BLOG REPORT

OF

INSURANCE CLAIM FRAUD DETECTION

MADE BY:

SHUHAM SHUKLA

INTRODUCTION

Insurance fraud detection refers to the identification and prevention of fraudulent activities related to money or property insurance. It involves the use of various software-based solutions to analyze historic patterns and incidents to predict future occurrences. The software performs statistical analysis using artificial intelligence (AI), machine learning and traditional rule-based fraud analytics models.

Insurance fraud detection is commonly used by organizations for:

1. Fraud analytics,
2. Authentication,
3. Governance, risk and
4. Compliance to safeguard databases and identify anomalies and vulnerabilities.

All this steps are involved along with the algorithm on the partial training datasets and then it is tested on the test datasets, finally it is then examined by some random splits .The data in the datasets are handled by certain rules which are as follows:.

1. **Problem Definition.**
2. **Data Analysis.**
3. **EDA Concluding Remark.**
4. **Pre-Processing Pipeline.**
5. **Building Machine Learning Models.**
6. **Concluding Remarks.**

Business case

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.

In this project, you are provided a dataset which has the details of the insurance policy along with the customer details. It also has the details of the accident on the basis of which the claims have been made.

Task:

In this example, we will be working with some auto insurance data to demonstrate how we can create a predictive model that predicts if an insurance claim is fraudulent or not.

Insurance Fraud Detection Market Trends:

The increasing occurrence of insurance frauds across industries is one of the key factors driving the growth of the market. Insurance fraud detection systems are widely used to identify any cover-up of the evidence, misinterpretation of the incident or inflating the severity of the loss caused.

Moreover, the rising incidence of inaccurate claims, fake medical records, postdated laws, abductions, deaths and other customer frauds is also providing a thrust to the market growth. In line with this, organizations are widely using artificial intelligence (AI) and the Internet of Things (IoT)-enabled fraud detection solutions for running automated business rules, self-learning models, text mining, image screening, network analysis, predictive analytics and device identification, which is also contributing to the growth of the market.

Importance of machine learning suited to fraud detection?

SUPER FAST

When it comes to fraud decisions, you need results FAST! Machine learning is like having several teams of analysts running hundreds of thousands of queries and comparing the outcomes to find the best result - this is all done in real-time and only takes milliseconds.

As well as making real-time decisions, machine learning is assessing individual customer behavior as it happens.

SCALABLE

Every online business wants to increase its transaction volume. Machine learning systems improve with larger datasets because this gives the system more examples of good and bad.

EFFICIENT

Remember that machine learning is like having several teams running analysis on hundreds of thousands of payments per second. The human cost of this would be immense - the cost of machine learning is just the cost of the servers running.

Machine learning does all the dirty work of data analysis in a fraction of the time it would take for even 100 fraud analysts

ACCURATE

Machine learning models are able to learn from patterns of normal behavior. They are very fast to adapt to changes in that normal behavior and can quickly identify patterns of fraud transactions.

This means that the model can identify suspicious customers even when there hasn’t been a chargeback yet.

Objective :

The techniques in the machine learning is to improve the accuracy of detection on various imbalanced datasets. A machine learning system works by:

1. INPUT DATA
2. EXTRACT FILES
3. TRAIN ALGORITHM
4. CREATE A MODEL

All this steps are involved along with the algorithm on the partial training datasets and then it is tested on the test datasets, finally it is then examined by some random splits .The data in the datasets are handled by certain rules.

About Dataset :

The dataset is about the auto claim insurance dataset along with the customer details for which they have claimed their insurance. Model is predicted whether the claim is authentic or fraud

DATA ANALYSIS

Importing Libraries:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
from sklearn.preprocessing import OrdinalEncoder,LabelEncoder,FunctionTransformer,power_transform
from sklearn.model_selection import train_test_split,GridSearchCV,cross_val_score,StratifiedKFold
!pip install mlrose
!pip install scikit-pyplot
import six
import sys
sys.modules['sklearn.externals.six']=six
import mlrose
from imblearn.over_sampling import SMOTE
from yellowbrick.classifier.rocauc import roc_auc
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
from sklearn.ensemble import ExtraTreesClassifier,RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from xgboost import XGBClassifier
from sklearn.neighbors import KNeighborsClassifier
!pip install scikit-plot
import scikitplot as skplt
from lightgbm import LGBMClassifier
!pip install kmeans-smote
from kmeans_smote import KMeansSMOTE
!pip install pyfiglet
import pyfiglet
```

Extracting dataset

```
data=pd.read_csv('https://raw.githubusercontent.com/dsrscientist/Data-Science-ML-Capstone-Projects/master/Automobile_insurance_fraud.cs
v')
```

```
data
```

	months_as_customer	age	policy_number	policy_bind_date	policy_state	policy_csl	policy_deductable	policy_annual_premium	umbrella_limit	insur
0	328	48	521585	17-10-2014	OH	250/500	1000	1406.91	0	4661
1	228	42	342868	27-06-2006	IN	250/500	2000	1197.22	5000000	4681
2	134	29	687698	06-09-2000	OH	100/300	2000	1413.14	5000000	4306
3	256	41	227811	25-05-1990	IL	250/500	2000	1415.74	6000000	6081
4	228	44	367455	06-06-2014	IL	500/1000	1000	1583.91	6000000	6107
...	...	...	...	...	...	...	...	...	...	...
995	3	38	941851	16-07-1991	OH	500/1000	1000	1310.80	0	4312
996	285	41	186934	05-01-2014	IL	100/300	1000	1436.79	0	6081
997	130	34	918516	17-02-2003	OH	250/500	500	1383.49	3000000	4427
998	458	62	533940	18-11-2011	IL	500/1000	2000	1356.92	5000000	4417
999	456	60	556080	11-11-1996	OH	250/500	1000	766.19	0	6122

1000 rows × 40 columns

```
data.describe()
```

	months_as_customer	age	policy_number	policy_deductable	policy_annual_premium	umbrella_limit	insured_zip	capital-gains	capita
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1.000000e+03	1000.000000	1000.000000	1000.0
mean	203.954000	38.948000	546238.648000	1136.000000	1256.406150	1.101000e+06	501214.488000	25126.100000	-2679
std	115.113174	9.140287	257063.005276	611.864673	244.167395	2.297407e+06	71701.610941	27872.187708	28104
min	0.000000	19.000000	100804.000000	500.000000	433.330000	-1.000000e+06	430104.000000	0.000000	-11110
25%	115.750000	32.000000	335980.250000	500.000000	1089.607500	0.000000e+00	448404.500000	0.000000	-5150
50%	199.500000	38.000000	533135.000000	1000.000000	1257.200000	0.000000e+00	466445.500000	0.000000	-2325
75%	276.250000	44.000000	759099.750000	2000.000000	1415.695000	0.000000e+00	603251.000000	51025.000000	0.000
max	479.000000	64.000000	999435.000000	2000.000000	2047.590000	1.000000e+07	620962.000000	100500.000000	0.000

```
data.shape
```

```
(1000, 40)
```

```
data.tail()
```

	months_as_customer	age	policy_number	policy_bind_date	policy_state	policy_csl	policy_deductable	policy_annual_premium	umbrella_limit	insur
995	3	38	941851	16-07-1991	OH	500/1000	1000	1310.80	0	4312
996	285	41	186934	05-01-2014	IL	100/300	1000	1436.79	0	6081
997	130	34	918516	17-02-2003	OH	250/500	500	1383.49	3000000	4427
998	458	62	533940	18-11-2011	IL	500/1000	2000	1356.92	5000000	4417
999	456	60	556080	11-11-1996	OH	250/500	1000	766.19	0	6122

5 rows × 40 columns

The given dataset contains:

1000 rows and 40 columns. Using this dataset we will be training the Machine Learning models on 80% of the data and the models will be tested on 20% data.

```
data.info()
```

```
<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
```

There is no null values in the given dataset. There are 64 float, 17 integer & 21 object Data types.

In [15]:
```
data.isna().sum()
```

Out[15]:
```
months_as_customer                0
age                               0
policy_number                     0
policy_bind_date                  0
policy_state                      0
policy_csl                        0
policy_deductable                 0
policy_annual_premium             0
umbrella_limit                    0
insured_zip                       0
insured_sex                       0
insured_education_level           0
insured_occupation                0
insured_hobbies                   0
insured_relationship              0
capital-gains                     0
capital-loss                      0
incident_date                     0
incident_type                     0
collision_type                    0
incident_severity                 0
authorities_contacted             0
incident_state                    0
incident_city                     0
incident_location                 0
incident_hour_of_the_day          0
number_of_vehicles_involved       0
property_damage                   0
bodily_injuries                   0
witnesses                         0
police_report_available           0
total_claim_amount                0
injury_claim                      0
property_claim                    0
vehicle_claim                     0
auto_make                         0
auto_model                        0
auto_year                         0
fraud_reported                    0
_c39                           1000
dtype: int64
```

Missing value found in last column and its seems to be empty would rather drop it.

- _c39 is the missing value found in last column. It must be dropped by me.

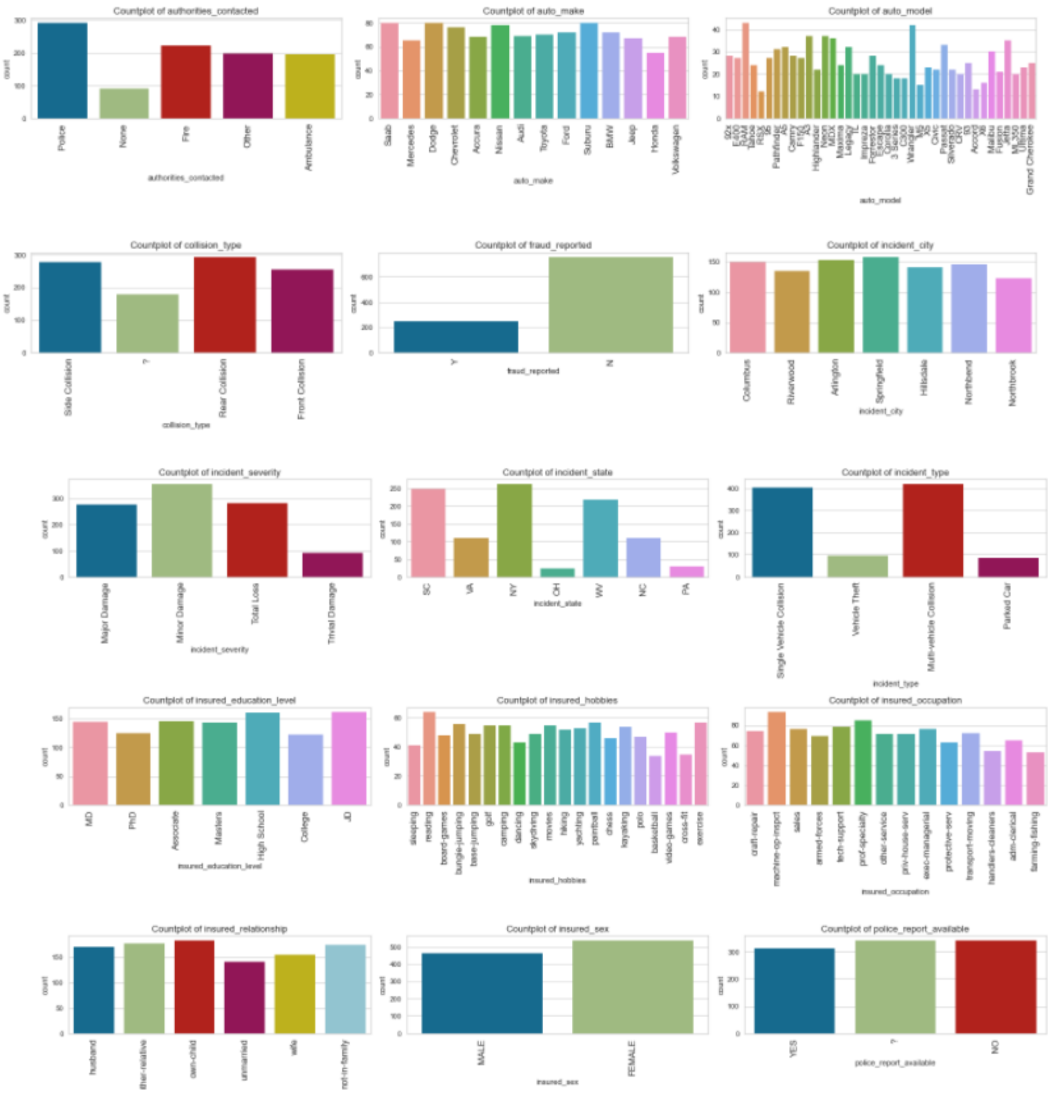
```
df.drop (['_c39','policy_bind_date','incident_location','incident_date'], axis=1, inplace=True)
```

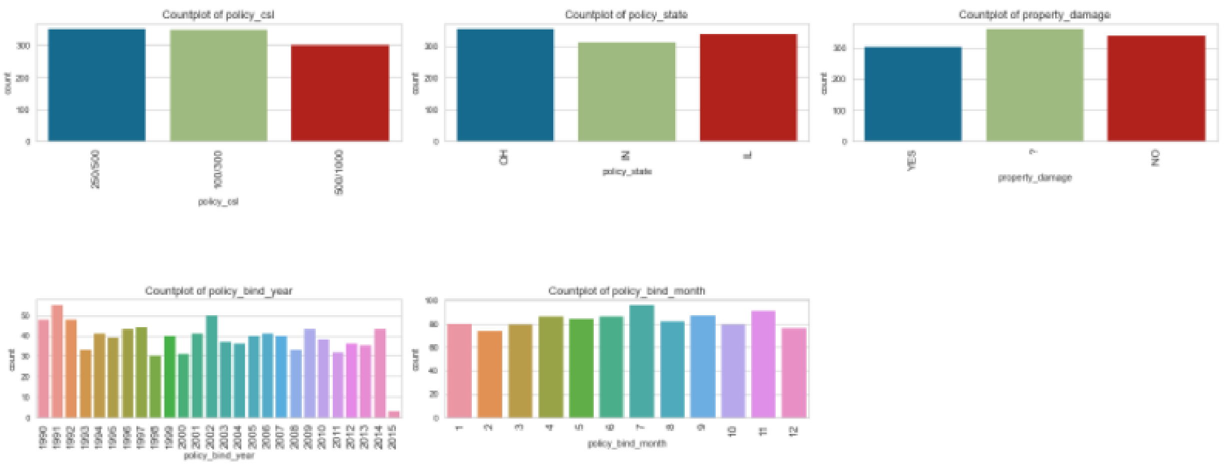
DATA PRE-PROCESSING

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

Univariate Analysis

