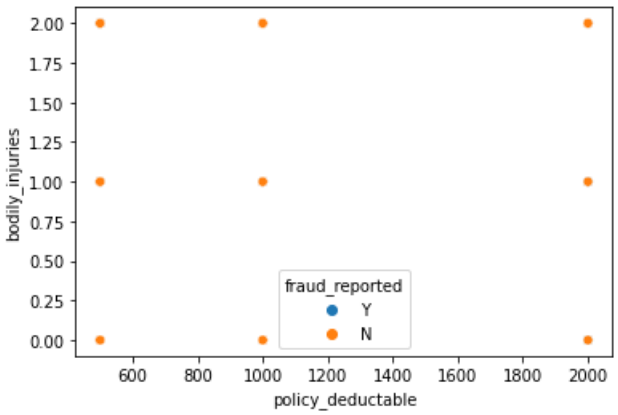

capital-gains
policy_deductable
fraud_reported
Y
N

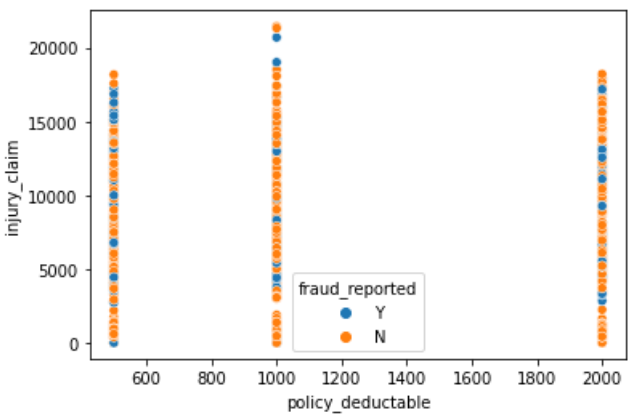

incident_date_month
age
fraud_reported
Y
N

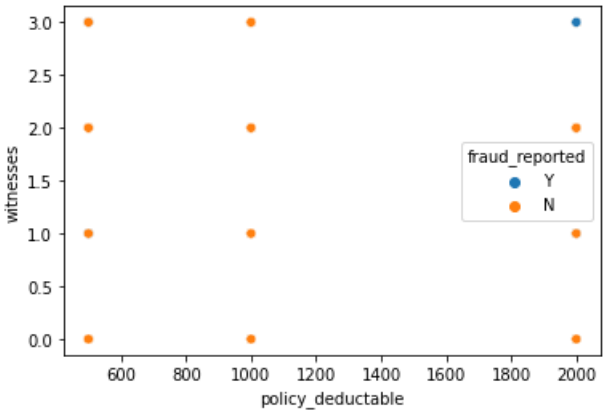

capital-loss
policy_deductable
fraud_reported
Y
N

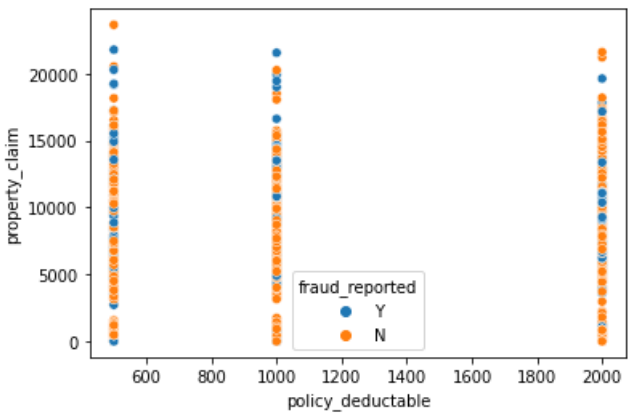

incident_hour_of_the_day
policy_deductable
fraud_reported
Y
N

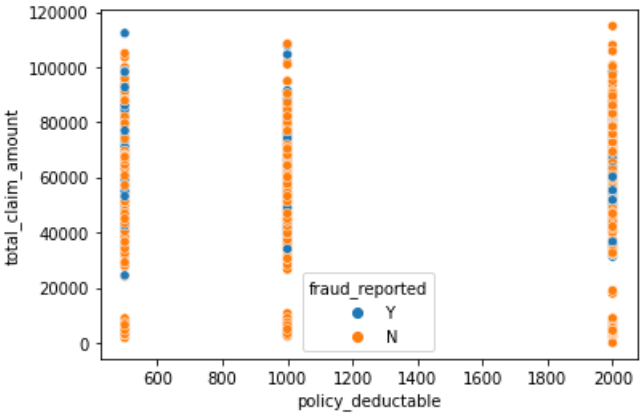

number_of_vehicles_involved
policy_deductable
fraud_reported
Y
N

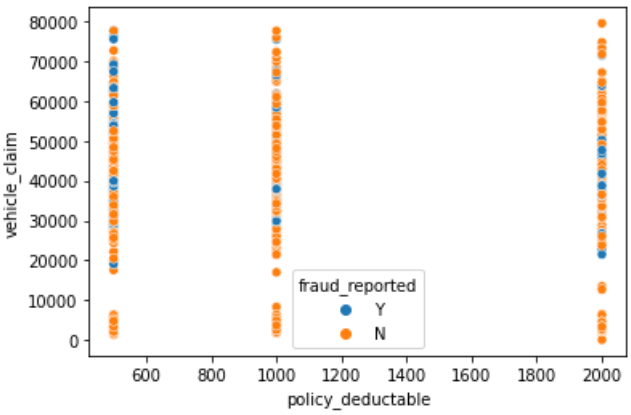

bodily_injuries
policy_deductable
fraud_reported
Y
N

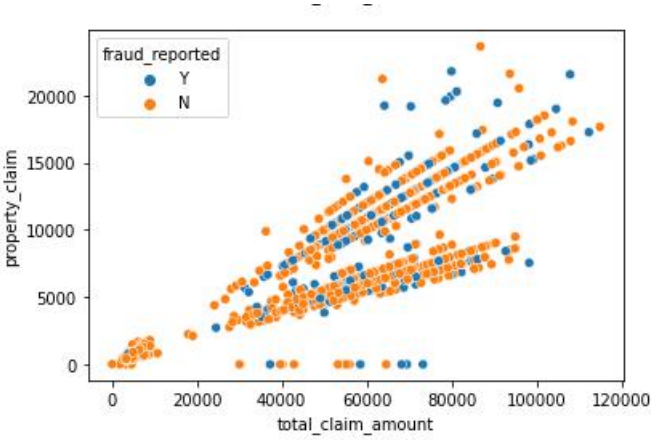

injury_claim
policy_deductable
fraud_reported
Y
N

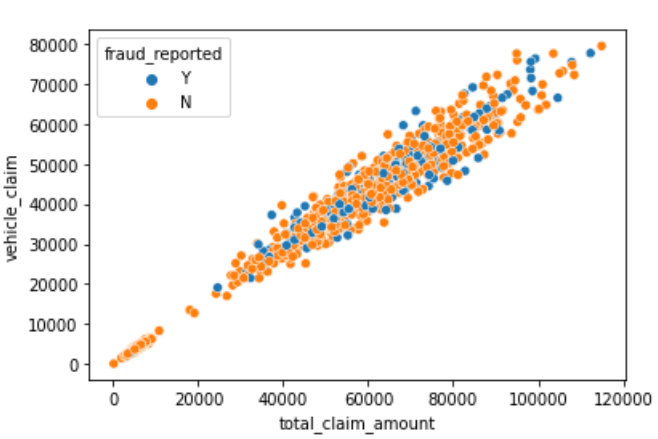

witnesses
policy_deductable
fraud_reported
Y
N


property_claim
policy_deductable
fraud_reported
Y
N

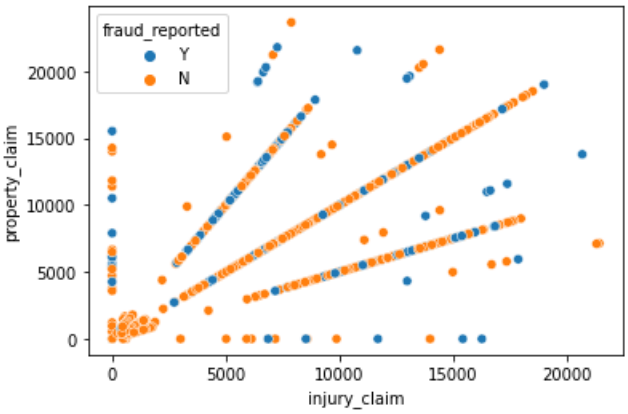

total_claim_amount
policy_deductable
fraud_reported
Y
N

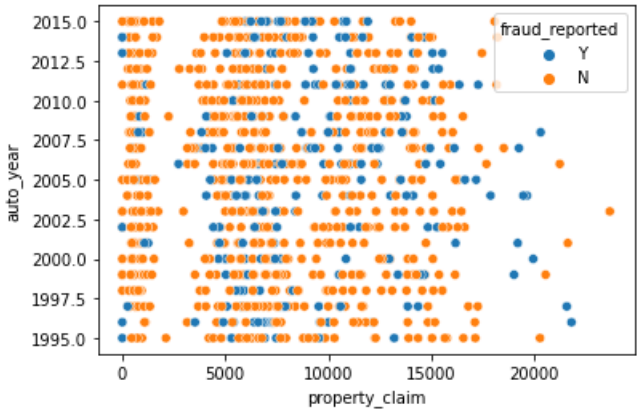

vehicle_claim
policy_deductable
fraud_reported
Y
N

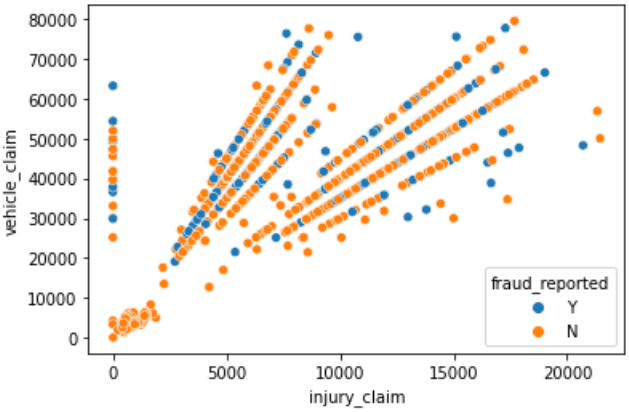

property_claim
total_claim_amount
fraud_reported
Y
N


vehicle_claim
total_claim_amount
fraud_reported
Y
N

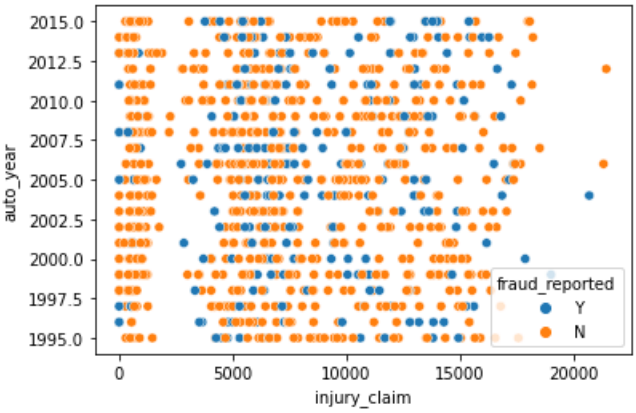

fraud_reported
Y
N
vehicle_claim
property_claim


fraud_reported
Y
N
property_claim
injury_claim

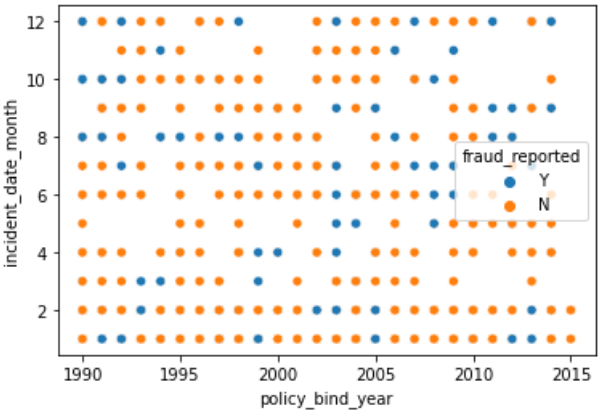

fraud_reported
Y
N
auto_year
property_claim


fraud_reported
Y
N
vehicle_claim
injury_claim


fraud_reported
Y
N
policy_bind_month
property_claim


fraud_reported
Y
N
auto_year
injury_claim


fraud_reported
Y
N
incident_date_month
policy_bind_month

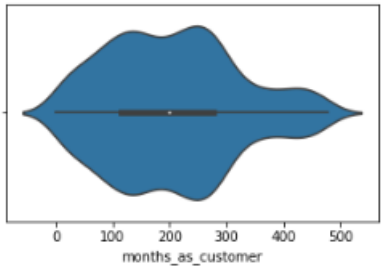

fraud_reported
Y
N
incident_date_month
policy_bind_year

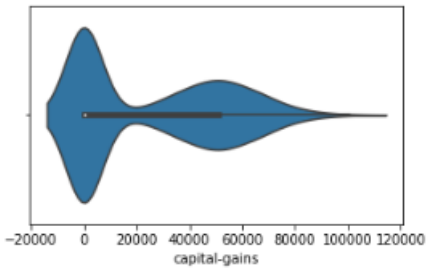
Observations:

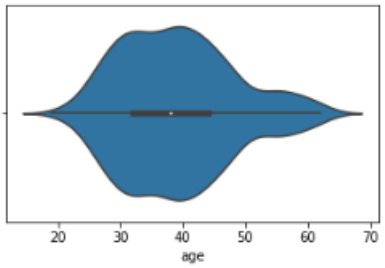
1. Inquiry_claim with total_claim_amount show linear increment accordingly.
2. Similarily property_claim and vehicle claim shows also linear increment ac- cording.
3. Vehicle_claim and property_claim shows linearly increament with respect to each other.
4. Age and month as customer are directly proportional to each others.
5. Rest graphs not shows liable details.

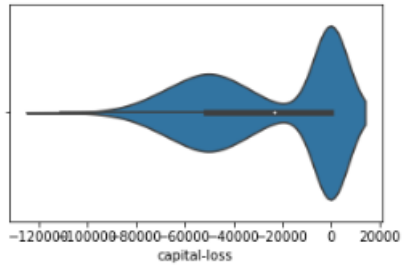
Violin Plot

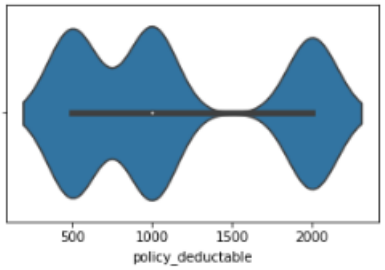
```
for i in df_num:
    plt.figure(figsize = (10,5))
    sns.violinplot(df[i] , data = df)


# checking spreading density of the data
```

