Blog -2

Detecting Insurance Fraudulent Claim:

Using Decision Tree Classifier

Batch Id :1829

1. Problem Definition:

Insurance fraud is a huge problem in the industry. It's difficult to identify fraud claims. Even though it rarely dominates the news cycle. But it does impact your wallet every day. About 75% of insurance industry professionals believe that 10% or more or all insurance claims have some element of fraud, according to the ***2020 Friss Insurance Fraud Report***. Some industry experts believe that frauds have nearly doubled during Covid-19.

With a rapid increase in the number of insurance claims, the fraud teams of the insurers have huge amount of task to deal with a busy workload even while facing disruption, where auto industry has hit hard. Machine Learning is in a unique position to help the Auto Insurance industry with this problem. It has the excellent ability to learn from historical fraud patterns and recognize them in future transactions.

2. Exploratory Data Analysis (EDA)

At first for classifying the insurance claim, 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. Dataset containing 1000 instances and 39 variables was obtained from GitHub Repository. In this part, we will try to gain some insights and a get familiar with the data.

2.1 Importing Libraries and Dataset

First import the required libraries like NumPy, pandas, matplotlib, and seaborn in our notebook. Then load the data set from CSV format and convert it into Data Frame.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import zscore
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from sklearn.model_selection import cross_val_score
import warnings
warnings.filterwarnings('ignore')
```

```
#Loading the dataset
data=pd.read_csv('Insurance-claim.csv')
data
```

2.2 **Attribute Information:**

```
# Gettign information about data set
ds.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
 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
```

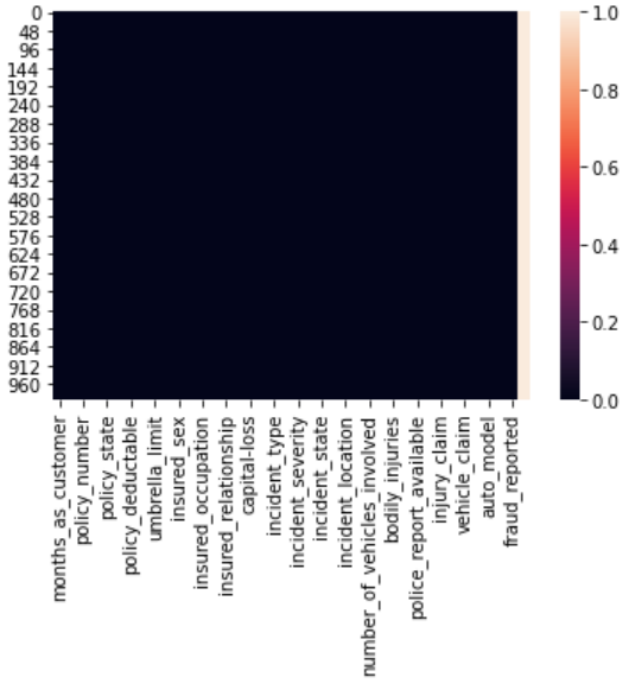
Key Observation: 1) We have 1000 rows and 40 columns in the dataset. 2) There are 21 object type variables and 19 numeric variables (3) The target variable is categorical.

2.3 **Missing Values:** There are no missing values in the data set.

Heat Map