```
plt.figure(figsize=(20,65))
for i in range(len(col)):
  plt.subplot(15,3,i+1)
  sns.countplot(data[col[i]])
  plt.title(f"Countplot of {col[i]}",fontsize=13)
  plt.xticks(rotation=90,fontsize=13)
  plt.tight_layout()
```





OBSERVATIONS MADE VIA THESE COUNTPLOTS:

1-From countplot authorities_contacted Police has the highest count followed by fire. In most of the cases people have contacted police first.

2-From countplot auto_make Saab,Suburu,Dodge has the highest count in production of automobiles.

3-From countplot auto_modal RAM and Wrangler has the highest count.

4-From countplot collision_type rear collision has highest count.

5-From countplot Fraud Reported maximum number of fraud has'nt been reported or claimed.

6-From countplot incident_city Springfield has highest count whereas Northbrook has least count among all.

7-From countplot incident_severity Minor damange has highest count means mostly people claim insurance for minor damage.

8-From countplot incident_state New york has the highest count means most of the accident happens there.

9-From countplot incident type Multi-vehicle Collision and Single-vehicle Collision has the highest count means most of the accident happens with multiple vehicle and singlw vehicle and very less accident happens with park cars

10-From countplot insured education JD and High school has highest count.

11-From countplot insured_hobbies reading is most popular among all others.

12-From countplot insured_occupation machine-op-inspct has highest count means most of the people who claim insurance has this occupation and people who have farming-fishing occupation has less claim insurance.

13-From countplot insured_relationship own-child has highest count means most of the people who have claimed insurance has child and unmaried has the least count.

14-From countplot insured sex male are less and female are high means people who have sex female has claimed insurance more than male.

15-From countplot policy cls 250/500 and 100/300 has same high count and 500/100 has less count.

16-From countplot policy state IN has less count and Il and ch has same high count.

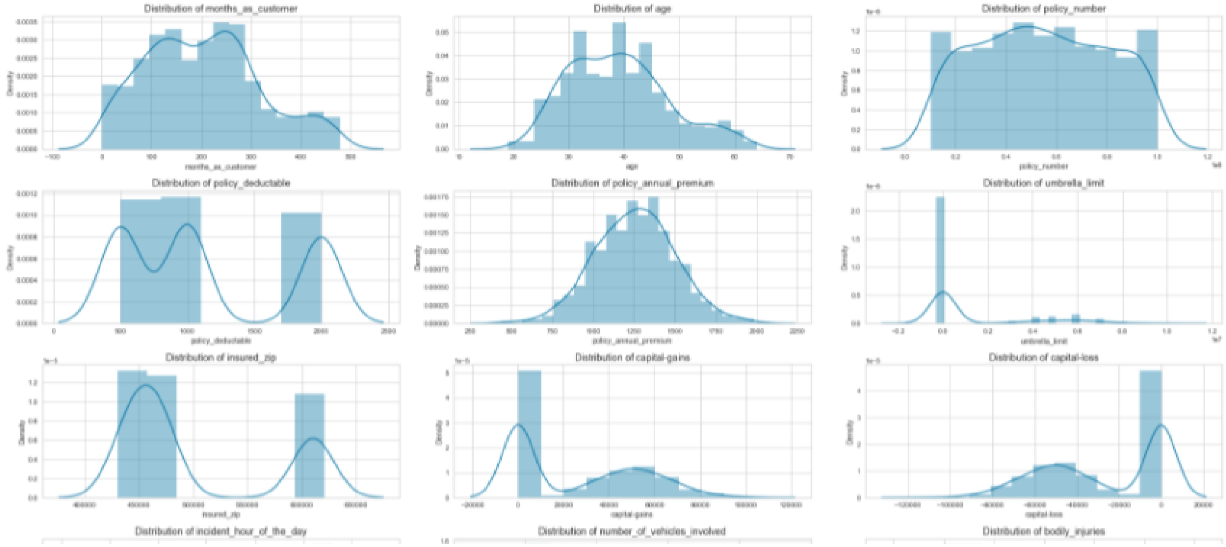
17-From countplot property damage ? and No has high count and yes has less count.

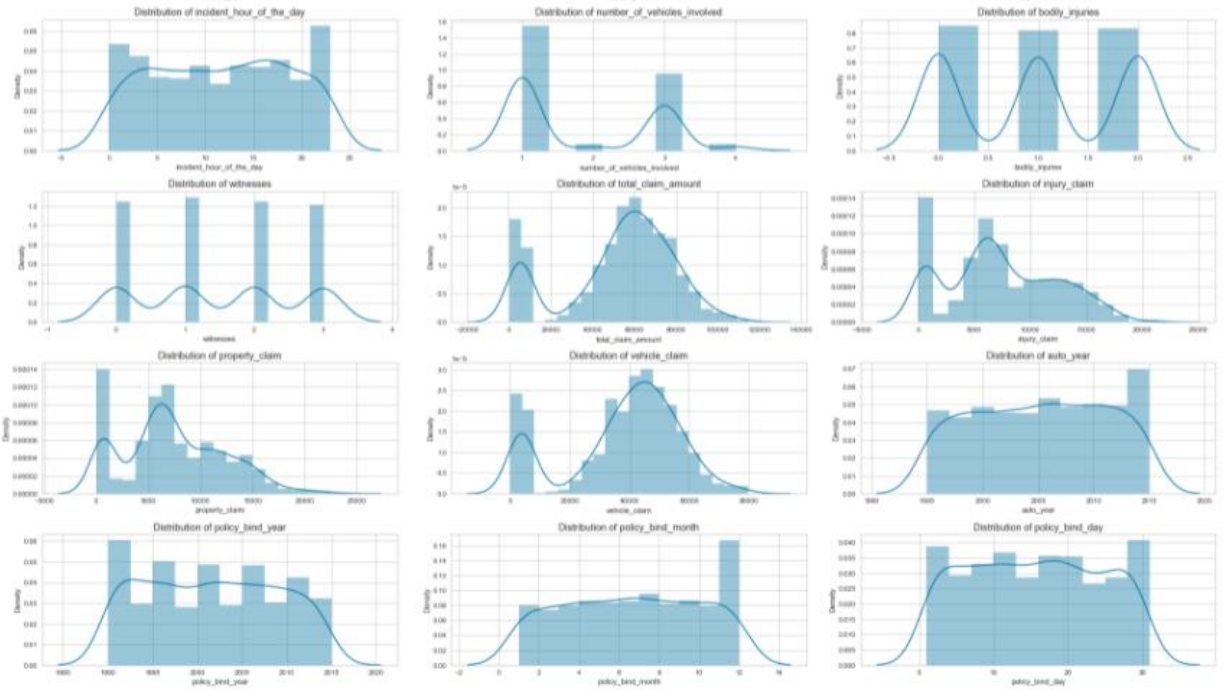
18-From countplot policy bind year most of the people have taken policy in 1991 and 2002 and only few people have taken policy in 2015.

DISTRIBUTION PLOTS OF COLUMN 1

```
col1=['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', 'policy_bind_year', 'policy_bind_month',
      'policy_bind_day']
```

```
plt.figure(figsize=(25,35))
for i in range(len(col1)):
  plt.subplot(10,3,i+1)
  plt.title(f"Distribution of {col1[i]}",fontsize=14)
  sns.distplot(data[col1[i]])
  plt.tight_layout()
```





OBSERVATIONS MADE VIA THESE DISTPLOTS:

1- From months_As_customers most of the people lies in 0-100 and there are less number of people who lies in b/w 300-500 who are loyal customers.

2-From distribution of age most of the people lies between 30-45 and less people are in between 50-60.

3-From policy_annual_premium it is normally distributed.

4-From policy deductable i can say the value is between 500-1000 and 17000-20000

5-From capital gain 0-10000 has highest peak and with capital loss 0 to -10000 has high peak rest of the distribution are same both features.

6-From total_claim_amount 0-10000 has highest peak and rest of all values are normally distributed.

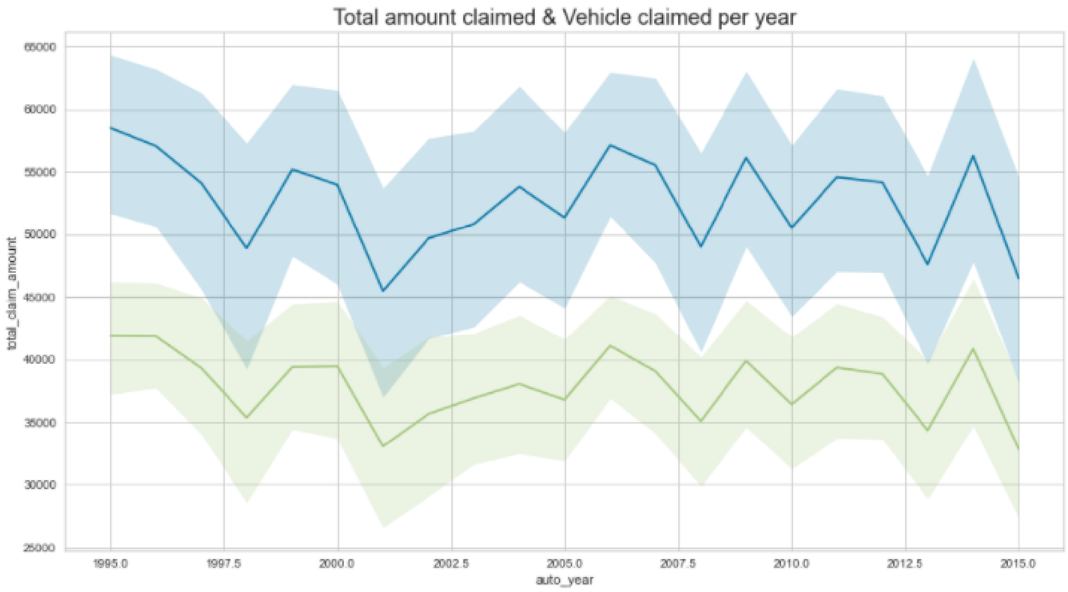
7-From distribution of property claim most of the people have claimed that 0-1000 values and from 4000-10000 these are 2nd highest people who have claimed for this values and there are very few people who claimed for 20000-25000 values.

8-From distribution of vehicle claim there are many people who claimed for 0-10000 and rest of the value has normal distribution.

Bivariate analysis

```
plt.figure(figsize=(15,8))
sns.lineplot(x='auto_year',y='total_claim_amount',data=data)
sns.lineplot(x='auto_year',y='vehicle_claim',data=data)
plt.title("Total amount claimed & Vehicle claimed per year",fontsize=18)
```

```
Text(0.5, 1.0, 'Total amount claimed & Vehicle claimed per year')
```

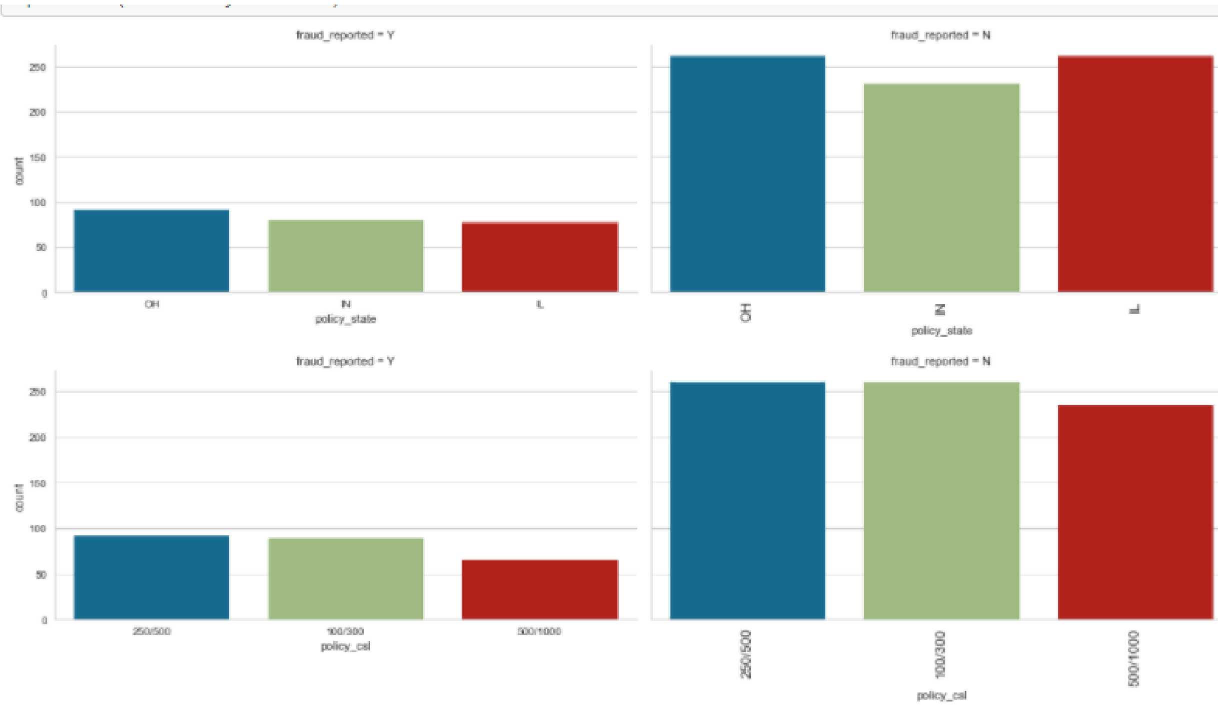


Total amount claimed has high value and vehicle claimed has low count although both have same distribution.

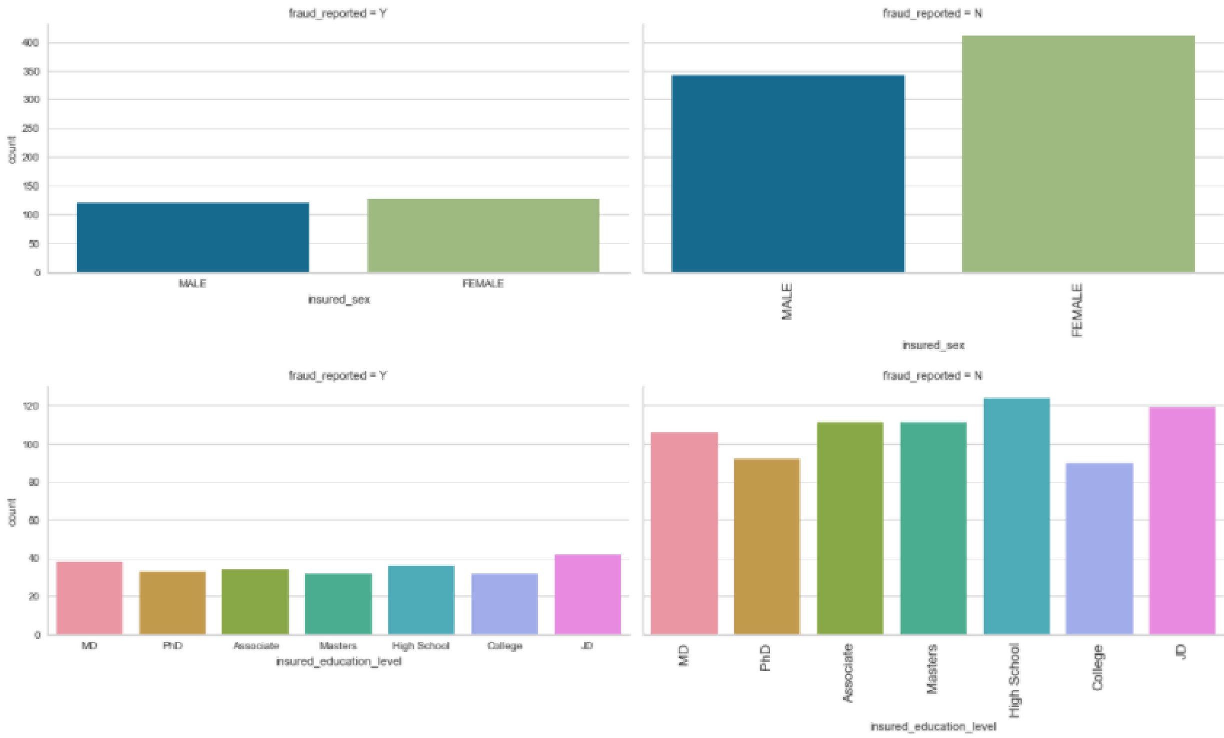
```
data1=data.select_dtypes(include='object')
```