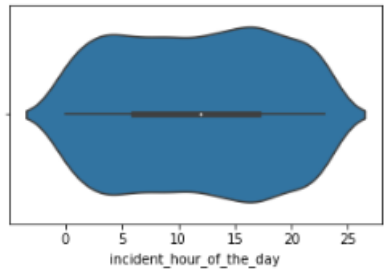


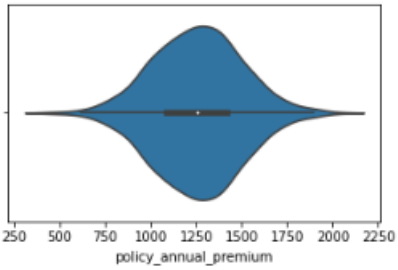


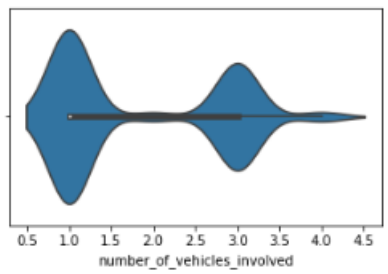


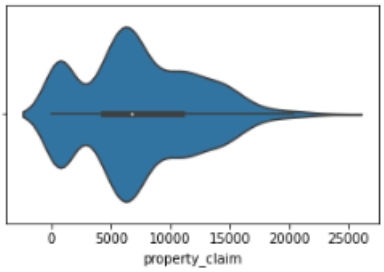


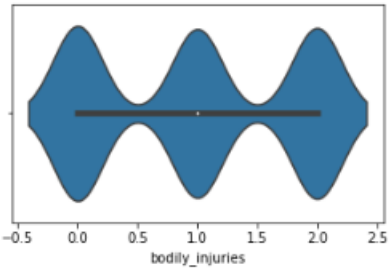


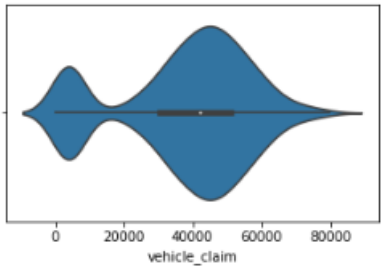


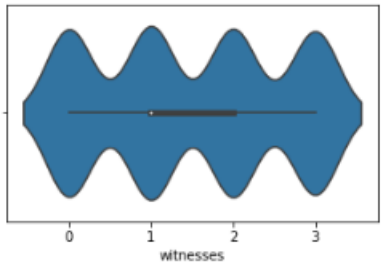


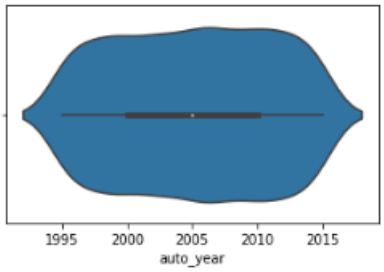


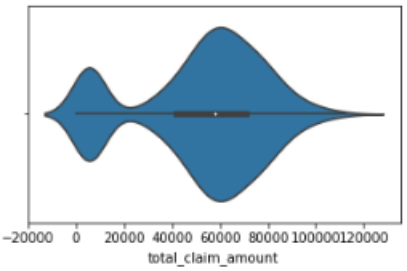

0
5000
10000
15000
20000
25000
property_claim

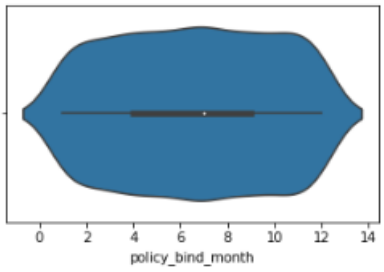

−0.5
0.0
0.5
1.0
1.5
2.0
2.5
bodily_injuries

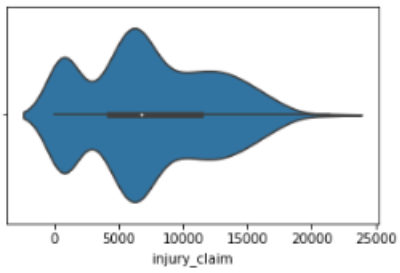

0
20000
40000
60000
80000
vehicle_claim

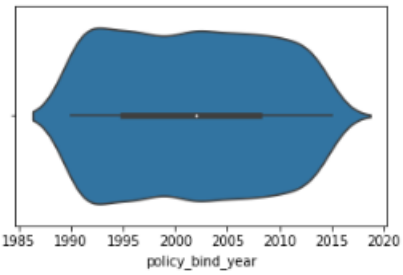

0
1
2
3
witnesses

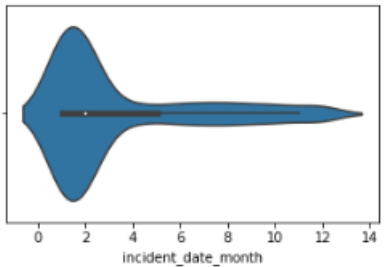

1995
2000
2005
2010
2015
auto_year


−20000
0
20000
40000
60000
80000
100000
120000
total_claim_amount


0
2
4
6
8
10
12
14
policy_bind_month


0
5000
10000
15000
20000
25000
injury_claim


1985
1990
1995
2000
2005
2010
2015
2020
policy_bind_year


0
2
4
6
8
10
12
14
incident_date_month

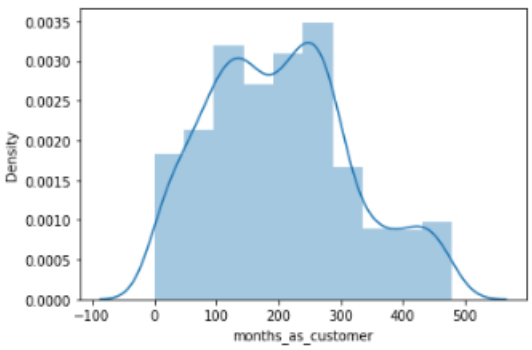
Observations:

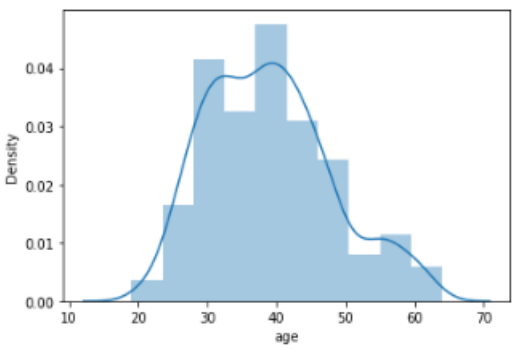
1. month_as_customer having density in the range of 0 to 500 but most of the customer lies in the range of 0 to 300 approx as per graph.
2. Age having density in the range of 20 to 70 but most of the customer lies in the range of 20 to 55 approx as per graph.
3. policy_deductable are divided into 3 parts of density with different different range as per above graph.
4. policy_annual_premium its density range lies between 250 to 2000 approx.
5. capital-gains As per graph most of the people didn't get any gains but a few people gains with the range of 20000 to 100000 approx.
6. capital-loss As per graph most of the people didn't get any loss but a few people loss with the range of -20000 to 600000 approx.
7. incident_hour_of the day shows common density pattern.
8. numbers of vehicle involved it is divided into two parts which are 1 and 3 as per graph.
9. bodily_injuries it is divided into three parts which are 0 to 2 approx as per graph.
10. witness it is divided into four parts which are 0 to 3 approx as per graph.
11. total_claim_amount most of the people claim amount lies in the density ranges of 20000 to 1000000 as per graph.
12. injury_claim most of the people claim amount lies in the density ranges of 0 to 20000 as per graph.
13. property_claim most of the people claim amount lies in the density ranges of 0 to 20000 as per graph.
14. vehicle_claim most of the claim amount lies in the range of 20000 to 80000.
15. auto_year, policy_bind_month and policy_bind_year show normal density ranges as per graph.
16. incident date month most of the incident lies in January to February as per density curve.

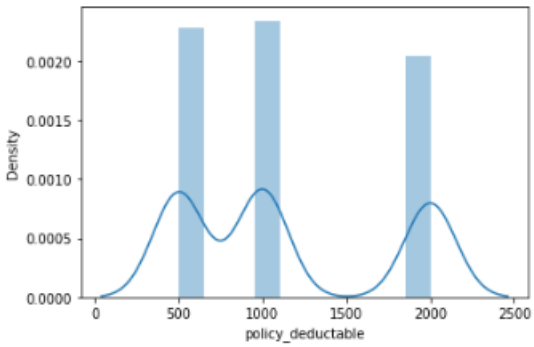
Checking Distribution of the Dataset

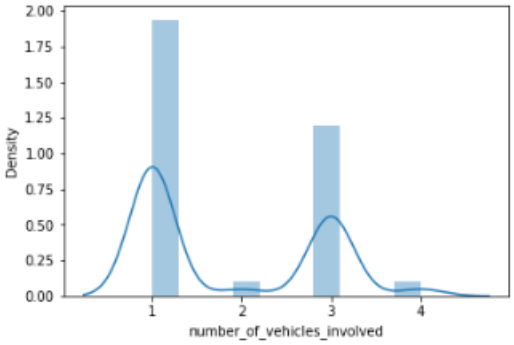
Distribution Plot

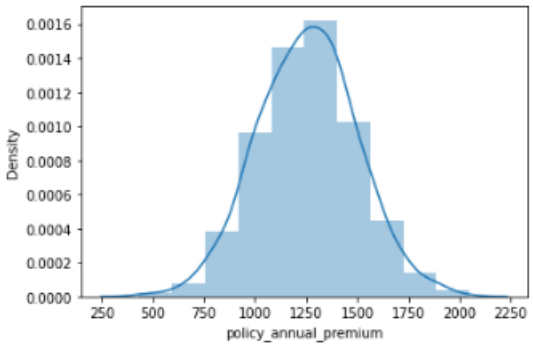
```
for i in df_num:
    plt.figure()
    sns.distplot(df[i], bins = 10)
```

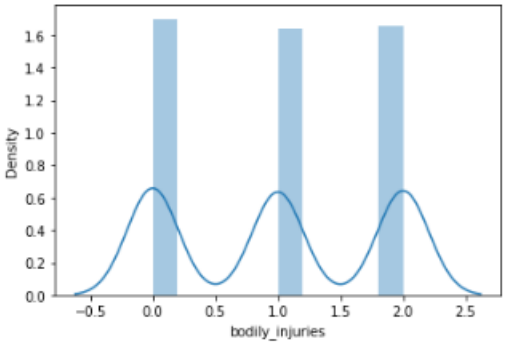


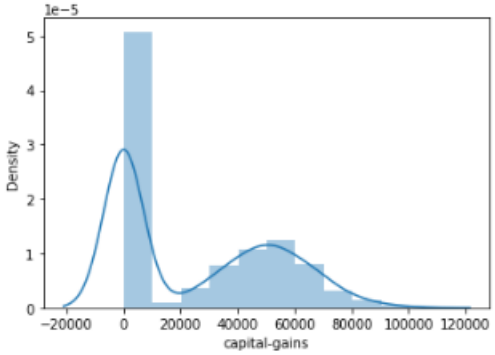


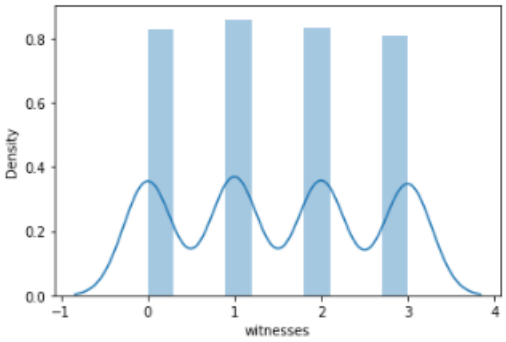

Density
0.0020
0.0015
0.0010
0.0005
0.0000
0
500
1000
1500
2000
2500
policy_deductable

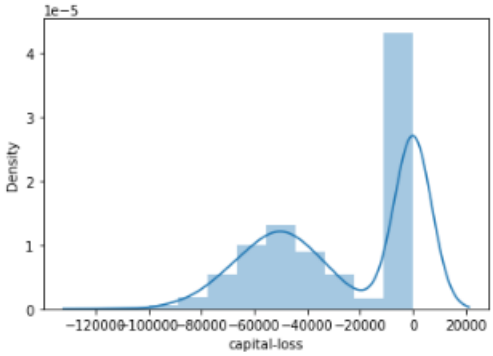

Density
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
1
2
3
4
number_of_vehicles_involved