```
# checking null values
ds.isnull().sum()
```

```
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
```



2.4 Statistical Summary

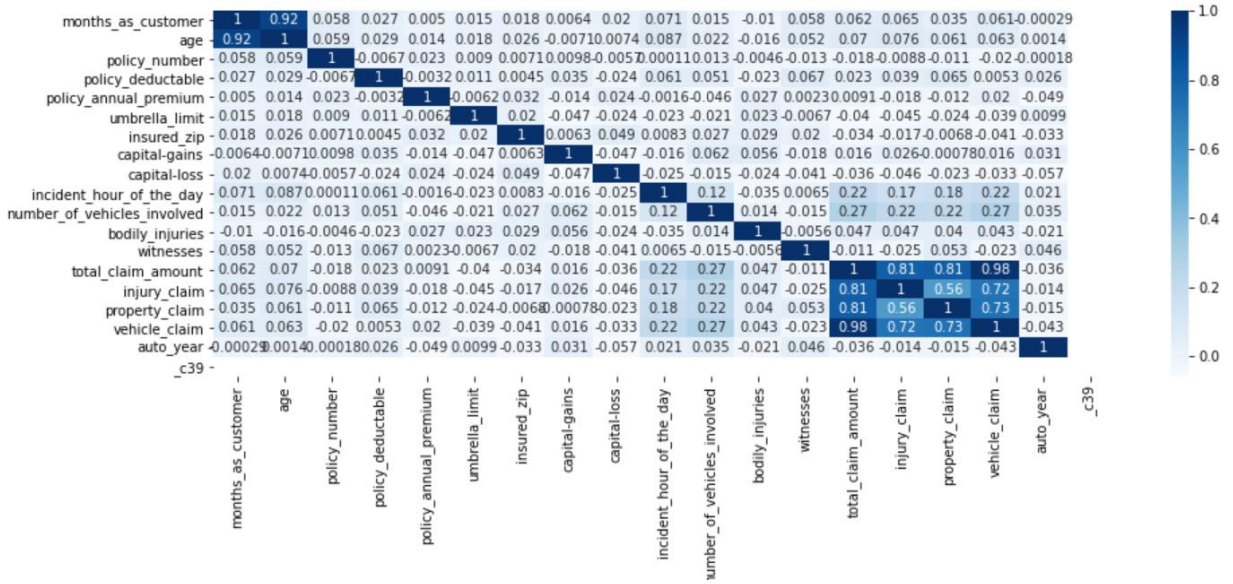
```
#Summary Statistics
ds.describe()
```

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

Key Observations:1) From the table above it can be seen that the column "months_as_customer", is not contributing to the data, so we can drop it. 2) The mean and median (50%) having similar values for almost all the variables. 3)There is a difference

between 75% & Max for most of the columns. It helps us to predict that outliers are absent from the data set.

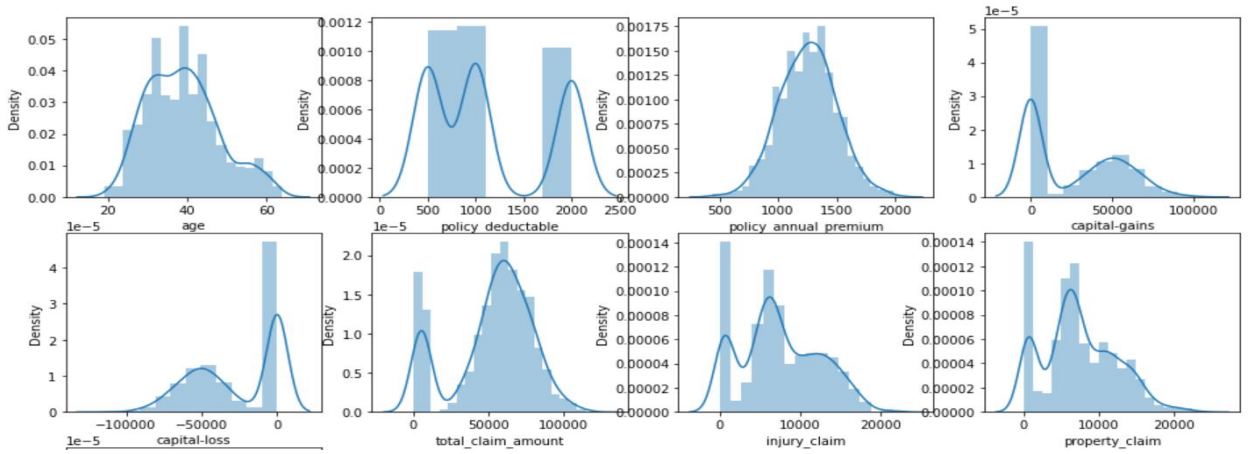
2.5 Correlation Matrix ---Pearson Method:



Key Observations: 1) *Injury_ claim* and *Vehicle_claim* are highly correlated 2) *Wittnes & Vehicle_claim* are negatively correlated 3) Part from it there don't seem to be much correlations in the data.

2.6 Checking the normalcy of the variables

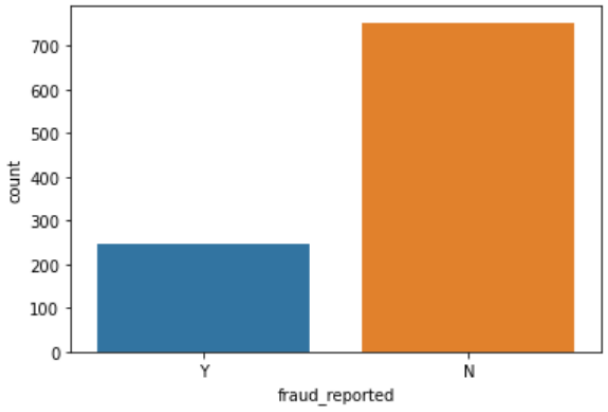
```
# Ploting the the hotogram to check the normal curve.
col=['age','policy_deductable','policy_annual_premium','capital-gains','capital-loss','total_claim_amount','injury_claim','proper
plt.figure(figsize=(16,14))
l=1
for i in col :
    plt.subplot(4,4,l)
    sns.distplot(ds[i])
    l=l+1
plt.show()
```



2.7 Checking Class Imbalance: There is a class imbalance in the target variable.