```
new_data=list(data1.columns)
new_list=new_data
new_data.remove('fraud_reported')
for col in new_list:
  sns.catplot(x=col,col='fraud_reported',data=data1,kind='count',height=4,aspect=2)
  plt.xticks(rotation=90,fontsize=13)
```

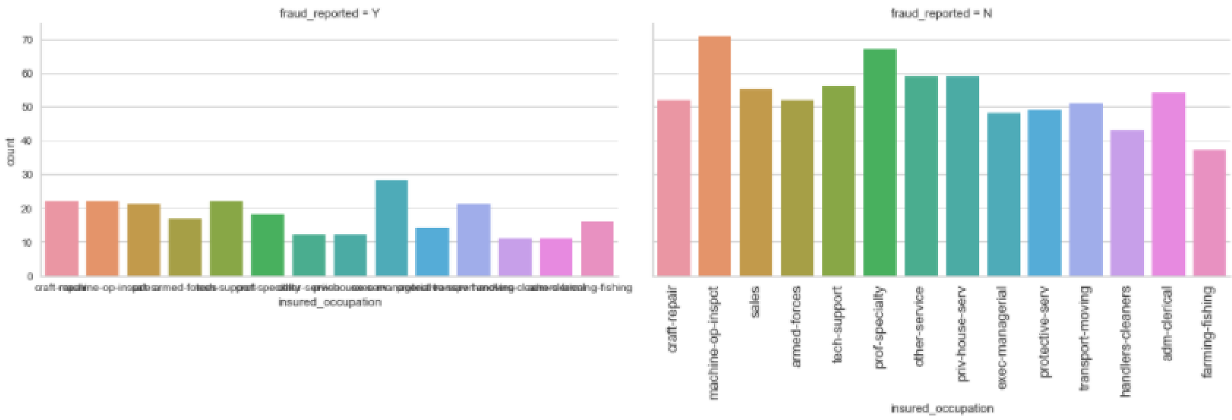
OUTPUT

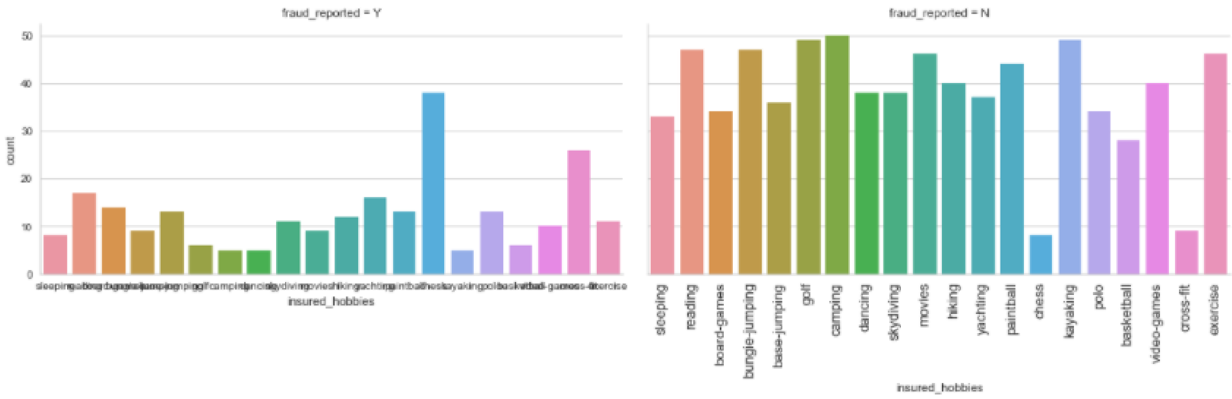


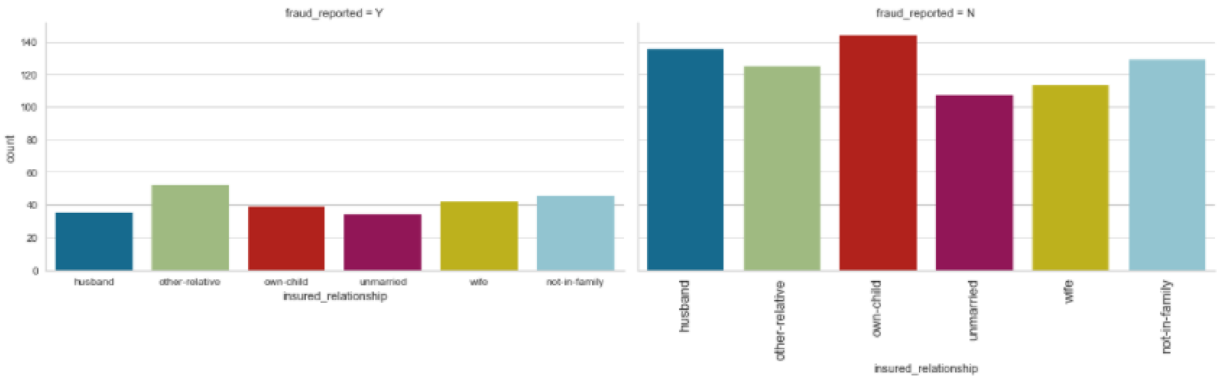
fraud_reported = Y
fraud_reported = N
count
OH
IN
IL
policy_state
250/500
100/300
500/1000
policy_csl

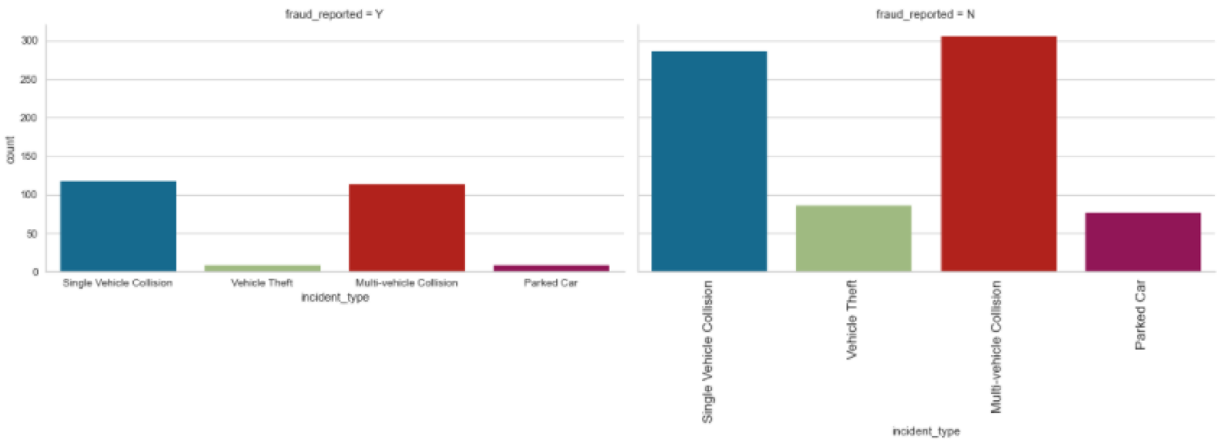


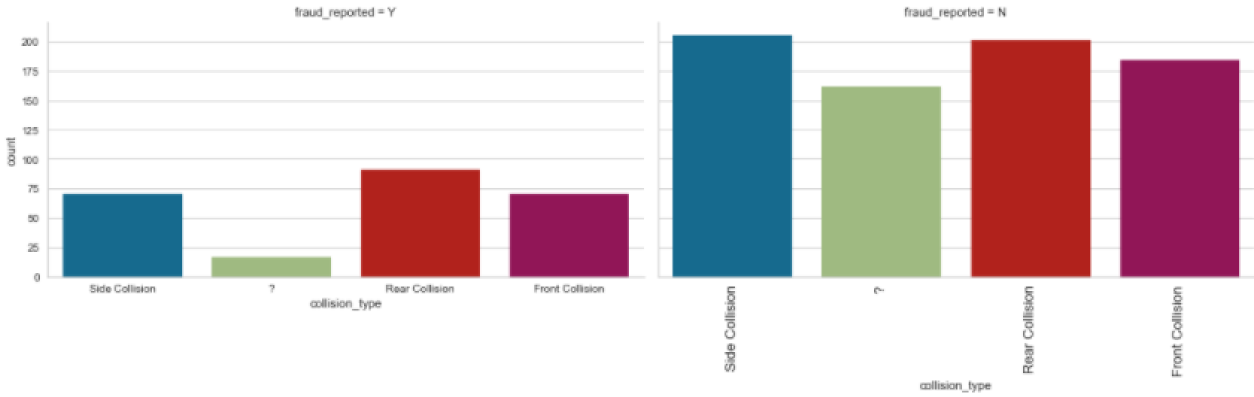
fraud_reported = Y
fraud_reported = N
count
MALE
FEMALE
insured_sex
MD
PhD
Associate
Masters
High School
College
JD
insured_education_level

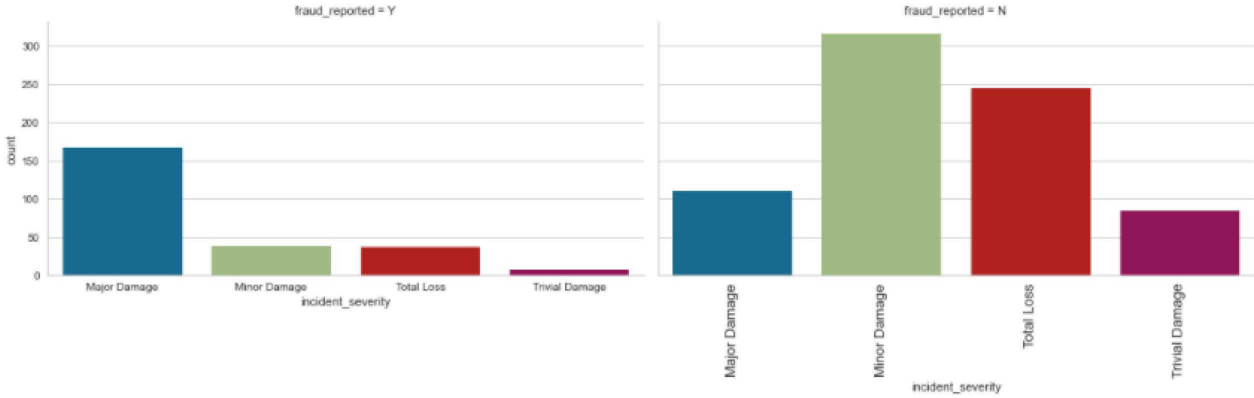

fraud_reported = Y
fraud_reported = N
count
insured_occupation
craft-repair
machine-op-inspct
sales
armed-forces
tech-support
prof-specialty
other-service
priv-house-serv
exec-managerial
protective-serv
transport-moving
handlers-cleaners
adm-clerical
farming-fishing
insured_occupation

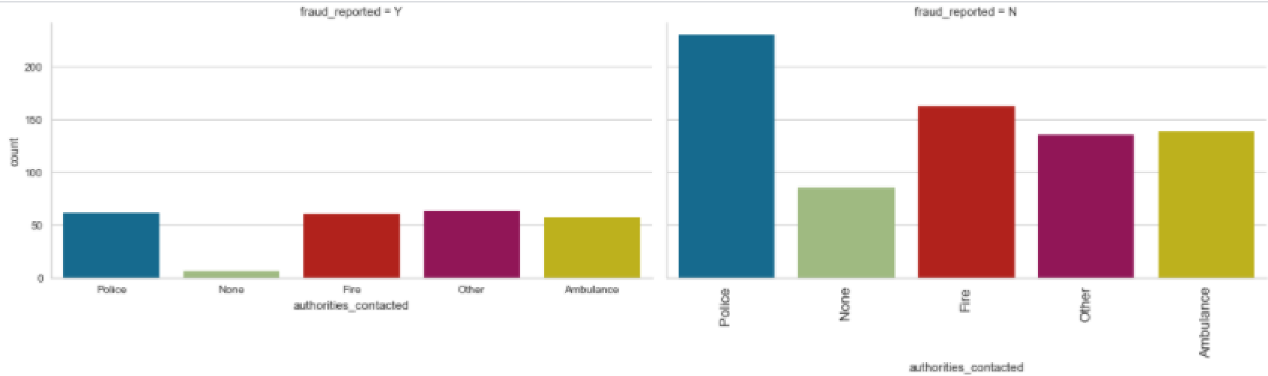

fraud_reported = Y
fraud_reported = N
count
insured_hobbies
sleeping
reading
board-games
bungie-jumping
base-jumping
golf
camping
dancing
skydiving
movies
hiking
yachting
paintball
chess
kayaking
polo
basketball
video-games
cross-fit
exercise
insured_hobbies

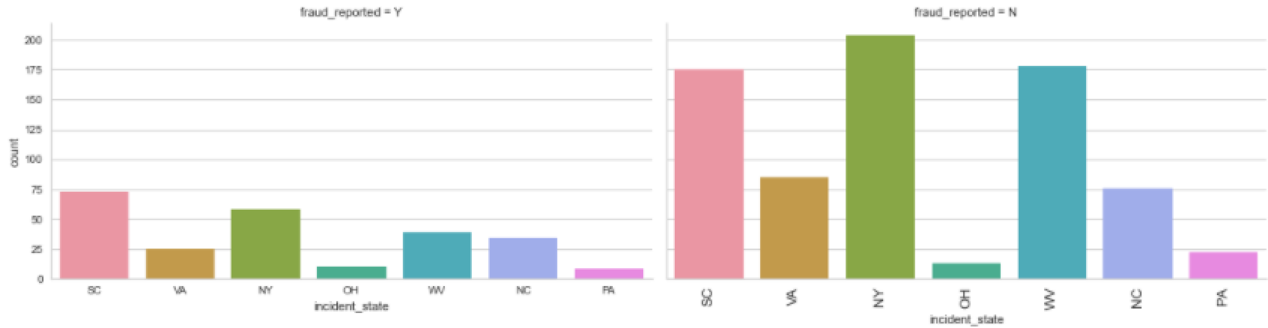

fraud_reported = Y
fraud_reported = N
count
husband
other-relative
own-child
unmarried
wife
not-in-family
insured_relationship
husband
other-relative
own-child
unmarried
wife
not-in-family
insured_relationship

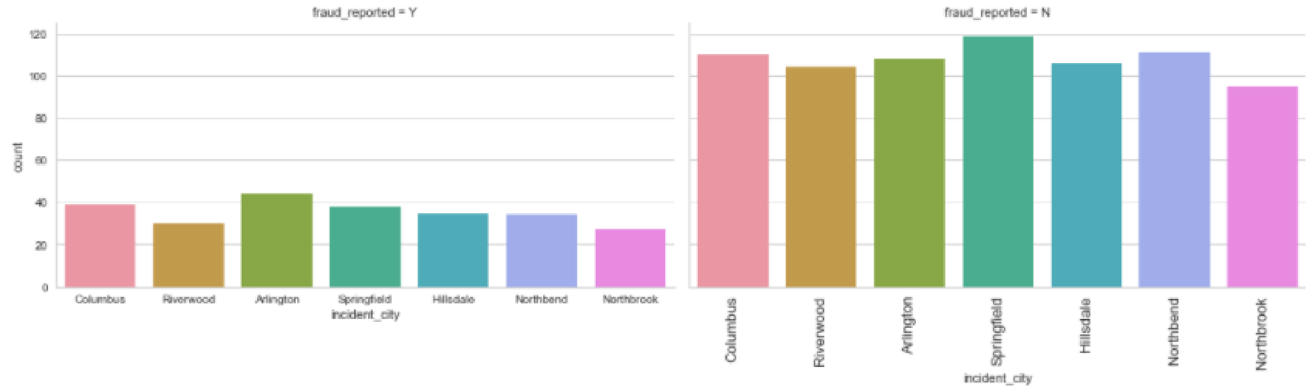

fraud_reported = Y
fraud_reported = N
count
Single Vehicle Collision
Vehicle Theft
Multi-vehicle Collision
Parked Car
incident_type
Single Vehicle Collision
Vehicle Theft
Multi-vehicle Collision
Parked Car
incident_type

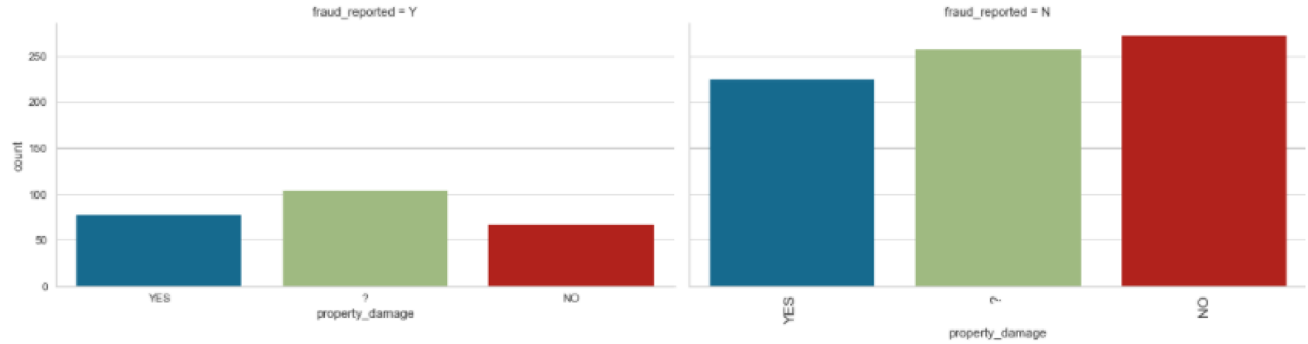

fraud_reported = Y
fraud_reported = N
count
0
25
50
75
100
125
150
175
200
Side Collision
?
Rear Collision
