**Insurance Claims- Fraud Detection**

# **Problem Statement:**

**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.

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

**SOLUTION WITH STEP BY STEP HOW CLASSIFICATION MODEL WAS PREDICTED**

As per the requirements of the data to check whether the fraud has happens or not we have to perform the following steps in detailed below:

**STEP1:-** First should import the required libraries for the data set of **"Insurance Claim Fraud Detection":**

import pandas as pd (For creating Data Frame)

import numpy as np (For Arrays & Numerical data)

import matplotlib.pyplot as plt (for plotting and EDA)

import seaborn as sns (for plotting and EDA)

from sklearn.linear\_model import LogisticRegression (classification model)

from sklearn.tree import DecisionTreeClassifier (classification model)

from sklearn.ensemble import RandomForestClassifier (classification model)

from sklearn.svm import SVC (classification model)

from sklearn.model\_selection import train\_test\_split (for splitting dataset into training) and testing

from sklearn.metrics import accuracy\_score ( for checking model accuracy)

from sklearn.metrics import roc\_auc\_score (for checking model accuracy)

from sklearn.metrics import confusion matrix, classification report (for checking model accuracy)

from sklearn.model\_selection import cross val\_score (for cross validating) model

import warnings

warnings.filterwarnings('ignore')

**STEP2:- Now loading the Data Set:**

dx=pd.read\_csv("https://raw.githubusercontent.com/dsrscientist/Data-Science-ML-Capstone-Projects/master/Automobile\_insurance\_fraud.csv")

(Here data set was loaded by assigning a variable i.e dx).

STEP3:- **Now checking the top (5) Rows of Data Set:**

dx.head(). (The head() means to display top 5 Rows).

| **months\_as\_customer** | **age** | **policy\_number** | **policy\_bind\_date** | **policy\_state** | **policy\_csl** | **policy\_deductable** | **policy\_annual\_premium** | **umbrella\_limit** | **insured\_zip** | **...** | **police\_report\_available** | **total\_claim\_amount** | **injury\_claim** | **property\_claim** | **vehicle\_claim** | **auto\_make** | **auto\_model** | **auto\_year** | **fraud\_reported** | **\_c39** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 328 | 48 | 521585 | 17-10-2014 | OH | 250/500 | 1000 | 1406.91 | 0 | 466132 | ... | YES | 71610 | 6510 | 13020 | 52080 | Saab | 92x | 2004 | Y | NaN |
| **1** | 228 | 42 | 342868 | 27-06-2006 | IN | 250/500 | 2000 | 1197.22 | 5000000 | 468176 | ... | ? | 5070 | 780 | 780 | 3510 | Mercedes | E400 | 2007 | Y | NaN |
| **2** | 134 | 29 | 687698 | 06-09-2000 | OH | 100/300 | 2000 | 1413.14 | 5000000 | 430632 | ... | NO | 34650 | 7700 | 3850 | 23100 | Dodge | RAM | 2007 | N | NaN |
| **3** | 256 | 41 | 227811 | 25-05-1990 | IL | 250/500 | 2000 | 1415.74 | 6000000 | 608117 | ... | NO | 63400 | 6340 | 6340 | 50720 | Chevrolet | Tahoe | 2014 | Y | NaN |
| **4** | 228 | 44 | 367455 | 06-06-2014 | IL | 500/1000 | 1000 | 1583.91 | 6000000 | 610706 | ... | NO | 6500 | 1300 | 650 | 4550 | Accura | RSX | 2009 | N | NaN |

5 rows × 40 columns

STEP5:- **Displaying the columns present in Data Set:-**

dx.columns. (It displays all the columns present in Data Numerical & Categorical).

Index(['months\_as\_customer', 'age', 'policy\_number', 'policy\_bind\_date',

'policy\_state', 'policy\_csl', 'policy\_deductable',

'policy\_annual\_premium', 'umbrella\_limit', 'insured\_zip', 'insured\_sex',

'insured\_education\_level', 'insured\_occupation', 'insured\_hobbies',

'insured\_relationship', 'capital-gains', 'capital-loss',

'incident\_date', 'incident\_type', 'collision\_type', 'incident\_severity',

'authorities\_contacted', 'incident\_state', 'incident\_city',

'incident\_location', 'incident\_hour\_of\_the\_day',

'number\_of\_vehicles\_involved', 'property\_damage', 'bodily\_injuries',

'witnesses', 'police\_report\_available', 'total\_claim\_amount',

'injury\_claim', 'property\_claim', 'vehicle\_claim', 'auto\_make',

'auto\_model', 'auto\_year', 'fraud\_reported', '\_c39'],

dtype='object')

**EDA (EXTRAPOLATORY DATA ANALYSIS) PART STARS**

This Dataset consists of **1000** entries with **40** different columns which are a mixture of Object, Int and Float data types with our target variable ‘**fraud\_reported’** being object data type, hence it’s a classification problem. We will go through the data and build a model which will predict whether the value for '**fraud\_reported'** will be ‘Yes’ or ‘No’, therefore, classification problem!

NOW FINDING DATA TYPES

'**dx.dtypes'.**

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

(So the columns in Data set having different types of Data types in (Float, Int &Object) respectively).

**NOW FINDING DATA SET SHAPE:**

**"dx.shape"**

(1000, 40) (Contains 1000 Rows and 40 Columns as it mentioned earlier also).

**NOW CHECKING THE MISSING DATA /NULL VALUES:-**

**"dx.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

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

Here by clear observation found that column "\_c39" Full Data is missing so will remove this. Which follow in next step.