```
sns.countplot(x='fraud_reported',data=ds)
```

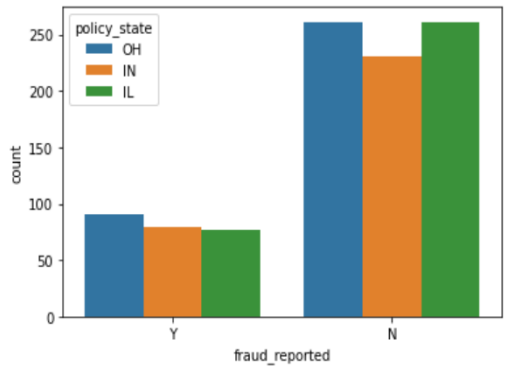
```
<AxesSubplot:xlabel='fraud_reported', ylabel='count'>
```



2.7 Checking count "Target variable vs Other variables"

```
# Ploting target variable against the policy_state
sns.countplot(x=ds['fraud_reported'],hue=ds['policy_state'],data=ds)
```

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




```
ds['insured_occupation'].value_counts()
```

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

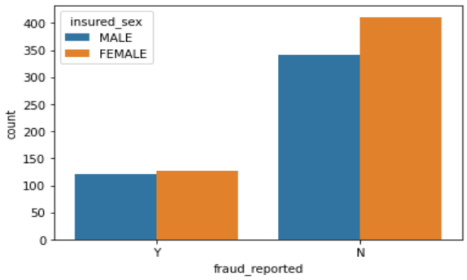
```
: ds['insured_sex'].value_counts()
```

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

Female are more in number then male.

```
sns.countplot(x=ds['fraud_reported'],hue='insured_sex',data=ds)
```

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



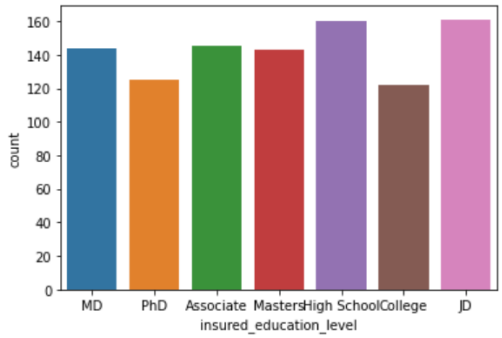
Key Observations:1) Most of the fraud cases have been reported from OH states then In state.

2) People who are working as machine-op-inspct has claimed more fraud transactions.

3) Male and female both have claimed same in number which are fraud

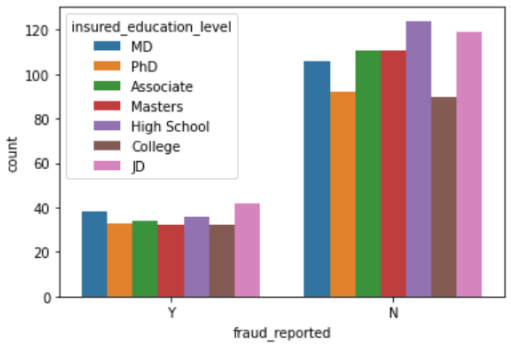
```
sns.countplot(x='insured_education_level',data=ds)
```

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

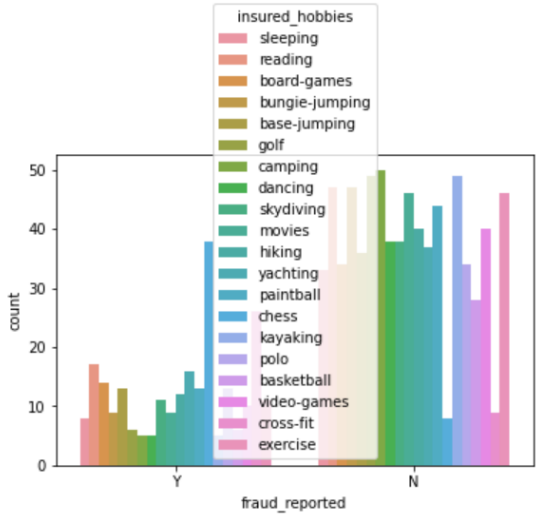