Front Collision
collision_type

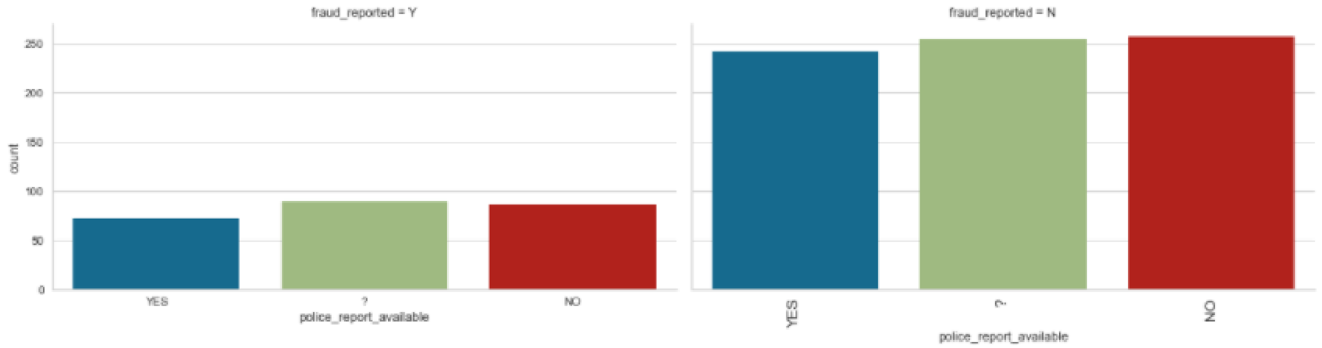

fraud_reported = Y
fraud_reported = N
count
0
50
100
150
200
250
300
Major Damage
Minor Damage
Total Loss
Trivial Damage
incident_severity

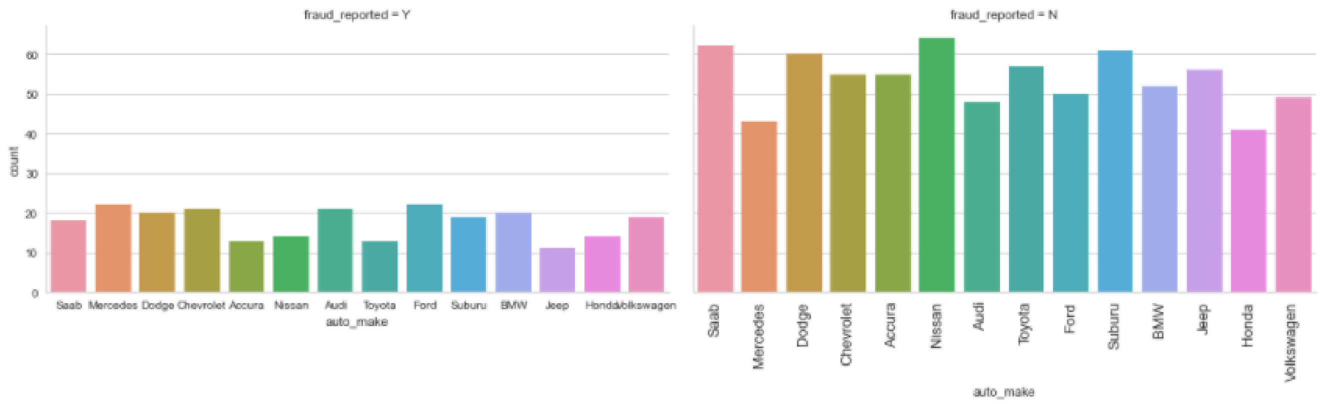

fraud_reported = Y
fraud_reported = N
count
0
50
100
150
200
Police
None
Fire
Other
Ambulance
authorities_contacted

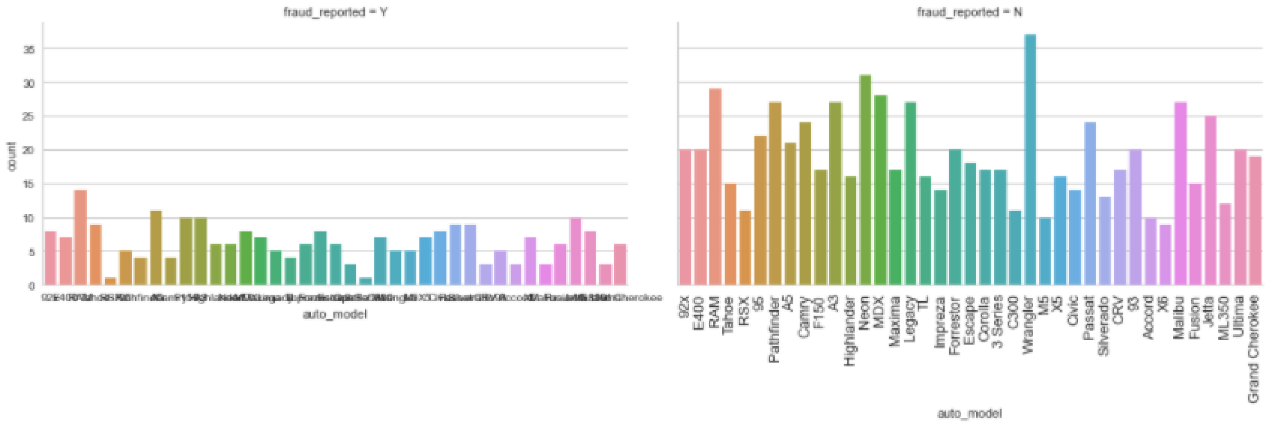

fraud_reported = Y
fraud_reported = N
count
0
25
50
75
100
125
150
175
200
SC
VA
NY
OH
WV
NC
PA
incident_state


fraud_reported = Y
fraud_reported = N
count
incident_city
Columbus
Riverwood
Arlington
Springfield
Hillsdale
Northbend
Northbrook


fraud_reported = Y
fraud_reported = N
count
property_damage
YES
?
NO


fraud_reported = Y
fraud_reported = N
count
police_report_available
YES
?
NO


fraud_reported = Y
fraud_reported = N
count
auto_make
Saab
Mercedes
Dodge
Chevrolet
Accura
Nissan
Audi
Toyota
Ford
Suburu
BMW
Jeep
Honda
Volkswagen



OBSERVATIONS

1-People who reported fraud have high value of CH of policy state.

2-People who reported fraud have equal count on both male & female.

3-Mostly People who reported fraud there hobbies are chess.

4-Mostly people who reported fraud mostly have relationship status other-relative.

5-Mostly people who reported fraud have incident_type status single and multi-vehicle collision

6-Mostly people who reported fraud have collision_type status of Rear collision and very less people who claim unknown (?)

7-Mostly People who reported fraud have incident_severity status major damage and very less people who have status trivial damage

8-Mostly People who reported fraud have incident_state SC who claimed more fraud

9-Mostly People who reported fraud have incident_city ARLINGTON who claimed more fraud.

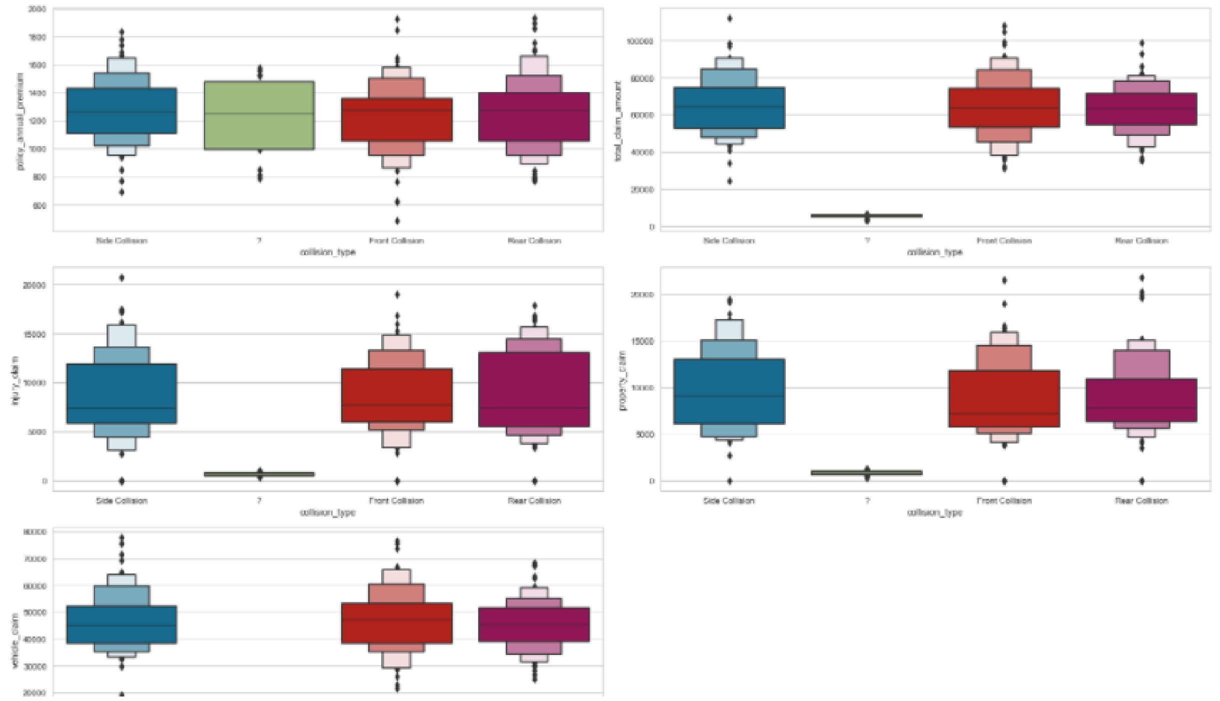
10-Mostly People who reported fraud have property_damage ? followed by YES.

11-Mostly People who reported fraud have auto_make Mercedes, Ford followed by Audi.