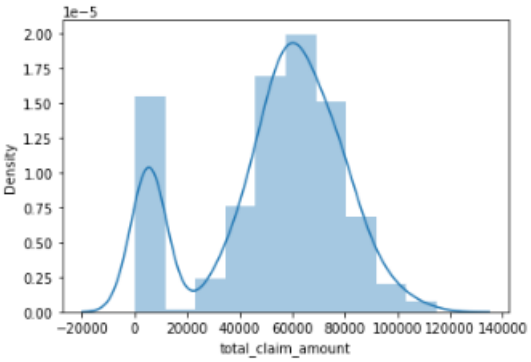

Density
0.0016
0.0014
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
250
500
750
1000
1250
1500
1750
2000
2250
policy_annual_premium

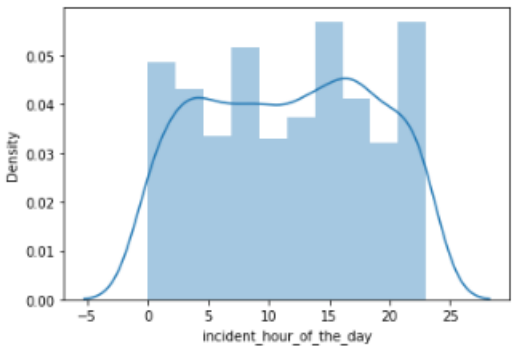

Density
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
bodily_injuries

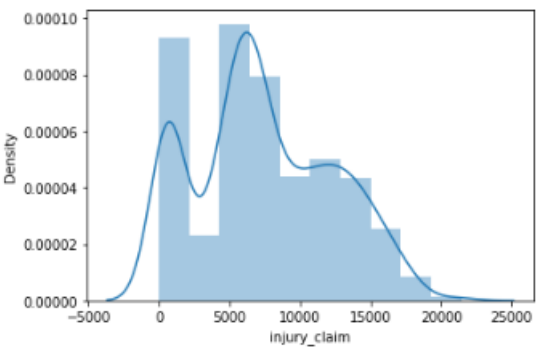

1e−5
Density
5
4
3
2
1
0
−20000
0
20000
40000
60000
80000
100000
120000
capital-gains

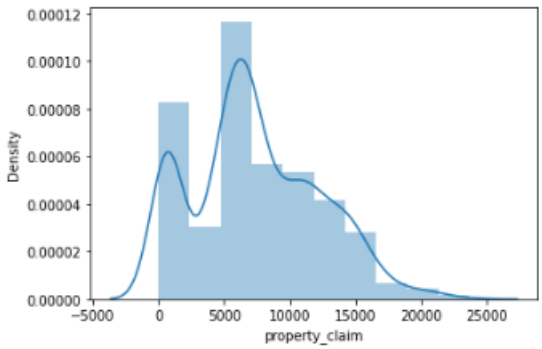

Density
0.8
0.6
0.4
0.2
0.0
−1
0
1
2
3
4
witnesses

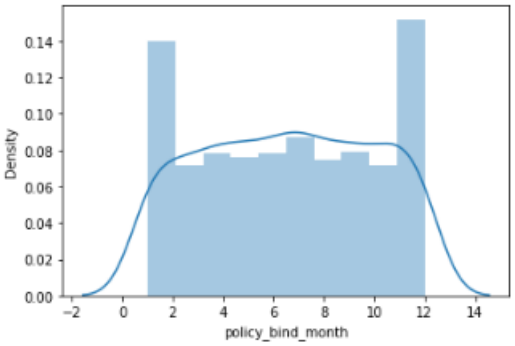

1e−5
Density
4
3
2
1
0
−120000
−100000
−80000
−60000
−40000
−20000
0
20000
capital-loss

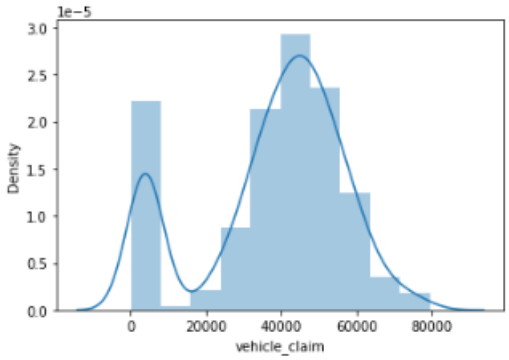

1e−5
Density
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−20000
0
20000
40000
60000
80000
100000
120000
140000
total_claim_amount

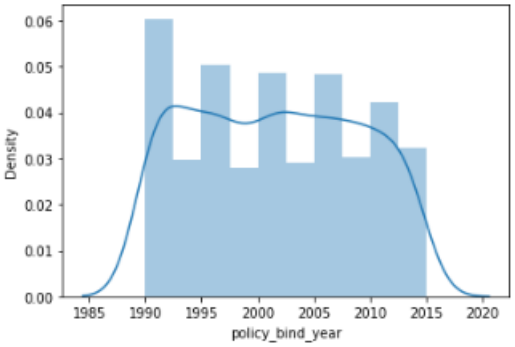

Density
0.05
0.04
0.03
0.02
0.01
0.00
−5
0
5
10
15
20
25
incident_hour_of_the_day

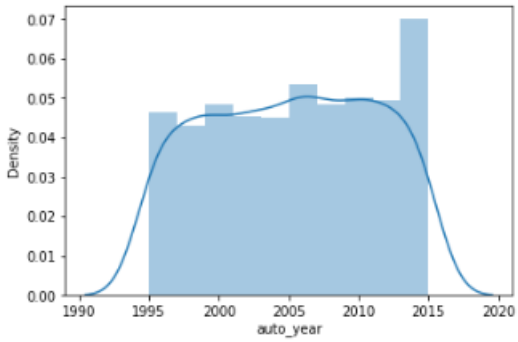

Density
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
−5000
0
5000
10000
15000
20000
25000
injury_claim

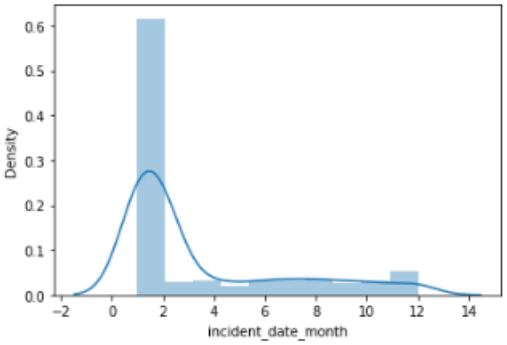








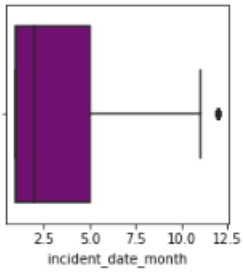


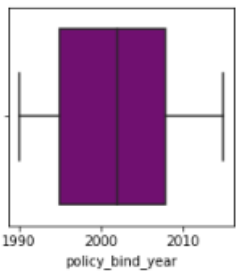


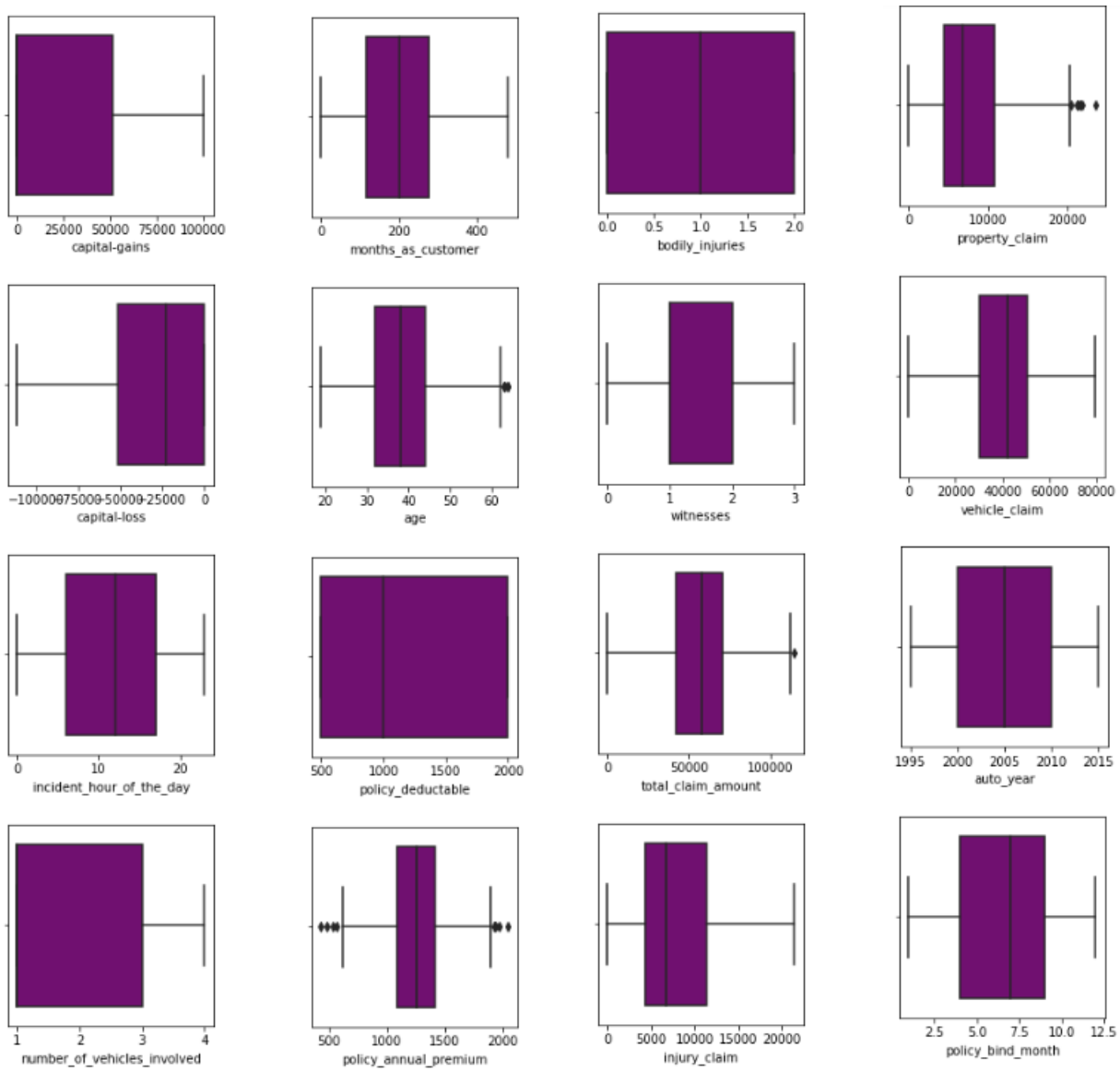
Observations: Distribution plot behave same as density plot with bins.