```
sns.countplot(x=ds['fraud_reported'],hue='insured_education
```

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




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




```
ds['incident_type'].value_counts()
```

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

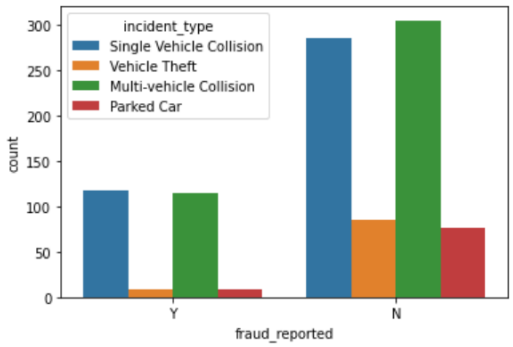
Key Observations1)The poeple with hobbies like Paintball,Yatching are reporting more fraud claims.

2) People with qualification MD,JD and High school have reported more fraud.

3) Most of the insured people have MD, PHD or Masters or JD as qualification.

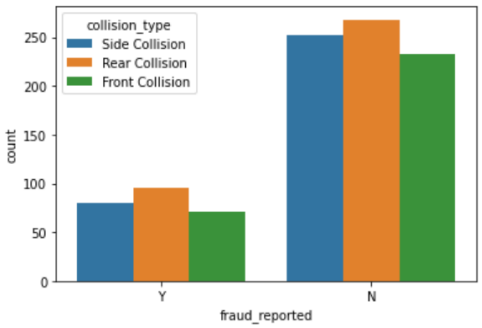
```
sns.countplot(x=ds['fraud_reported'],hue='incident_type'
```

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



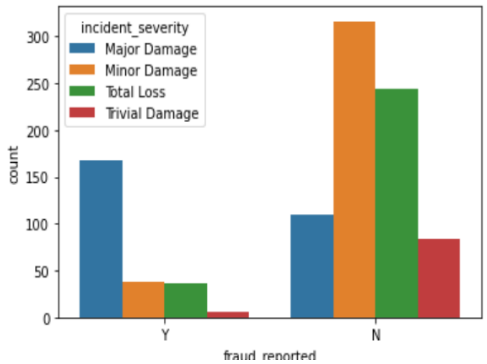

```
sns.countplot(x=ds['fraud_reported'],hue='collision_type',
```

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



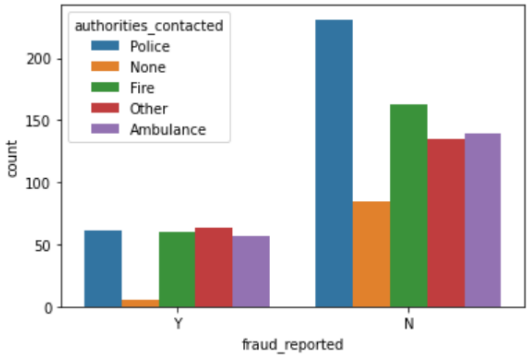

```
sns.countplot(x=ds['fraud_reported'],hue='incident_severity'
```

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