```
data_no=data.query("fraud_reported=='Y'")
```

```
col3=['policy_annual_premium','total_claim_amount','injury_claim','property_claim', 'vehicle_claim']
```

```
plt.figure(figsize=(20,85))
for i in range(len(col3)):
  plt.subplot(20,2,i+1)
  sns.boxenplot(x=data_no['collision_type'],y=data_no[col3[i]])
  plt.tight_layout()
```



OBSERVATION

1-People who have claimed fraud insurance they have policy_annual_premium whoes mean value is betwenn 1300 and max value is approx 1800 and where collision type is ? People mostly have claimed 25% to 75% and for front and collision type people have claimed mostly 25% of the value.

2-From total_claim_amount vs fraud for side and front collision people have claimed maximum of 90000 and for front collision value is around 75000.

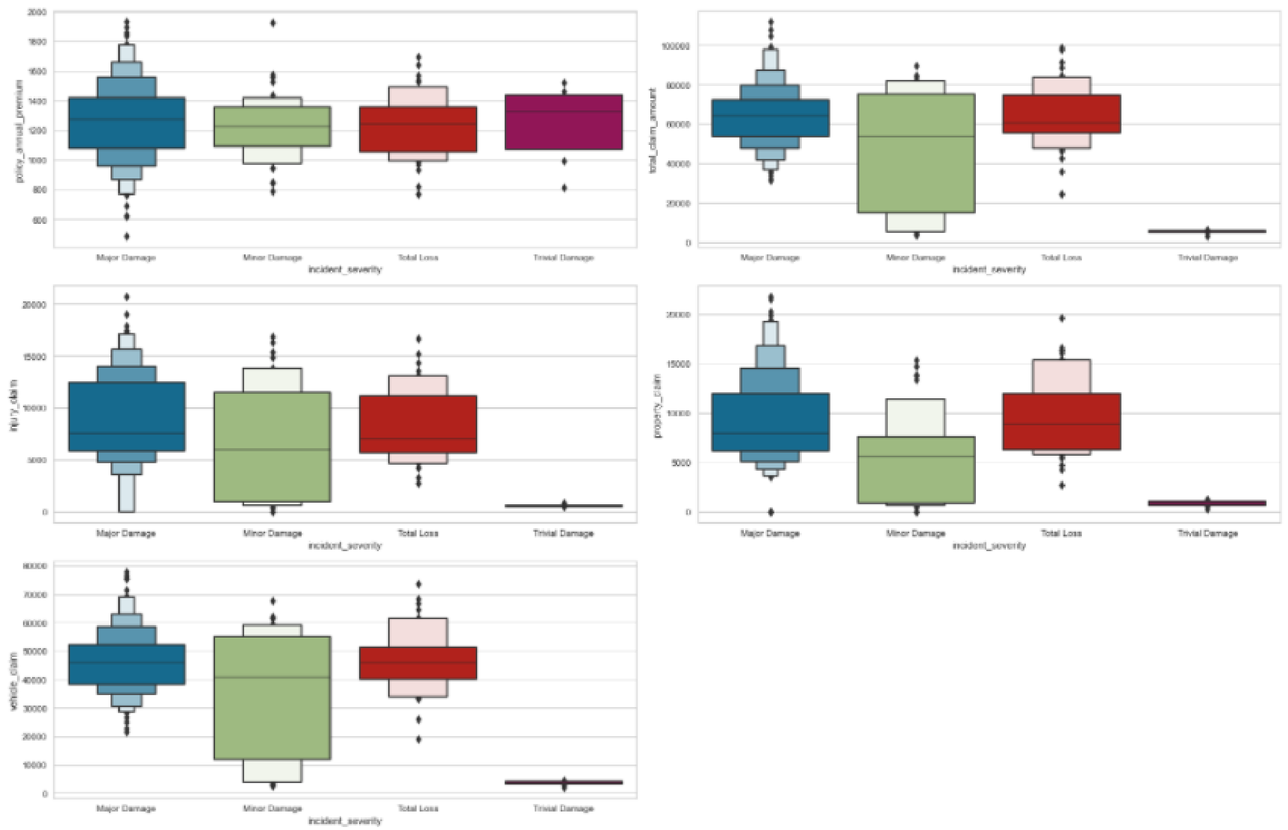
3-From injury claim where is fraud is yes Side and rear collision have same max value 15000 and mean value of 700 and for injury claim people have claimed mostly 75% to max value and less people are there who claimed for less than 25%

4-From property_claim who have fraud side collision have 9000 mean value and 17000 max values and front and rear collision almost have approx values and for side collision mostly people have claimed more than 50% of value same with front and real collision most of the people have claimed large value that is greater than 50%.

5-From vehicle claim where fraud is rear and side have almost same mean value and front side have a different value.

6-From the values that is mentioned above that means people have claimed these much of money from company in name of fraud means value represent the average value that people have claimed and max value represent the value that the maximum money that people have claimed in name of fraud.

```
plt.figure(figsize=(20,85))
for i in range(len(col3)):
  plt.subplot(20,2,i+1)
  sns.boxenplot(x=data_no['incident_severity'],y=data_no[col3[i]])
  plt.tight_layout()
```



OBSERVATIONS:

1-From policy annual premium minor damage most of the people have claimed value between 25-75%. Also, there are some outliers who have claimed for minimum and maximum values & for trivial damage most of the people have claimed for the value that is less than average.

2-From total claim amount for minor damage mostly people have value that is less than average.

3-From injury claim for major damage people have claimed more than the mean and for major damage most of the people have claimed 25% or 75% of value and for total loss most of the people have claimed more than mean value.

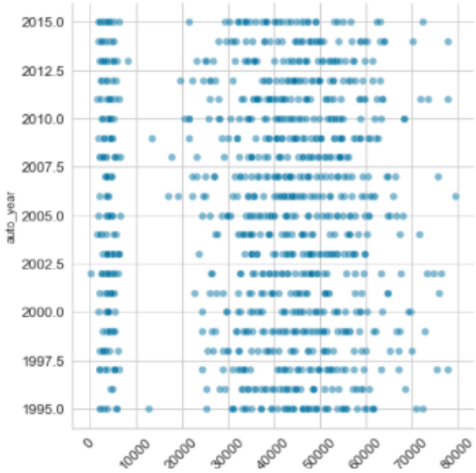
4-From property claim for major damage the most of the value lies above mean means for major damage and people get more money & for minor damage people get most of the value that is less than mean or average.

5-From vehicle claim for minor damage maximum people get money that is less than mean less no. of people get money that is greater than mean value.

```
col4=['total_claim_amount',
 'injury_claim',
 'property_claim',
 'vehicle_claim']
```

```
sns.relplot(y='auto_year',x='vehicle_claim',data=data,alpha=.5,palette='muted',sizes=(40,4
plt.xticks(rotation=45,fontsize=13)
plt.yticks(fontsize=13)
```

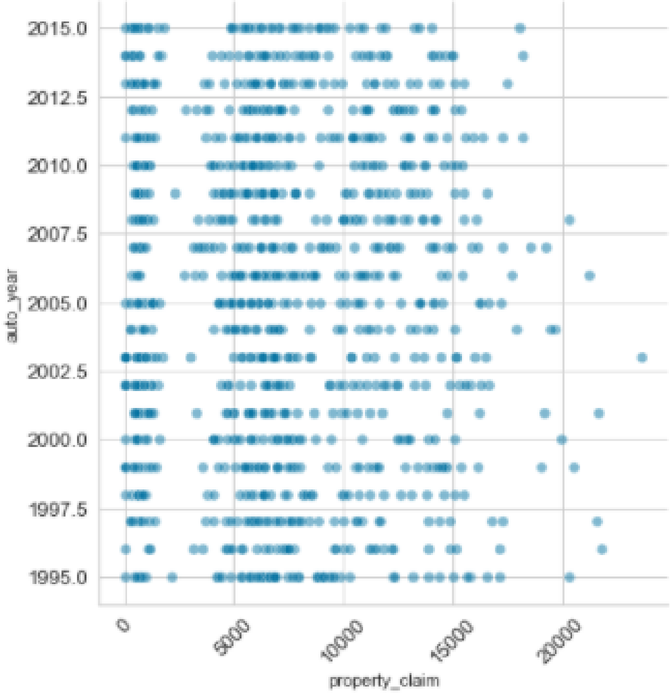
```
(array([1992.5, 1995. , 1997.5, 2000. , 2002.5, 2005. , 2007.5, 2010. ,
        2012.5, 2015. , 2017.5]),
 [Text(0, 1992.5, '1992.5'),
  Text(0, 1995.0, '1995.0'),
  Text(0, 1997.5, '1997.5'),
  Text(0, 2000.0, '2000.0'),
  Text(0, 2002.5, '2002.5'),
  Text(0, 2005.0, '2005.0'),
  Text(0, 2007.5, '2007.5'),
  Text(0, 2010.0, '2010.0'),
  Text(0, 2012.5, '2012.5'),
  Text(0, 2015.0, '2015.0'),
  Text(0, 2017.5, '2017.5')])
```



Vehicle claim value is not proportional to year, it remains same by the years increasing.

```
sns.relplot(y='auto_year',x='property_claim',data=data,alpha=.5,palette='muted',sizes=(40,400),height=6)
plt.xticks(rotation=45,fontsize=13)
plt.yticks(fontsize=13)
```

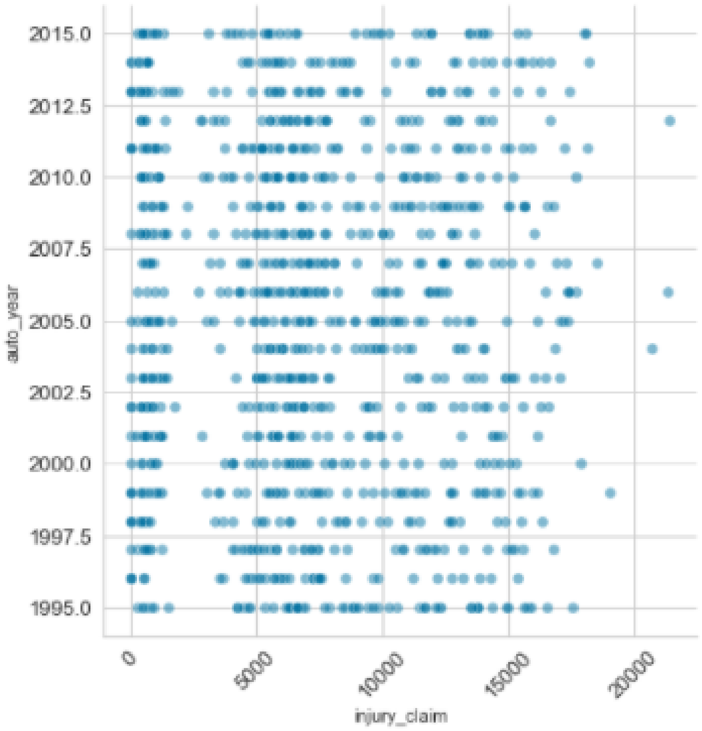
```
(array([1992.5, 1995. , 1997.5, 2000. , 2002.5, 2005. , 2007.5, 2010. ,
        2012.5, 2015. , 2017.5]),
 [Text(0, 1992.5, '1992.5'),
  Text(0, 1995.0, '1995.0'),
  Text(0, 1997.5, '1997.5'),
  Text(0, 2000.0, '2000.0'),
  Text(0, 2002.5, '2002.5'),
  Text(0, 2005.0, '2005.0'),
  Text(0, 2007.5, '2007.5'),
  Text(0, 2010.0, '2010.0'),
  Text(0, 2012.5, '2012.5'),
  Text(0, 2015.0, '2015.0'),
  Text(0, 2017.5, '2017.5')])
```



Property claim value is not proportional to year, it remains same by the years increasing.

```
sns.relplot(y='auto_year',x='injury_claim',data=data,alpha=.5,palette='muted',sizes=(40,400),height=6)
plt.xticks(rotation=45,fontsize=13)
plt.yticks(fontsize=13)
```

```
(array([1992.5, 1995. , 1997.5, 2000. , 2002.5, 2005. , 2007.5, 2010. ,
        2012.5, 2015. , 2017.5]),
 [Text(0, 1992.5, '1992.5'),
  Text(0, 1995.0, '1995.0'),
  Text(0, 1997.5, '1997.5'),
  Text(0, 2000.0, '2000.0'),
  Text(0, 2002.5, '2002.5'),
  Text(0, 2005.0, '2005.0'),
  Text(0, 2007.5, '2007.5'),
  Text(0, 2010.0, '2010.0'),
  Text(0, 2012.5, '2012.5'),
  Text(0, 2015.0, '2015.0'),
  Text(0, 2017.5, '2017.5')])
```



Injury claim value is not proportional to year, it almost remains same by the years increasing.