Checking Outliers

```
for i in df_num:
    plt.figure(figsize = (3,3))
    sns.boxplot(df[i], orient='v', color='purple')
```







Observations:

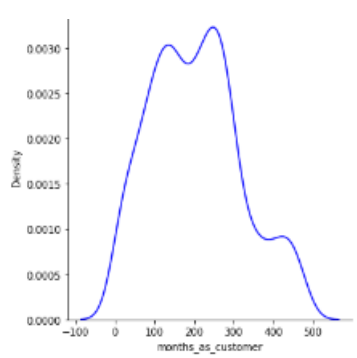
1. age: age column dataset having a neglible amount of outliers as per above graph.
2. policy_annual_premimum having outliers in both criteria with fraud reported yes and no.
3. totol _claim_amount having outliers in the dataset of fraud reported.
4. property_claim having outliers in both criteria.
5. incident_date_month having outliers only in the dataset of a single month.

Checking Skewness

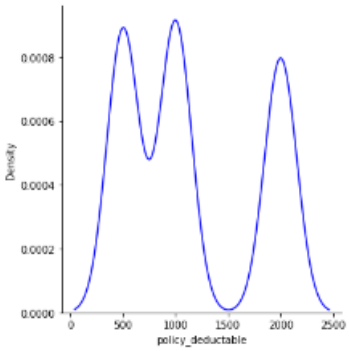
```
df[df_num].skew()
```

```
months_as_customer            0.362177
age                           0.478988
policy_deductable             0.477887
policy_annual_premium         0.004402
capital-gains                 0.478850
capital-loss                 -0.391472
incident_hour_of_the_day     -0.035584
number_of_vehicles_involved   0.502664
bodily_injuries               0.014777
witnesses                     0.019636
total_claim_amount           -0.594582
injury_claim                  0.264811
property_claim                0.378169
vehicle_claim                -0.621098
auto_year                    -0.048289
policy_bind_month            -0.029321
policy_bind_year              0.052511
incident_date_month           1.375894
dtype: float64
```

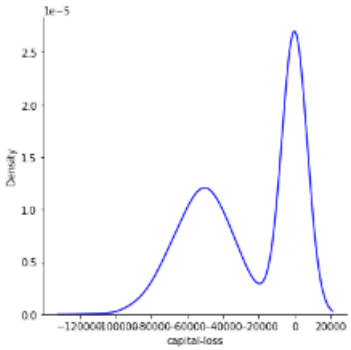
```
for i in df_num:
    plt.figure(figsize = (1,0.5))
    sns.displot(df[i], kind = 'kde', color = 'blue',)
```