```
sns.countplot(x=ds['fraud_reported'],hue='authorities_
```

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



Key Observations:1) The people with Minor damage have reported second most more fraud claims. 2) The people who have not contacted the authorities have reported second most more fraud claims. 3) The people with rear collision incident type have reported more fraud claims

4)The people with incident type Single and Multi-vehicle collision have reported more fraud claims.

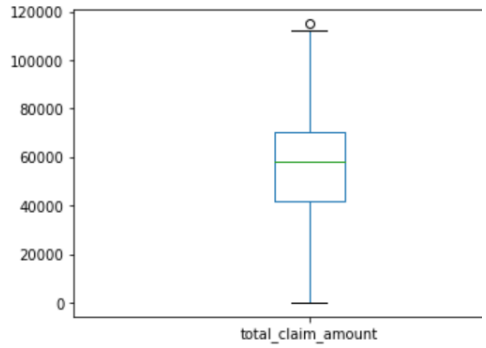
EDA Summary: The above graphs show that the features like incidence, education level, hobbies, age and incidence severity and their attributes have high correlation with the class of the claim. Secondly, the collision type , authority contacted have significant relationship with target variable.

3 Data Preprocessing :

3.1 Outliers detection and removal:

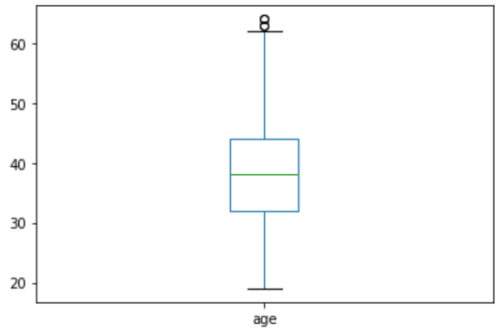
```
ds['total_claim_amount'].plot.box()
```

```
<AxesSubplot:>
```



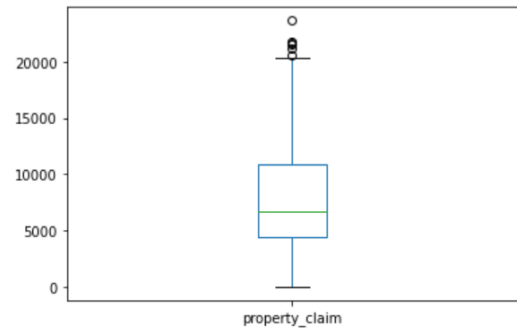

```
ds['age'].plot.box()
```

```
<AxesSubplot:>
```




```
ds['property_claim'].plot.box()
```

```
<AxesSubplot:>
```



There is outliers present in the above columns. Lets remove the outliers.

```
# Removing outliers
m=ds[['age','property_claim','total_claim_amount']]
```

```
from scipy.stats import zscore
z=np.abs(zscore(m))
z
threshold=3
print(np.where(z>3))
```

```
(array([500], dtype=int64), array([1], dtype=int64))
```

```
# lets check the presence of outlier
print('shape before remooving the outliers:',ds.shape)
dsnew=ds[(z<3).all(axis=1)]

print('shape after remooving the outliers:',dsnew.shape)
# here one rows will be droped
```

```
shape before remooving the outliers: (1000, 40)
shape after remooving the outliers: (999, 40)
```

3.2 Handling the variables having sign '?'

```
# Treating the variables having ?
dsnew['police_report_available']=dsnew['police_report_available'].replace({'?':np.nan})
dsnew['police_report_available']=dsnew['police_report_available'].fillna(method='ffill')
```

```
# Replacing the ? in collision_type varibale
dsnew['collision_type']=dsnew['collision_type'].replace({'?':np.nan})
dsnew['collision_type']=dsnew['collision_type'].fillna(method='ffill')
```

```
# Replacing the ? in property_damage varibale
dsnew['property_damage']=dsnew['property_damage'].replace({'?':np.nan})
dsnew['property_damage']=dsnew['property_damage'].fillna(method='ffill')
```

```
le since it is not contributing
er','_c39','policy_number','policy_bind_date','policy_csl','incident_date','incident_location','insured_zip','auto_year'],axis=1)
```

3.3 Handling the categorical Variables: Since we have all the variables categorical, we need to encode them first using label encoding method.