```
sns.relplot(y='auto_year',x='total_claim_amount',data=data,alpha=.5,palette='muted',sizes=(40,400),height=6)
plt.xticks(rotation=45,fontsize=13)
plt.yticks(fontsize=13)
```

```
(array([1992.5, 1995. , 1997.5, 2000. , 2002.5, 2005. , 2007.5, 2010. ,
        2012.5, 2015. , 2017.5]),
 [Text(0, 1992.5, '1992.5'),
  Text(0, 1995.0, '1995.0'),
  Text(0, 1997.5, '1997.5'),
  Text(0, 2000.0, '2000.0'),
  Text(0, 2002.5, '2002.5'),
  Text(0, 2005.0, '2005.0'),
  Text(0, 2007.5, '2007.5'),
  Text(0, 2010.0, '2010.0'),
  Text(0, 2012.5, '2012.5'),
  Text(0, 2015.0, '2015.0'),
  Text(0, 2017.5, '2017.5')])
```

Total claim amount value is almost same as the year increasing.

CORRELATION GRAPH

```
X_corr=X.corr()
plt.figure(figsize=(12,8))
sns.heatmap(X_corr,annot=True)
```

```
<AxesSubplot:>
```

OBSERVATION:

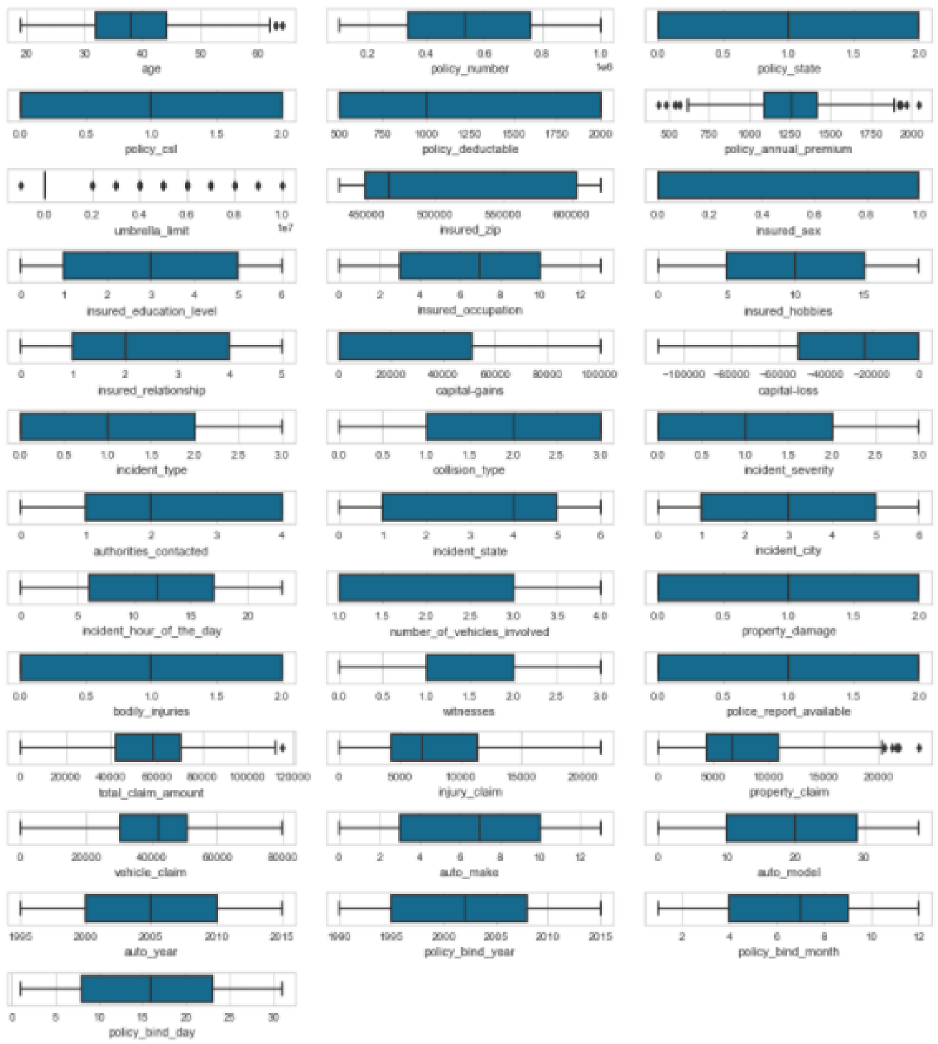
From above heat map months as customer and age are having correlation more than 90% while rest of all features are having good correlation.

I should drop month as customer.

OUTLIERS BEFORE REMOVING

Outliers

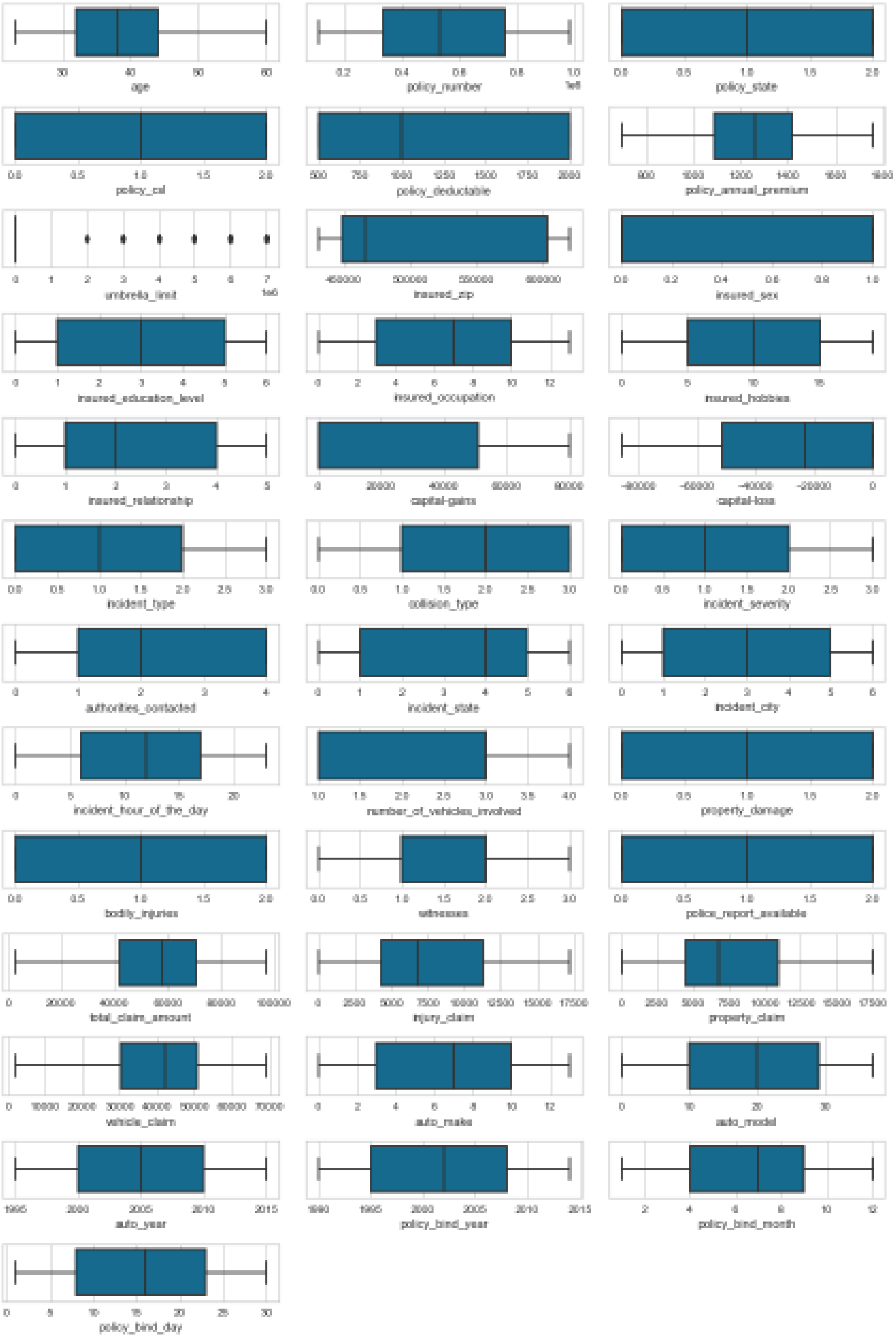
```
X_val=X.columns.values
plt.figure(figsize=(12,20))
for i in range(len(X_val)):
  plt.subplot(20,3,i+1)
  sns.boxplot(X[X_val[i]])
  plt.tight_layout()
```



OUTLIERS SHOULD BE REMOVED

AFTER REMOVING OUTLIERS

```
X_val=X.columns.values
plt.figure(figsize=(12,20))
for i in range(len(X_val)):
  plt.subplot(15,3,i+1)
  sns.boxplot(X[X_val[i]])
  plt.tight_layout()
```



BALANCING DATA

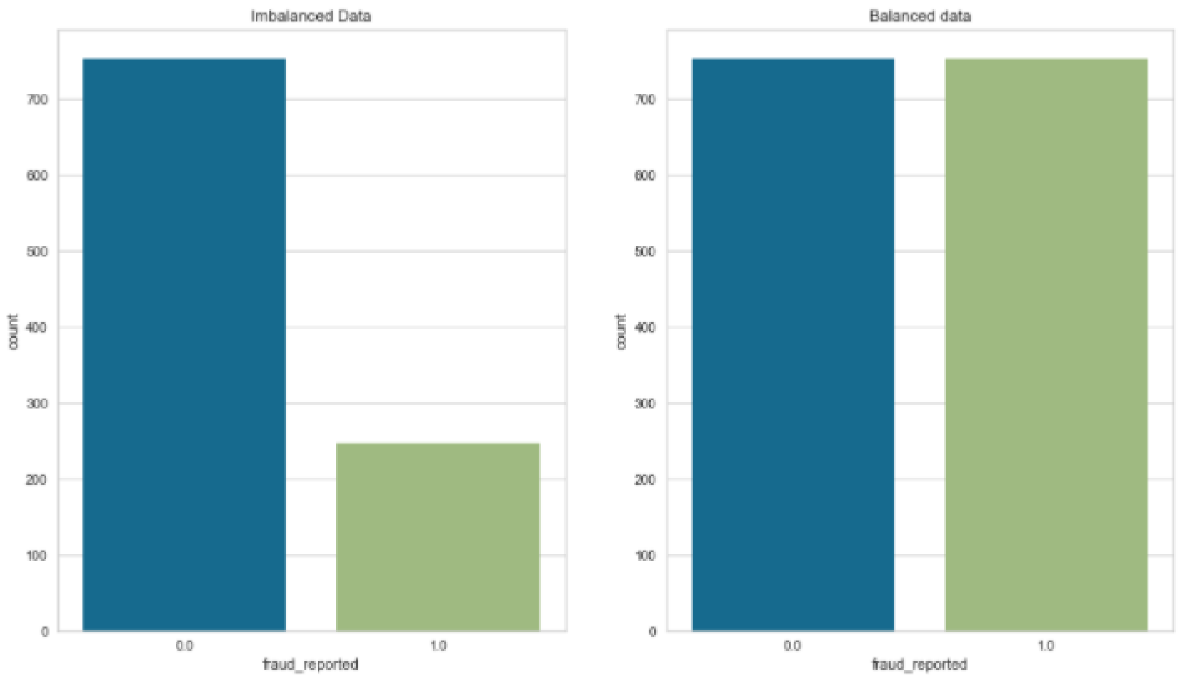
```
def balancing_data(X,y):
    x=X
    smote=SMOTE(random_state=42)
    X_res,y_res=smote.fit_resample(X,y)
    X_new=pd.DataFrame(X_res,columns=x.columns)
    y_new=pd.DataFrame(y_res,columns=['fraud_reported'])
    return X_new,y_new
```

```
X_new,y_new=balancing_data(X,y)
```

CHECKING BALANCED DATA

```
plt.figure(figsize=(15,8))
plt.subplot(1,2,1)
sns.countplot(data['fraud_reported'])
plt.title("Imbalanced Data")
plt.subplot(1,2,2)
sns.countplot(y_new['fraud_reported'])
plt.title("Balanced data")
```

```
Text(0.5, 1.0, 'Balanced data')
```



COUNT has been leveled up. The given data is now BALANCED.

SKEWNESS

Skewness is a quantifiable measure of how distorted a data sample is from the normal distribution.

age	0.461109
policy_number	0.036105
policy_state	-0.026177
policy_csl	0.088928
policy_deductable	0.477887
policy_annual_premium	-0.046551
umbrella_limit	1.712094
insured_zip	0.816445
insured_sex	0.148630
insured_education_level	-0.000148
insured_occupation	-0.058881
insured_hobbies	-0.061563
insured_relationship	0.077488
capital-gains	0.437885
capital-loss	-0.366324
incident_type	0.101507
collision_type	-0.193345
incident_severity	0.279016
authorities_contacted	-0.121744
incident_state	-0.148865
incident_city	0.049531
incident_hour_of_the_day	-0.035584
number_of_vehicles_involved	0.502664
property_damage	0.106418
bodily_injuries	0.014777
witnesses	0.019636
police_report_available	0.052967
total_claim_amount	-0.646051
injury_claim	0.213656
property_claim	0.247848
vehicle_claim	-0.674588
auto_make	-0.018797
auto_model	-0.088625
auto_year	-0.048289
policy_bind_year	0.050075
policy_bind_month	-0.029321
policy_bind_day	0.017380
dtype: float64	

We could see most of the skewness is present in "umbrella_limit". It is with the ordinal data, so we will ignore the skewness.

SLIP TEST AND TRAIN DATA