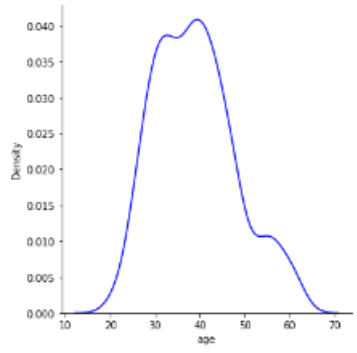
<Figure size 36x36 with 0 Axes>



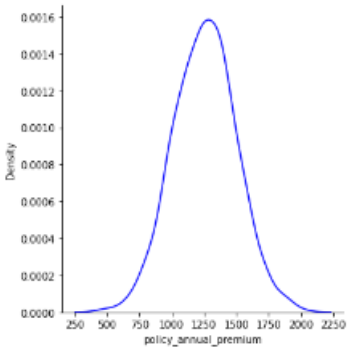
<Figure size 36x36 with 0 Axes>



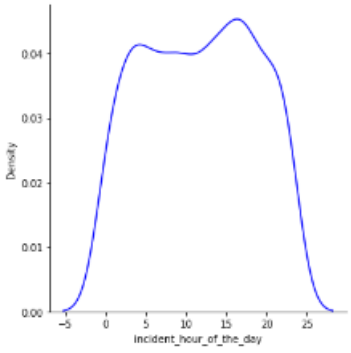
<Figure size 36x36 with 0 Axes>



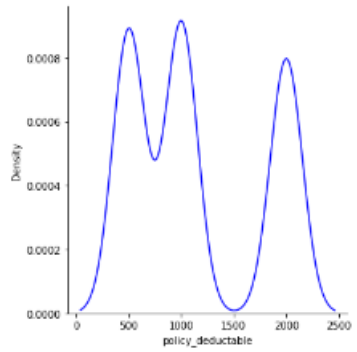
<Figure size 36x36 with 0 Axes>

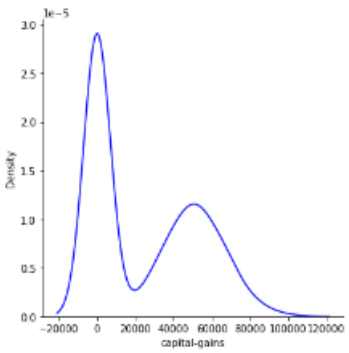


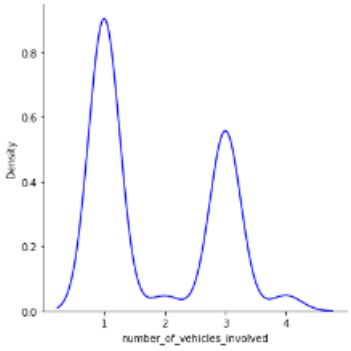
<Figure size 36x36 with 0 Axes>

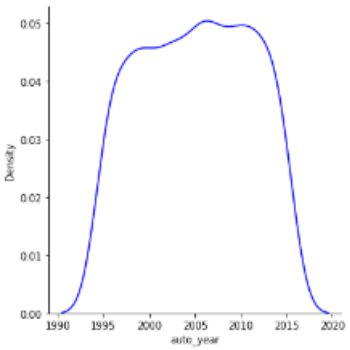


<Figure size 36x36 with 0 Axes>

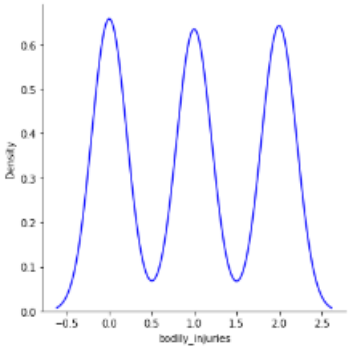




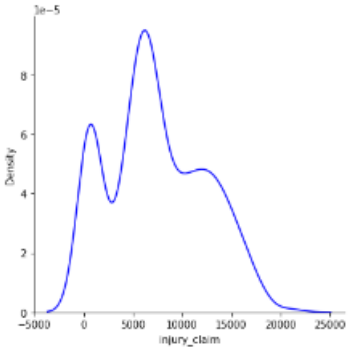




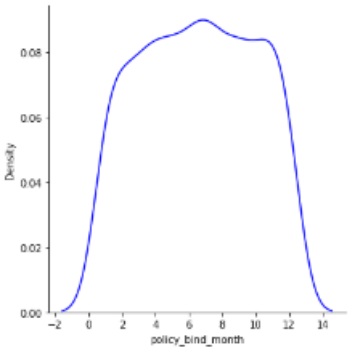
<Figure size 36x36 with 0 Axes>



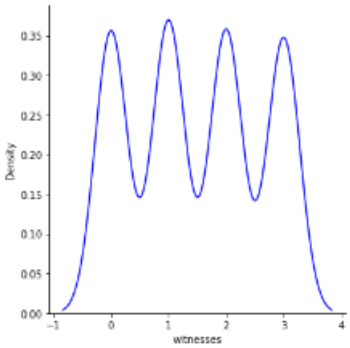
<Figure size 36x36 with 0 Axes>



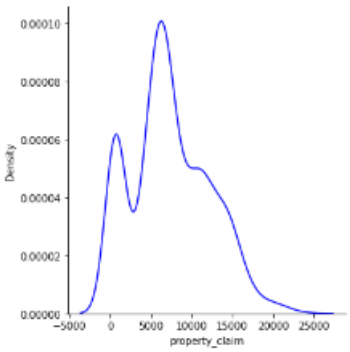
<Figure size 36x36 with 0 Axes>



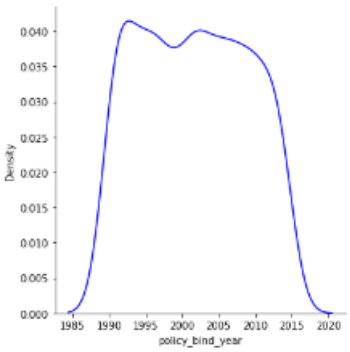
<Figure size 36x36 with 0 Axes>

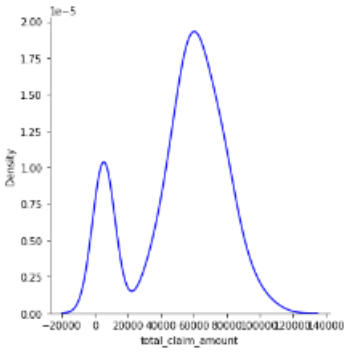


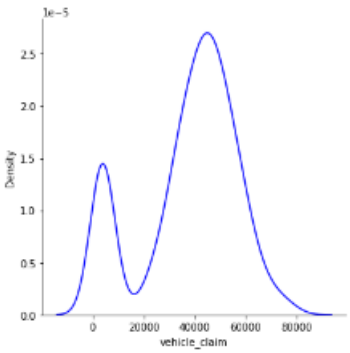
<Figure size 36x36 with 0 Axes>

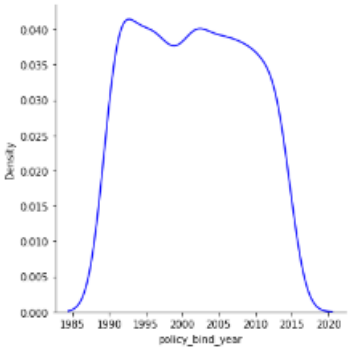


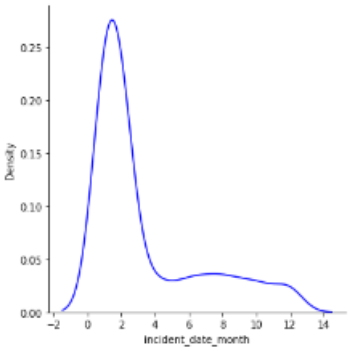
<Figure size 36x36 with 0 Axes>









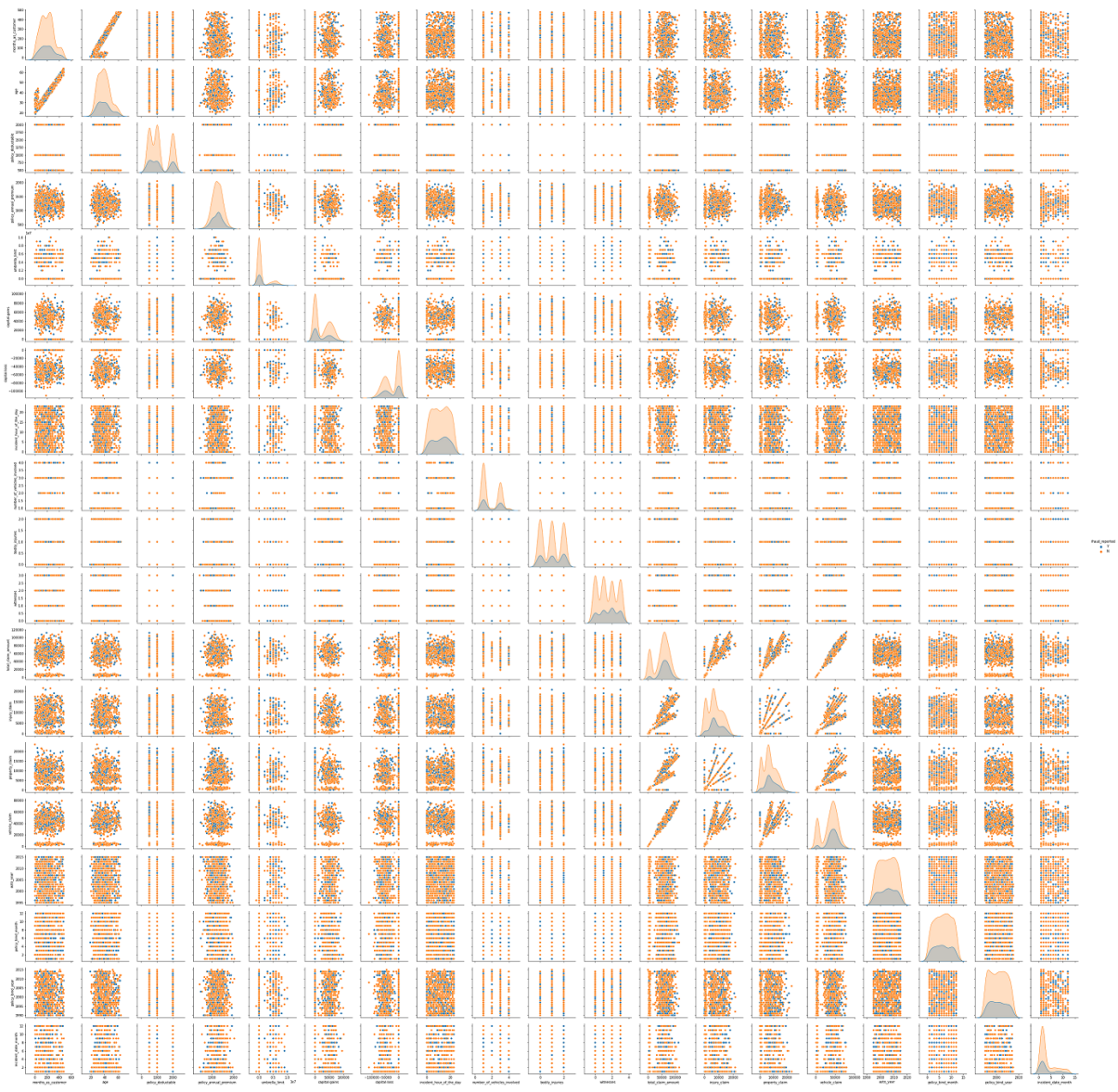


Observations:

Month as customer, age, policy_deductable, capital gains , number of vehicle involved, injury claim, property claim and incident date month shows right skewness towards the dataset rest of incident hour of the day, capital loss, total claim amount vehicle claim, auto year, policy bind month show left skewness towards the dataset.