```
from sklearn.preprocessing import OrdinalEncoder
from sklearn.compose import make_column_transformer
oe =  OrdinalEncoder()
for i in dsnew.columns:
    if dsnew[i].dtypes=='object':
        dsnew[i]=oe.fit_transform(dsnew[i].values.reshape(-1,1))
dsnew
```

	age	policy_state	policy_deductable	policy_annual_premium	umbrella_limit	insured_sex	insured_education_level	insured_occupation
0	48	2.0	1000	1406.91	0	1.0	4.0	2.0
1	42	1.0	2000	1197.22	5000000	1.0	4.0	6.0
2	29	2.0	2000	1413.14	5000000	0.0	6.0	11.0
3	41	0.0	2000	1415.74	6000000	0.0	6.0	1.0
4	44	0.0	1000	1583.91	6000000	1.0	0.0	11.0
...	...	...	...	...	...	...	...	...
995	38	2.0	1000	1310.80	0	0.0	5.0	2.0
996	41	0.0	1000	1436.79	0	0.0	6.0	9.0
997	34	2.0	500	1383.49	3000000	0.0	5.0	1.0
998	62	0.0	2000	1356.92	5000000	1.0	0.0	5.0

4 Final Dataframe: After preprocessing, we now split the data into training/testing subsets.

```
x=dsnew.drop(columns=['fraud_reported'])
y=dsnew['fraud_reported']
```

```
x.shape
```

```
(999, 30)
```

```
y.shape
```

```
(999,)
```

4.1 Handling the class imbalance using the smote transformation

```
# Handling class imbalance using SMOTE
from imblearn.over_sampling import SMOTE
sm=SMOTE()
x_over,y_over = sm.fit_resample(x,y)
```

```
x_over.shape
```

```
(1504, 30)
```

```
y_over.shape
```

```
(1504,)
```

```
y_over.value_counts()
```

```
0.0    752
1.0    752
Name: fraud_reported, dtype: int64
```

4.2 Scaling dataset: using standard scalar transformation.

```
# Lets bring all feature into common scale
from sklearn.preprocessing import StandardScaler
scale= StandardScaler()
x=scale.fit_transform(x)
x
```

```
array([[ 1.02971224,  0.14012794, -0.19824983, ..., -0.67019486,
        -0.5143892 ,  2.03002809],
       [ 1.02971224,  0.14012794,  1.76587407, ..., -0.2504706 ,
        -1.31310821, -0.29572966],
       [-2.08704716,  0.14012794,  1.37304929, ..., -0.2504706 ,
        -1.31310821,  0.86714922],
       ...,
       [-0.8403434 ,  0.14012794, -0.19824983, ..., -1.50964337,
        -2.11182722,  0.28570978],
       [-0.21699152,  0.95327039, -0.19824983, ...,  1.42842641,
         0.28432981,  0.28570978],
       [ 1.02971224,  0.14012794, -0.19824983, ...,  0.16925365,
        -2.11182722,  0.28570978]])
```

4.3 To find the best random state using ***Logistic Regressor Model***

```
# To find the best random state using Logistic Regressor model
maxAccu=0
maxRS=0
for i in range(1,100):
    x_train,x_test,y_train,y_test=train_test_split(x_over,y_over,test_size=.30,random_state=i)
    mod= LogisticRegression()
    mod.fit(x_train,y_train)
    pred=mod.predict(x_test)
    acc=accuracy_score(y_test,pred)
    if acc>maxAccu:
        maxAccu=acc
        maxRS=i
print ('best accuracy is',maxAccu,'on random state',maxRS)
```

```
best accuracy is 0.7986725663716814 on random state 66
```

4.4 Train Test Split: split the data into training/testing subsets.

```
# Sending the data for train and test using Train_test_Split
# 30 % data will go for testing and 70% data will go for training the model
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.30,random_state=maxRS)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(699, 30)
(300, 30)
(699,)
(300,)
```

70% of the data (699 rows) will be available for training the model & 30% (300 rows) will be available for testing the model

5 Model Building & Evaluation

Since the target variable is categorical. we can build the classification models. Therefore our evaluation Matrices will be accuracy_score, confusion_matrix, classifcation_report, AUC-ROC curve.

We will be adopting Decision Tree Classification model.