```
x_train,x_test,y_train,y_test=train_test_split(X_new,y_new,test_size=0.3,random_state=42)
```

MACHINE LEARNING MODELS:

A machine learning model is the output of the training process and is defined as the mathematical representation of the real-world process. The machine learning algorithms find the patterns in the training dataset, which is used to approximate the target function and is responsible for mapping the inputs to the outputs from the available dataset.

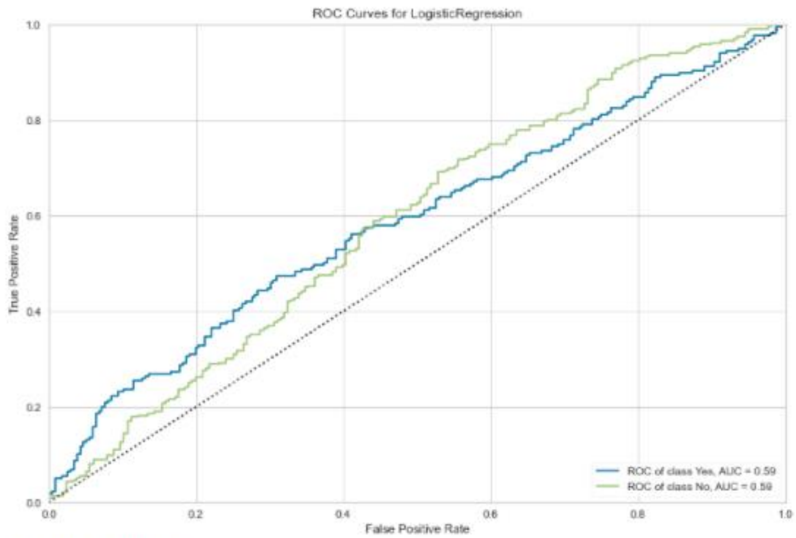
Since the dataset is large to my system configurations, ensemble techniques will be efficient although I’m testing the results with the below algorithms.:

```
models={
    "Logistic Regression":LogisticRegression(),
    "DecisionTree Classifier":DecisionTreeClassifier(),
    "ExtraTrees Classifier":ExtraTreesClassifier(),
    "RandomForest Classifier":RandomForestClassifier(),
    "XGB Classifier":XGBClassifier(),
    "LGBM Classifier":LGBMClassifier()
}
```

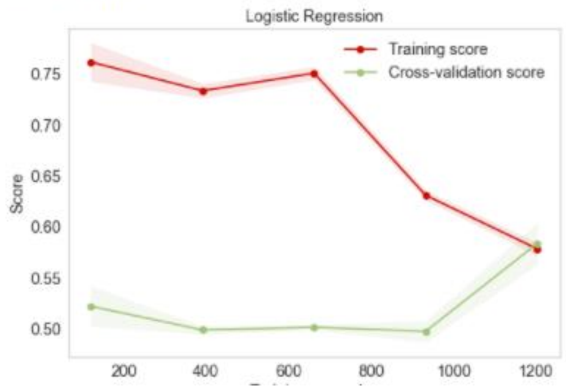
```
skf=StratifiedKFold(n_splits=5,shuffle=True)
Score=[]
CVS=[]
MODEL=[]
for name,model in models.items():
  MODEL.append(name)
  model.fit(X_train,y_train)
  y_pred=model.predict(X_test)
  print('\n')
  ac=accuracy_score(y_test,y_pred)
  Score.append(ac)
  print("Accuracy_Score",ac)
  print('\n')
  print("SCORE",model.score(X_test,y_test))
  print("\n")
  cm=confusion_matrix(y_test,y_pred)
  print('Confusion metrics')
  print('\n')
  print(cm)
  print("CLASSIFICATION REPORT")
  report=classification_report(y_test,y_pred)
  print('\n')
  print(report)
  csv=cross_val_score(model,X_new,y_new,cv=skf).mean()
  CVS.append(csv*100)
  print("Cross_Val_Score",csv)
  print('\n')
  print("ROC AUC CURVE")
  plt.figure(figsize=(12,8))
  roc_auc(model,X_train,y_train,X_test=X_test,y_test=y_test,classes=['Yes','No'],micro=False,macro=False)
  print("MODEL LEARNING CURVE")
  skplt.estimators.plot_learning_curve(model,X_new,y_new,cv=skf,scoring='accuracy',text_fontsize='large',title=name)
  plt.show()
```

Logistic Regression Model:

Accuracy Score 0.5685840707964602
SCORE 0.5685840707964602
Confusion metrics
[125 91]
[104 132]
CLASSIFICATION REPORT
precision   recall f1-score   support
0.0   0.55   0.58   0.56   216
1.0   0.59   0.56   0.58   236
accuracy   0.57   452
macro avg   0.57   0.57   0.57   452
weighted avg   0.57   0.57   0.57   452
Cross_Val_Score 0.5650810763239533



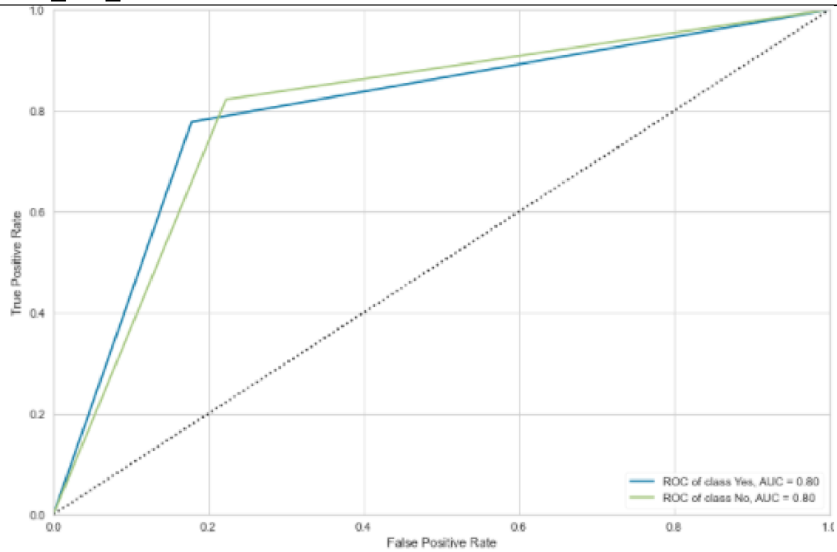
MODEL LEARNING CURVE



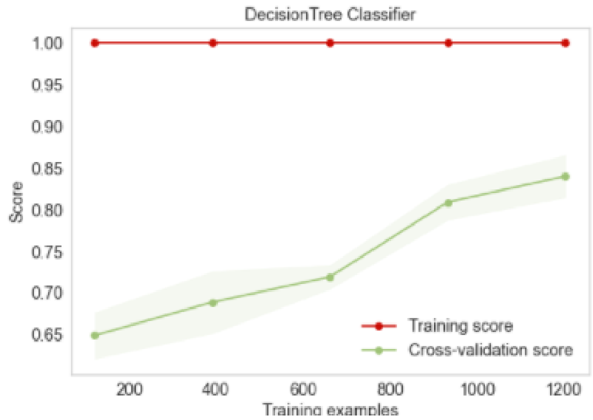
By **Logistic Regression Model,** we were able to get the accuracy score of 0.5685840707964602Cross_Validation_Score: 0.5650810763239533

Decision Tree Classifier Model:

Accuracy Score 0.8008849557522124
SCORE 0.8008849557522124
Confusion metrics
[168 48]
[42 194]]
CLASSIFICATION REPORT
precision   recall f1-score   support
0.0   0.80   0.78   0.79   216
1.0   0.80   0.82   0.81   236
accuracy   0.80   452
macro avg   0.80   0.80   0.80   452
weighted avg   0.80   0.80   0.80   452
Cross_Val_Score 0.8426173241512839



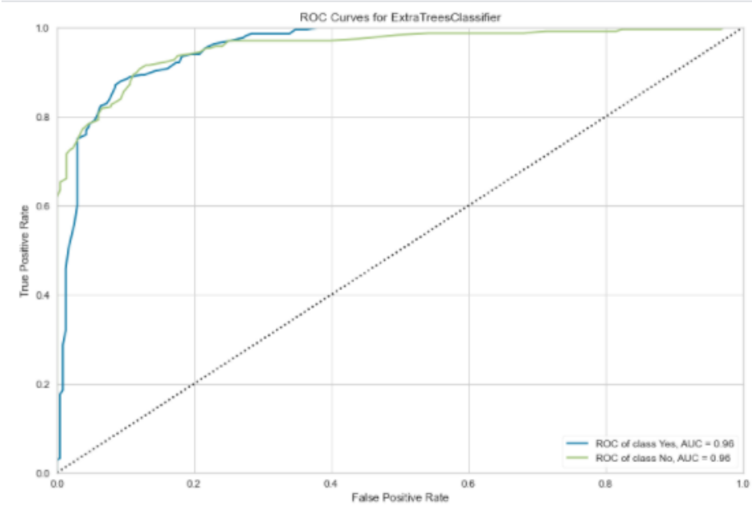
MODEL LEARNING CURVE



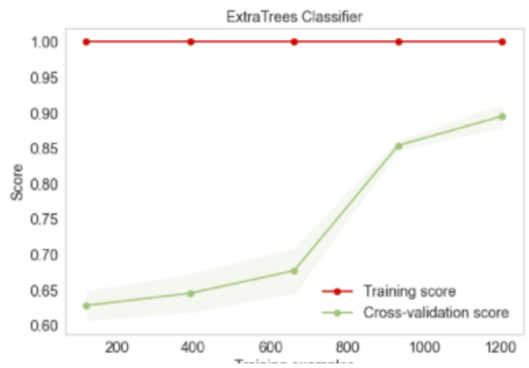
By **Decision Tree Classifier,** we get the accuracy score of 0.80088
Cross_Validation_Score: 0.8426

Extra Trees Classifier

Accuracy Score 0.8915929203539823
SCORE 0.8915929203539823
Confusion metrics
[191 25]
[24 212]
CLASSIFICATION REPORT
precision recall f1-score support
0.0 0.89 0.88 0.89 216
1.0 0.89 0.90 0.90 236
accuracy 0.89 452
macro avg 0.89 0.89 0.89 452
weighted avg 0.89 0.89 0.89 452
Cross_Val_Score 0.8877758465160281



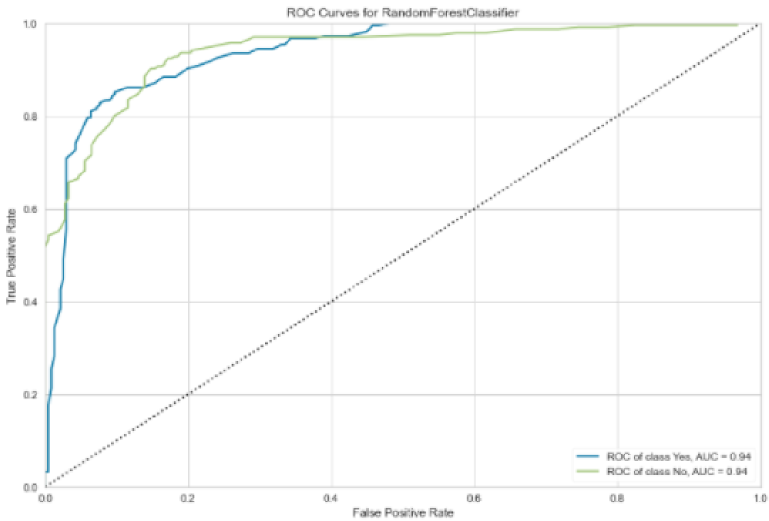
MODEL LEARNING CURVE



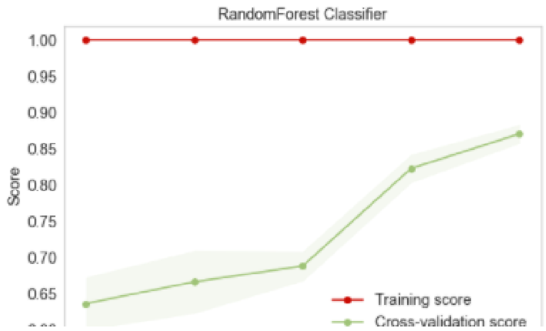
By **ExtraTrees Classifier model,** we were able to get the accuracy score of 0.8915929203539823. Cross_Validation_Score: 0.8877758465160281

Random Forest Classifier

Accuracy_Score 0.8628318584070797
SCORE 0.8628318584070797
Confusion metrics
[186 30]
[32 204]
CLASSIFICATION REPORT
precision recall f1-score support
0.0 0.85 0.86 0.86 216
1.0 0.87 0.86 0.87 236
accuracy 0.86 452
macro avg 0.86 0.86 0.86 452
weighted avg 0.86 0.86 0.86 452
Cross_Val_Score 0.8725066555191304



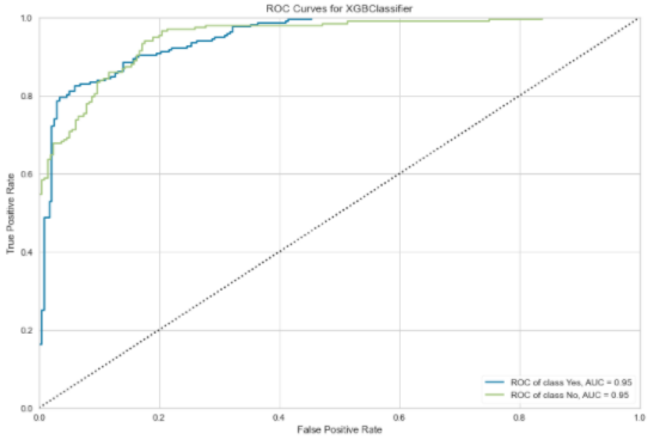
MODEL LEARNING CURVE

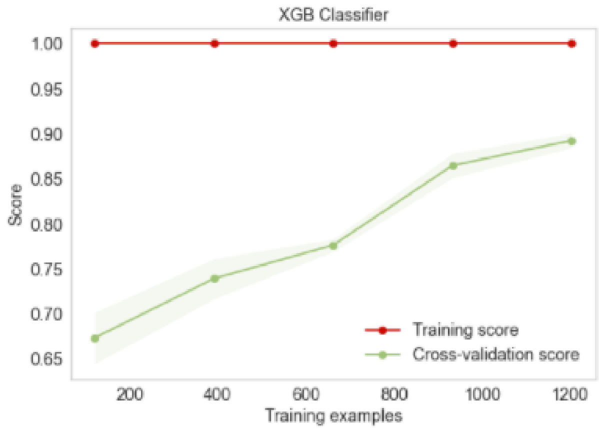


By **Random Forest Classifier,** we were able to get the accuracy score of 0.86283.
Cross_Validation_Score: 0.87250

XGB Classifier Model:

Accuracy_Score 0.8716814159292036
SCORE 0.8716814159292036
Confusion metrics
[180 36]
[22 214]
CLASSIFICATION REPORT
precision   recall f1-score   support
0.0   0.89   0.83   0.86   216
1.0   0.86   0.91   0.88   236
accuracy   0.87   452
macro avg   0.87   0.87   0.87   452
weighted avg   0.87   0.87   0.87   452
Cross_Val_Score 0.894411564102



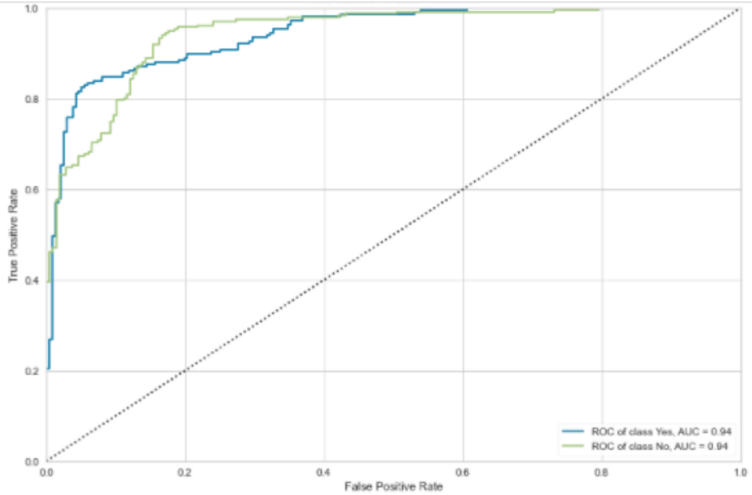


By **XGB Classifier model,** we were able to get the accuracy score of 0.87

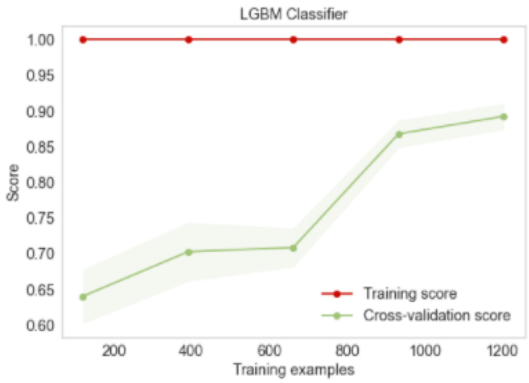
Cross_Validation_Score: 0.89

LGBM Classifier Model:

Accuracy_Score 0.8805309734513275
SCORE 0.8805309734513275
Confusion metrics
[183 33]
[21 215]
CLASSIFICATION REPORT
precision recall f1-score support
0.0 0.90 0.85 0.87 216
1.0 0.87 0.91 0.89 236
accuracy 0.88 452
macro avg 0.88 0.88 0.88 452
weighted avg 0.88 0.88 0.88 452
Cross_Val_Score 0.8857890915491409



MODEL LEARNING CURVE



By **LGBM model,** we were able to get the accuracy score of 0.88053.
Cross_Validation_Score: 0.8857.

XGM CLASSIFIER has high accuracy score" and with least difference between the Accuracy Score and the Cross validation score

"XGB Classifier" is our best model with 87.34 % Accuracy Score.