Checking Overall of the Dataset by Pair Plot

```
plt.figure(figsize = (15,15))
sns.pairplot(df, hue = 'fraud_reported')
```



Observations: Measured all analysis above are similar to pair plot.

Pre- Processing Pipelines

Label Encoder for target variable

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

```
df['fraud_reported'] = le.fit_transform(df['fraud_reported'])
df.head(2)
```

For making similar predictor variable we are using Ordinal Encoder

```
from sklearn.preprocessing import OrdinalEncoder
onc = OrdinalEncoder()
```

```
for i in df.select_dtypes(include = 'object').columns:
    df[i] = onc.fit_transform(df[i].values.reshape(-1,1))
```

Removing Outliers from the Dataset

1. **Z score Method**

```
from scipy.stats import zscore

z = np.abs(zscore(df))

# taking thresold value = 3
#np.where(z>3)

df_z = df[(z<3).all(axis = 1)]
```

2. Interquartile Method

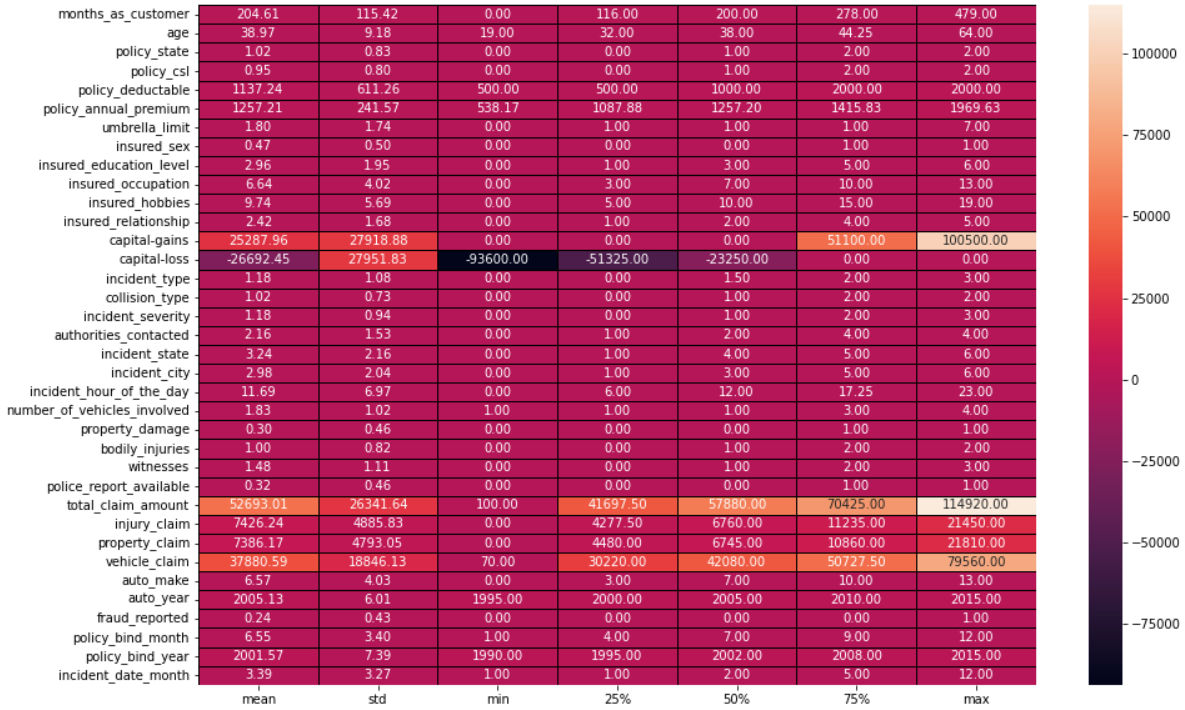
```
Q1 = df.quantile(.25)
Q3 = df.quantile(.75)
IQR = Q3 - Q1
```

```
df_IQR = df[~((df<(Q1 - 1.5*IQR)) | (df>(Q3 + 1.5*IQR))).any(axis = 1)]
```

From above two methods of removing outliers we find that Z score give lesser data loss with respect to inter quartile method hence we are selecting Z score method for removing outliers.