```
# Decision Tree Classifier
dtc=DecisionTreeClassifier()
dtc.fit(x_train,y_train)
dtc.score(x_train,y_train)
preddtc=dtc.predict(x_test)
print(accuracy_score(y_test,preddtc))
print(confusion_matrix(y_test,preddtc))
print(classification_report(y_test,preddtc))
```

```
0.8133333333333334
[[197  32]
 [ 24  47]]
              precision    recall  f1-score   support

         0.0       0.89      0.86      0.88       229
         1.0       0.59      0.66      0.63        71

    accuracy                           0.81       300
   macro avg       0.74      0.76      0.75       300
weighted avg       0.82      0.81      0.82       300
```

The Decision Tree Classification model gives 81% accuracy and good f1 and precision scores.

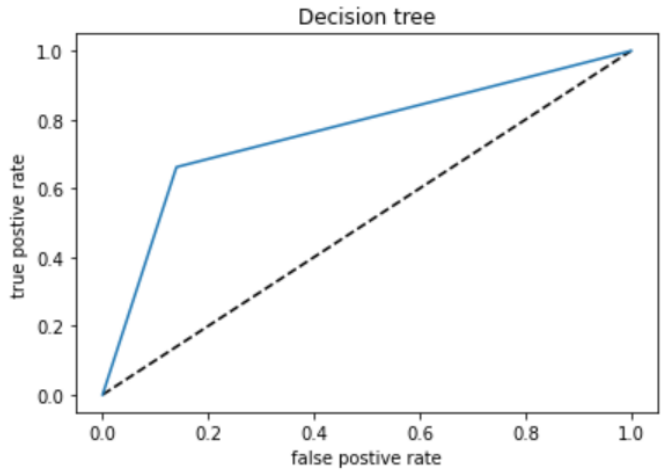
5.1 **Cross Validation:**

```
dtcscores=cross_val_score(dtc,x,y,cv=5)
print(dtcscores)
print(dtcscores.mean(),dtcscores.std())
```

```
[0.74       0.755      0.755      0.785      0.76884422]
0.7607688442211056 0.015167495770771294
```

It can be seen from the above that the accuracy score of the Decision Tree Classification model is similar before and after the cross validation.

5.2 **AUC-ROC Curve:**




```
0.76111691985977
```

We can depict from the above figure that the are under the AUC-ROC curve is 1 for the model which means the model is 100% efficient for detecting the insurance claim fraud.

5.3 Hyper Parameter Tuning

```
from sklearn.model_selection import GridSearchCV
parameter={'max_depth':np.arange(5,15),'criterion':['gini','entropy'],'max_features':['auto', 'sqrt'],'random_state':np.arange(4
```

```
GCV=GridSearchCV(DecisionTreeClassifier(),parameter,cv=5)
```

```
GCV.best_params_
```

```
{'criterion': 'entropy',
 'max_depth': 5,
 'max_features': 'auto',
 'random_state': 66}
```

```
# Decision Tree Classifier
final=DecisionTreeClassifier(criterion= 'entropy',max_depth= 10,max_features= 'auto',random_state= 100)
final.fit(x_train,y_train)
final.score(x_train,y_train)
preddtc=final.predict(x_test)
print(accuracy_score(y_test,preddtc))
print(confusion_matrix(y_test,preddtc))
print(classification_report(y_test,preddtc))
```

```
0.7833333333333333
[[194  35]
 [ 30  41]]
              precision    recall  f1-score   support

         0.0       0.87      0.85      0.86       229
         1.0       0.54      0.58      0.56        71

    accuracy                           0.78       300
   macro avg       0.70      0.71      0.71       300
weighted avg       0.79      0.78      0.79       300
```

This is the final model for the deployment purpose.

6 Conclusion

We have trained a good model with perfect score for predicting the auto industry insurance fraud claim. Firstly, we tried to understand the key attributes that helps in the better classification of claim by EDA and preprocessing of the data. Some of the features like incidence, education level, hobbies, age and incidence severity have contributed the most for training the model with good accuracy. The Decision Tree Classifier model has given the good accuracy score 81% and area under AUC-ROC curve 0.76. Proposed model predicts whether the claimed insurance is “FRAUD” or “GENUINE”. Thus, helping the insurance companies to spot frauds with fewer amount of time and with good accuracy rate. Our future work shall focus on extracting more data from insurance policy and individual’s health history and track record about the safe driving or rough driving and try to expand the database to improve the classification process.