```
: XGB=XGBClassifier()
  XGB.fit(X_train,y_train)

  [03:01:36] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.(
  ult evaluation metric used with the objective 'binary:logistic' was changed fro
  u'd like to restore the old behavior.

: XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
                colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
                importance_type='gain', interaction_constraints='',
                learning_rate=0.300000012, max_delta_step=0, max_depth=6,
                min_child_weight=1, missing=nan, monotone_constraints='()',
                n_estimators=100, n_jobs=8, num_parallel_tree=1, random_state=0,
                reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
                tree_method='exact', validate_parameters=1, verbosity=None)

: XGB.get_params()

: {'objective': 'binary:logistic',
   'use_label_encoder': True,
   'base_score': 0.5,
   'booster': 'gbtree',
   'colsample_bylevel': 1,
   'colsample_bynode': 1,
   'colsample_bytree': 1,
   'gamma': 0,
   'gpu_id': -1,
   'importance_type': 'gain',
   'interaction_constraints': '',
   'learning_rate': 0.300000012,
   'max_delta_step': 0,
   'max_depth': 6,
   'min_child_weight': 1,
   'missing': nan,
   'monotone_constraints': '()',
   'n_estimators': 100,
   'n_jobs': 8,
   'num_parallel_tree': 1,
   'random_state': 0,
   'reg_alpha': 0,
   'reg_lambda': 1,
   'scale_pos_weight': 1,
   'subsample': 1,
   'tree_method': 'exact',
   'validate_parameters': 1,
   'verbosity': None}
```

Hyper-parameter tuning

```
params={
 'booster':['gbtree','dart'],
 'gamma': [0,1,2,3],
 'importance_type': ['gain','split'],
 'max_depth': [6,5,7],
 'n_estimators': [100,200,500],
}
```

```
Grid=GridSearchCV(estimator=XGB,param_grid=params,n_jobs=-1,cv=skf,scoring='accuracy')
```

```
Grid.fit(X_new,y_new)
```

```
[03:38:43] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGBoost 1.3.0, the d
ult evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if y
u'd like to restore the old behavior.
GridSearchCV(cv=StratifiedKFold(n_splits=5, random_state=None, shuffle=True),
             estimator=XGBClassifier(base_score=0.5, booster='gbtree',
                                     colsample_bylevel=1, colsample_bynode=1,
                                     colsample_bytree=1, gamma=0, gpu_id=-1,
                                     importance_type='gain',
                                     interaction_constraints='',
                                     learning_rate=0.300000012,
                                     max_delta_step=0, max_depth=6,
                                     min_child_weight=1, missing=nan,
                                     monoton...)',
                                     n_estimators=100, n_jobs=8,
                                     num_parallel_tree=1, random_state=0,
                                     reg_alpha=0, reg_lambda=1,
                                     scale_pos_weight=1, subsample=1,
                                     tree_method='exact', validate_parameters=1,
                                     verbosity=None),
             n_jobs=-1,
             param_grid={'booster': ['gbtree', 'dart'], 'gamma': [0, 1, 2, 3],
                         'importance_type': ['gain', 'split'],
                         'max_depth': [6, 5, 7],
                         'n_estimators': [100, 200, 500]},
             scoring='accuracy')
```

```
Grid.best_params_
```

```
{'booster': 'gbtree',
 'gamma': 3,
 'importance_type': 'gain',
 'max_depth': 7,
 'n_estimators': 100}
```

```
Grid.best_score_
```

```
0.8971045741567842
```

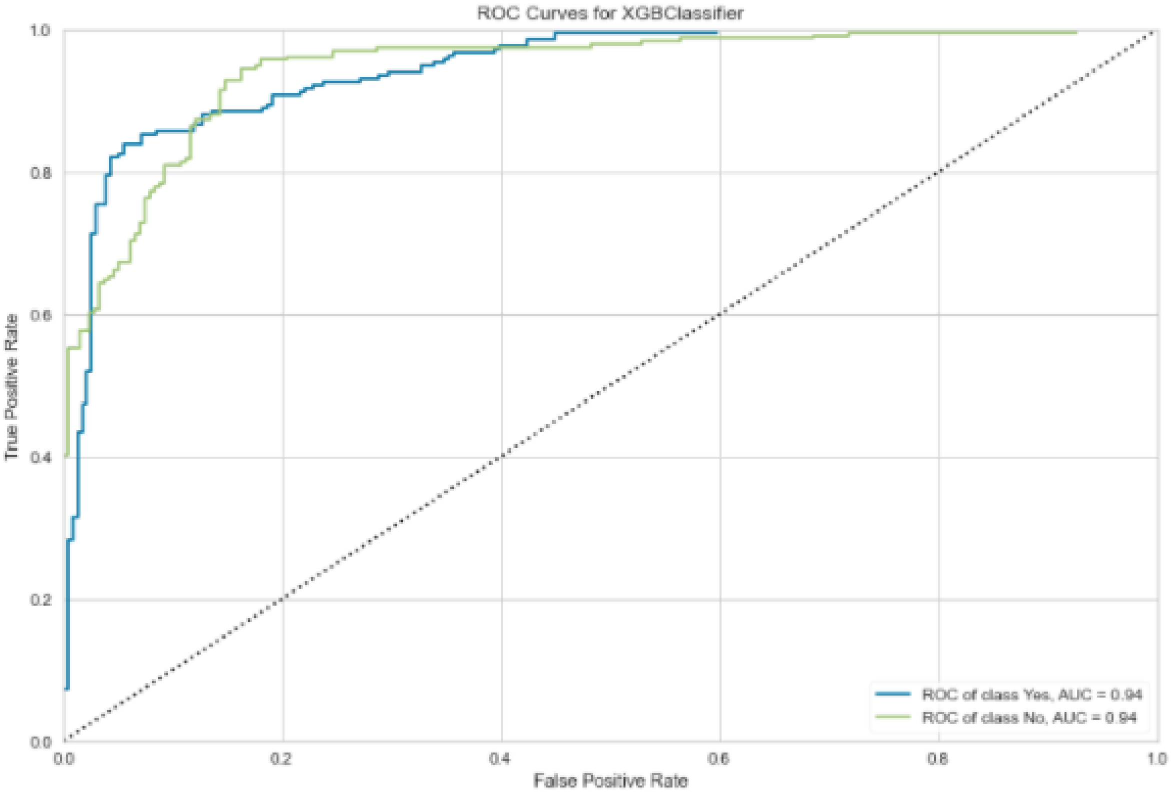
```
Xgb=XGBClassifier(booster= 'gbtree',
 gamma= 1,
 importance_type= 'gain',
 max_depth= 6,
 n_estimators= 100)
```

```
Xgb.fit(X_train,y_train)
```

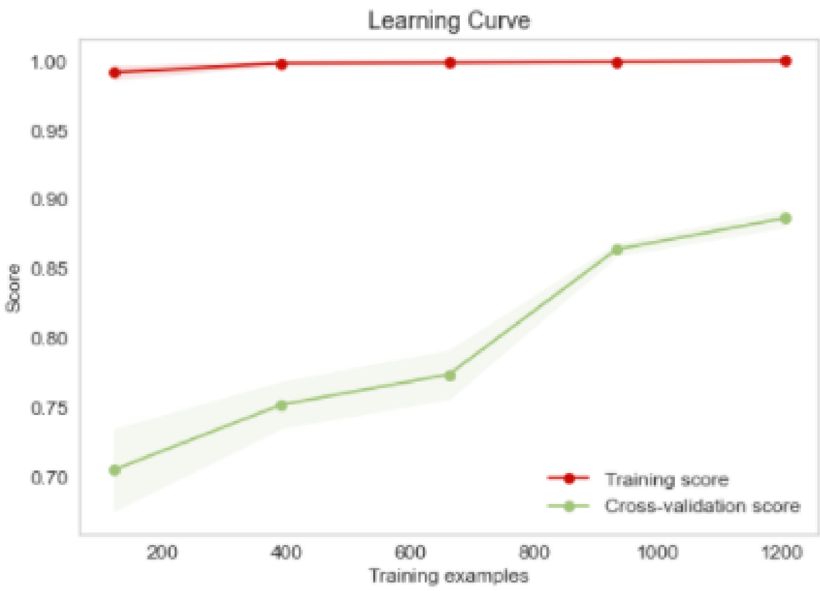
```
[03:41:33] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGBoost 1.3.0, the defa
ult evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if yo
u'd like to restore the old behavior.
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=1, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=8, num_parallel_tree=1, random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

```
print("ROC AUC CURVE")
plt.figure(figsize=(12,8))
roc_auc(Xgb,X_train,y_train,X_test=X_test,y_test=y_test,classes=['Yes','No'],micro=False,macro=False)
print("MODEL LEARNING CURVE")
skplt.estimators.plot_learning_curve(Xgb,X_new,y_new,cv=skf,scoring='accuracy')
plt.show()
```

```
ROC AUC CURVE
```




```
MODEL LEARNING CURVE
```



Final model metrics

```
y_predicted=Xgb.predict(X_test)
print("Accuracy_score",accuracy_score(y_test,y_predicted))
print("CVS",cross_val_score(Xgb,X_new,y_new,scoring='accuracy',cv=skf).mean())
print("Confusion metrics")
print('\n')
print(confusion_matrix(y_test,y_predicted))
print('\n')
print("Classification Report")
print("\n")
print(classification_report(y_test,y_predicted))
```

```
CVS 0.8951046181602166
Confusion metrics


[[184  32]
 [ 17 219]]


Classification Report


              precision    recall  f1-score   support

         0.0       0.92      0.85      0.88       216
         1.0       0.87      0.93      0.90       236

    accuracy                           0.89       452
   macro avg       0.89      0.89      0.89       452
weighted avg       0.89      0.89      0.89       452
```

Saving model

```
import joblib
joblib.dump(Grid,"Automobile_insurance_fraud.obj")
```

```
['Automobile_insurance_fraud.obj']
```

CONCLUSION

As you can see, financial fraud and machine learning are practically inseparable at present times. By applying various rules and synthetic algorithms, it becomes just a perfect technology for automated financial fraudulent detection.

Unlike the traditional system of analysis, which is mostly performed by human decisions, it allows covering much more information and processes the big data in shorter periods of time, thus saving lots of investments, resources, and time for the financial units.

Fraud detection using machine learning allows creating new rules and more complex algorithms for analyzing various transactions and suspicious financial behavior thus minimizing the risks of financial loss.

For the detailed coding please go through the below direct link which relates to the coding part of our model

https://github.com/shubhamshuklaa/INTERSHIP-17-FLIP-ROBO-/blob/main/Automobile_insurance_fraud.ipynb

Our model is now ready to predict the "Fraud Insurance Claim" with 89.51% Accuracy

THANK-YOU