Describe

```
plt.figure(figsize = (15,10))
sns.heatmap(df_z.describe()[1:].transpose(), annot = True, linecolor='black', linewidths=0.5, fmt = '.2f')
```

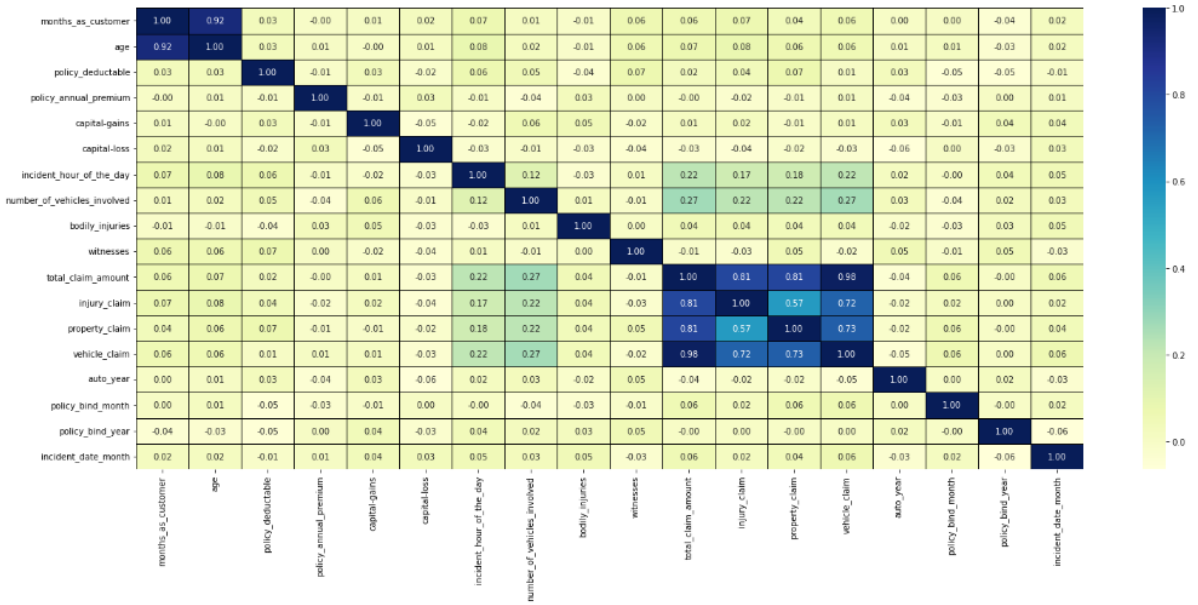


From above code we find that total claim amount , vehicle claim, capital gains values are much higher than its dataset value from this we conclude that this column dataset probably having outliers.

Checking Correlation of the Dataset

```
plt.figure(figsize = (25,10))
sns.heatmap(df_z[df_num].corr(), annot = True, linecolor = 'black', linewidths = 1, fmt = '.2f', cmap="YlGnBu" )
```

```
<AxesSubplot:>
```

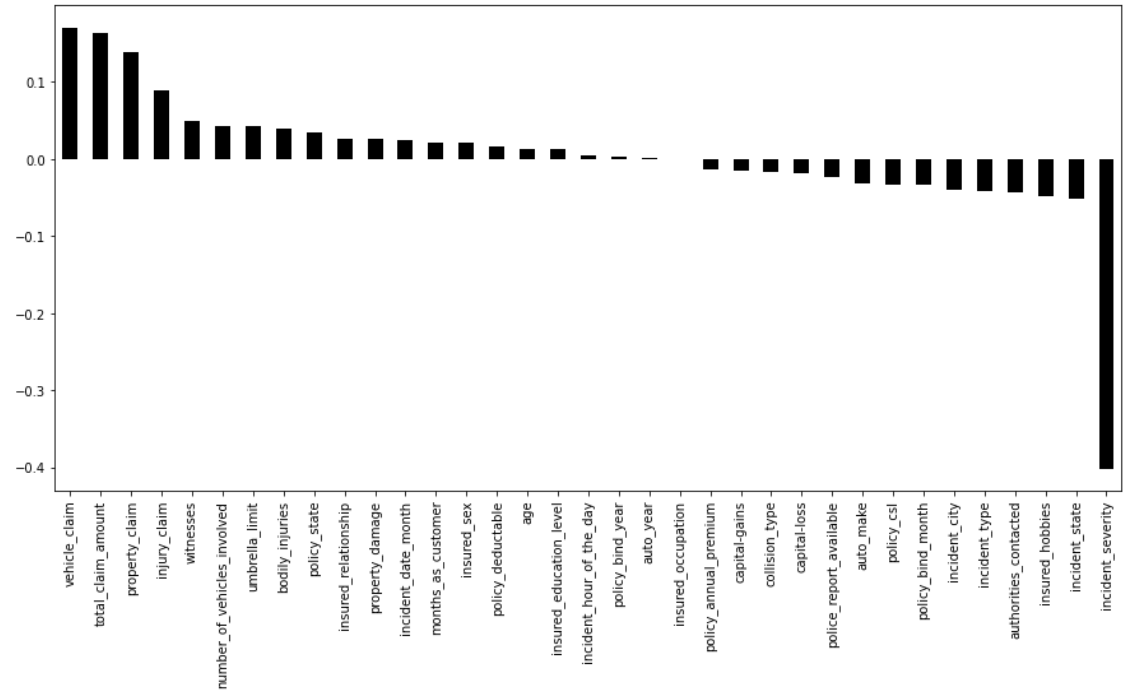


Observations: total claim amount, injury claim, property claim and vehicle claim shows very high correlation with each other as per graph.

Checking Positive and Negative Correlation

```
plt.figure(figsize = (15,7))
df_z.corr()['fraud_reported'].sort_values(ascending = False).drop('fraud_reported').plot(kind = 'bar', color = 'k')
```

```
<AxesSubplot:>
```



Observations:

'vehicle_claim', 'total_claim_amount','property_claim', 'injury_claim', 'witnesses', 'umbrella_limit', 'number_of_vehicles_involved', 'bodily_injuries', 'policy_state', 'insured_relationship', 'property_damage', 'incident_date_month','months_as_customer', 'insured_sex', 'policy_deductable', 'age', 'insured_education_level', 'incident_hour_of_the_day', 'policy_bind_year', 'auto_year' **shows positive correlation with respect to fraud reported** while rest of dataset shows **negative correlation.**

Dividing data for feature selection

```
x = df_z.drop('fraud_reported', axis = 1)
y = df_z['fraud_reported']
```

Using Get Dummies Method to prevent overfitting of Model as Model having so many columns with Multiple Unique Values

```
x = pd.get_dummies(data = x, columns = ['policy_state', 'policy_csl', 'umbrella_limit', 'insured_sex',
       'insured_education_level', 'insured_occupation', 'insured_hobbies',
       'insured_relationship', 'incident_type', 'collision_type',
       'incident_severity', 'authorities_contacted', 'incident_state',
       'incident_city', 'property_damage', 'police_report_available',
       'auto_make'])
```

Removing Skewness

```
skewed_columns = ['months_as_customer', 'age', 'policy_deductable',
       'policy_annual_premium',
       'capital-gains', 'capital-loss', 'incident_hour_of_the_day',
       'number_of_vehicles_involved', 'bodily_injuries', 'witnesses',
       'total_claim_amount', 'injury_claim', 'property_claim', 'vehicle_claim',]  #removing date columns like year, month
```

Using Power Transformer method to remove skewness

```
from sklearn.preprocessing import PowerTransformer
pw = PowerTransformer()
```

```
x[skewed_columns] = pw.fit_transform(x[skewed_columns])
x[skewed_columns].skew()  # skewness done on main dataframe
```

Checking Mutlicollinearity

```
import statsmodels.api as sm
from scipy import stats
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

```
def calc_vif(x):

    vif = pd.DataFrame()
    vif['Variance'] = x.columns
    vif["VIF Factor"] = [variance_inflation_factor(x.values, i) for i in range(x.shape[1])]
    return vif
```

```
calc_vif(x[skewed_columns])  # checking VIF of numerical columns
```

	Variance	VIF Factor
0	months_as_customer	4.677760
1	age	4.692092
2	policy_deductable	1.024122
3	policy_annual_premium	1.012786
4	capital-gains	1.011828
5	capital-loss	1.009981
6	incident_hour_of_the_day	1.068954
7	number_of_vehicles_involved	1.130874
8	bodily_injuries	1.011551
9	witnesses	1.021387
10	total_claim_amount	222.887129
11	injury_claim	10.150957
12	property_claim	10.319029
13	vehicle_claim	112.761998

As we can see that total claim amount and vehicle claim show very high vif hence we need to drop it. But due to less number of dataset we are dropping it one by one after checking observations.

In [117]:

```
x.drop('total_claim_amount', inplace = True, axis = 1)
```

Using SMOTE to balance the dataset

```
from imblearn.over_sampling import SMOTE
sm = SMOTE()
```

```
x, y = sm.fit_resample(x, y)
```

Standard Scalling

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
```

```
scaler_columns = ['months_as_customer',
 'age',
 'policy_deductable',
 'policy_annual_premium',
 'capital-gains',
 'capital-loss',
 'incident_hour_of_the_day',
 'number_of_vehicles_involved',
 'bodily_injuries',
 'witnesses',
 'injury_claim',
 'property_claim',
 'vehicle_claim']  # taking numerical columns
```

```
x[scaler_columns] = sc.fit_transform(x[scaler_columns])
```

```
x = pd.DataFrame(data = x, columns = x.columns)
```

Building Machine Learnings Models

- **Split the data into train and test**

Both the dependent variable and independent variable was split into train and test data (70% train and 30% test)

- Four different types of model classifiers were used in this project for finding best accuracy score:

- Logistics Regression
- GaussianNB
- Support Vector Classifier
- AdaBoostClassifier

Model 1 Logistics Regression

```
best_model.final_model(x, y, model, 69)
```

```
At random state 69 model giving best accuracy score

Training accuracy:-  87.64478764478764
Testing accuracy:-  88.06306306306307


--------------------------------------
Confusion Matrix
[[208  22]
 [ 31 183]]


Classification Report
              precision    recall  f1-score   support

           0       0.87      0.90      0.89       230
           1       0.89      0.86      0.87       214

    accuracy                           0.88       444
   macro avg       0.88      0.88      0.88       444
weighted avg       0.88      0.88      0.88       444
```

Model 2 GaussianNB

```
best_model.final_model(x, y, model, 144)
```

```
At random state 144 model giving best accuracy score

Training accuracy:-  87.45173745173746
Testing accuracy:-  88.06306306306307


--------------------------------------
Confusion Matrix
[[226  49]
 [  4 165]]


Classification Report
              precision    recall  f1-score   support

           0       0.98      0.82      0.90       275
           1       0.77      0.98      0.86       169

    accuracy                           0.88       444
   macro avg       0.88      0.90      0.88       444
weighted avg       0.90      0.88      0.88       444
```

Model 3 Support Vector Classifier

```
best_model.final_model(x, y, model, 132)
```

```
At random state 132 model giving best accuracy score

Training accuracy:-  64.38223938223938
Testing accuracy:-  66.44144144144144


--------------------------------------
Confusion Matrix
[[106  33]
 [116 189]]


Classification Report
              precision    recall  f1-score   support

           0       0.48      0.76      0.59       139
           1       0.85      0.62      0.72       305

    accuracy                           0.66       444
   macro avg       0.66      0.69      0.65       444
weighted avg       0.73      0.66      0.68       444
```

Model 4 AdaBoostClassifier

```
best_model.final_model(x, y, model, 44)
```

```
At random state 44 model giving best accuracy score

Training accuracy:-  89.57528957528957
Testing accuracy:-  90.09009009009009


--------------------------------------
Confusion Matrix
[[190  19]
 [ 25 210]]


Classification Report
              precision    recall  f1-score   support

           0       0.88      0.91      0.90       209
           1       0.92      0.89      0.91       235

    accuracy                           0.90       444
   macro avg       0.90      0.90      0.90       444
weighted avg       0.90      0.90      0.90       444
```

Cross validation score of each of the algorithm:

LogisticRegression	86.08108108108108
GaussianNB	85.17697298185102
SupportVectorClassifier	57.56756756756756
AdaboostClassifier	86.51350907448467

Observations:-

- LogisticRegression: Model shows similar training and testing accuracy but accuracy score is very less than other models hence we can't consider it for model building.
- GassianNB: Model shows similar training and testing accuracy but accuracy score is very less hence we can't consider it for model building.
- SupportVectorClassifier: Model shows similar training and testing accuracy but accuracy score is very less hence we can't consider it for model building.
- AdaboostClassifier: Model show higher accuracy also having less difference in accuracy of testing and training. Hence we can't consider it.

From above Machine Learning Models we find that AdaBoostClassifer show highest accuracy with rangeable cv score with lower differences in accuracy score.

Ensemble Technique

Hyper Parameter Tuning

```
# using hyper parameter tuning for Support Vector Classifier to find out best criterion

model = AdaBoostClassifier()

#default params: base_estimator=None,
#     *, n_estimators=50,
#     learning_rate=1.0,
#     algorithm='SAMME.R',
#     random_state=None,

param = {'base_estimator' : [LogisticRegression(), None],
             'learning_rate' : [0.1, 0.01, 1],
             'algorithm' : ['SAMME', 'SAMME.R'],
             'n_estimators': [100, 50, 10]}

gd = GridSearchCV(model, param_grid=param, cv = 9)
gd.fit(x, y)
gd.best_params_

# {'algorithm': 'SAMME.R',
#  'base_estimator': None,
#  'learning_rate': 1,
#  'n_estimators': 100}
```

Final Model AdaBoostClassifier

```
model = AdaBoostClassifier(algorithm='SAMME.R', base_estimator = None, learning_rate = 1 )
```

```
best_model.best_fit(x, y, model)  #finding best random state with best params
```

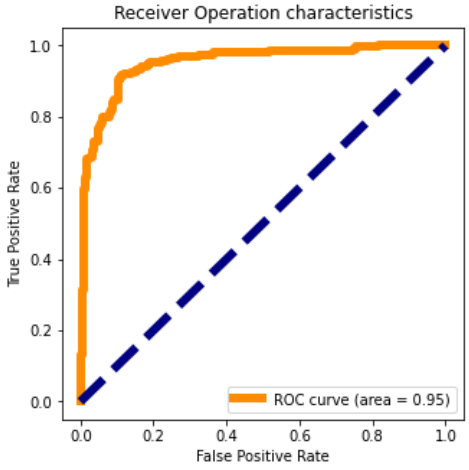
```
At random state 44 model giving best accuracy score

Training accuracy:-  89.57528957528957
Testing accuracy:-  90.09009009009009


-------------------------------------
Confusion Matrix
[[190  19]
 [ 25 210]]


Classification Report
              precision    recall  f1-score   support

           0       0.88      0.91      0.90       209
           1       0.92      0.89      0.91       235

    accuracy                           0.90       444
   macro avg       0.90      0.90      0.90       444
weighted avg       0.90      0.90      0.90       444
```

AOC – ROC Curve

Using Predict Proba function for finding accuracy target variable

```
final_pred_prob = model.predict_proba(x_test)[:,1]
```

```
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresolds = roc_curve(y_test, final_pred_prob)
roc_auc = auc(fpr, tpr)

plt.figure(figsize = (5,5))
plt.plot(fpr, tpr, color = 'darkorange', lw = 6, label = "ROC curve (area = %0.2f)" % roc_auc)
plt.plot([0,1],[0,1], color = 'navy', lw = 6, linestyle = '--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver Operation characteristics")
plt.legend(loc = "lower right")
plt.show()
```



Deploy Model

```
import pickle

filename = "Insurance_claim_predictor.pkl"
pickle.dump(model, open(filename, 'wb'))
```

Loading Model

```
load = pickle.load(open('Insurance_claim_predictor.pkl', 'rb'))
result = load.score(x_test, y_test)
print(result)
```

```
0.9009009009009009
```

CONCLUDING REMARKS

This project has built a model that can detect auto insurance fraudulent customers via customer data. Because these days investigating a single case being very time consuming also it becomes very complex to identifying a whole number of fraudulent.

To identifying fraudulent through dataset will be very grateful to the industry specific.

Four different classifiers were used in this project as above mention but the best model for the dataset chosen is AdaBoostClassifier model.

The Model can predict with an accuracy of 90.09 %, that, the customer is fraudulent or not

Conclusion

```
conclusion = pd.DataFrame()
conclusion['Predicted fraud_report'] = np.array(model.predict(x_test))
conclusion['Actual fraud_report'] = np.array(y_test)
```

```
conclusion.sample(25)
```

	Predicted fraud_report	Actual fraud_report
426	1	1
259	1	1
166	0	0
394	1	1
306	1	1
184	0	0

Github link:

https://github.com/arjun0200/Evaluation_Projects/blob/main/Insurance_Claim_Fraud_Detection.ipynb

https://github.com/arjun0200/Evaluation_Projects/blob/main/Insurance_Claim_Fraud_Detection_final.ipynb