**NOW DROPPING A COLUMN FROM DATA SET:-**

**"dx.drop(["\_c39"], axis=1, inplace=True)"**

**NOW SEPERATING CATEGORICAL & NUMERICAL VALUES:-**

I Separated this Columns which will be easy for detecting skewness ,Outliers and many more.

**s = (dx.dtypes == 'object')**

**cate\_cols = list(s[s].index)**

**print("Categorical variables:")**

**print(cate\_cols)**

Categorical variables:

['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']

**s = (dx.dtypes == 'int64')**

**cont\_cols = list(s[s].index)**

**print("Continuous variables:")**

**print(cont\_cols)**

Continuous variables:

['months\_as\_customer', 'age', 'policy\_number', 'policy\_deductable', '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']

(By this Successfully separated the '**Categorical variables' & 'Continuous variables')**

**NOW PRESENTING THE STASTICAL SUMMARY:-**

**"dx[cont\_cols].describe()"**

| **months\_as\_customer** | **age** | **policy\_number** | **policy\_deductable** | **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\_annual\_premium** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **count** | 1000.000000 | 1000.000000 | 1000.000000 | 1000.000000 | 1.000000e+03 | 1000.000000 | 1000.000000 | 1000.000000 | 1000.000000 | 1000.00000 | 1000.000000 | 1000.000000 | 1000.00000 | 1000.000000 | 1000.000000 | 1000.000000 | 1000.000000 | 1000.000000 |
| **mean** | 203.954000 | 38.948000 | 546238.648000 | 1136.000000 | 1.101000e+06 | 501214.488000 | 25126.100000 | -26793.700000 | 11.644000 | 1.83900 | 0.992000 | 1.487000 | 52761.94000 | 7433.420000 | 7399.570000 | 37928.950000 | 2005.103000 | 1256.406150 |
| **std** | 115.113174 | 9.140287 | 257063.005276 | 611.864673 | 2.297407e+06 | 71701.610941 | 27872.187708 | 28104.096686 | 6.951373 | 1.01888 | 0.820127 | 1.111335 | 26401.53319 | 4880.951853 | 4824.726179 | 18886.252893 | 6.015861 | 244.167395 |
| **min** | 0.000000 | 19.000000 | 100804.000000 | 500.000000 | -1.000000e+06 | 430104.000000 | 0.000000 | -111100.000000 | 0.000000 | 1.00000 | 0.000000 | 0.000000 | 100.00000 | 0.000000 | 0.000000 | 70.000000 | 1995.000000 | 433.330000 |
| **25%** | 115.750000 | 32.000000 | 335980.250000 | 500.000000 | 0.000000e+00 | 448404.500000 | 0.000000 | -51500.000000 | 6.000000 | 1.00000 | 0.000000 | 1.000000 | 41812.50000 | 4295.000000 | 4445.000000 | 30292.500000 | 2000.000000 | 1089.607500 |
| **50%** | 199.500000 | 38.000000 | 533135.000000 | 1000.000000 | 0.000000e+00 | 466445.500000 | 0.000000 | -23250.000000 | 12.000000 | 1.00000 | 1.000000 | 1.000000 | 58055.00000 | 6775.000000 | 6750.000000 | 42100.000000 | 2005.000000 | 1257.200000 |
| **75%** | 276.250000 | 44.000000 | 759099.750000 | 2000.000000 | 0.000000e+00 | 603251.000000 | 51025.000000 | 0.000000 | 17.000000 | 3.00000 | 2.000000 | 2.000000 | 70592.50000 | 11305.000000 | 10885.000000 | 50822.500000 | 2010.000000 | 1415.695000 |
| **max** | 479.000000 | 64.000000 | 999435.000000 | 2000.000000 | 1.000000e+07 | 620962.000000 | 100500.000000 | 0.000000 | 23.000000 | 4.00000 | 2.000000 | 3.000000 | 114920.00000 | 21450.000000 | 23670.000000 | 79560.000000 | 2015.000000 | 2047.590000 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**Observations:**

Count for all the columns is 1000, hence there are no missing values in these columns.

The mean of months\_as\_customer is 203.95 and the mode is 199.50. The range is 0 - 479 and IQR is 115.75 - 276.25.

The mean of age is 38.95 and the mode is 38. The range of age is 19 - 64 and IQR is 32 - 44.

The mean of policy\_deductable is 1136 and the mode is 1000. The range of policy\_deductable is 500 - 2000 and IQR is also 500 - 2000.

Mode of umbrella\_limit is 0 and range is -1000000 - 1000000.

The mean of capital-gains is 25126.10 and the mode is 0. The range is 0 - 100500.

The mean of capital-loss is -26793.70 and the mode is -23250.00. The range is -111100.00 - 0.

The mode of incident\_hour of the day is 12.

The mean of total\_claim\_amount is 52761.94 and the mode is 58055.00. The range is 100 - 114920.00 and IQR is 41812.50 - 70592.50.

The mean of injury\_claims is 7433.42 and the mode is 6775.00. The range is 0 - 21450 and IQR is 4295.00 - 11305.00.

The mean of property\_claim is 7399.57 and the mode is 6750.00. The range is 0 - 23670.00 and IQR is 4445.00 - 10885.00.

The mean of vehicle\_claim is 37928.95 and the mode is 42100.00. The range is 70 - 79560.00 and IQR is 30292.50 - 50822.50.

The mode of auto\_year is 2005 and the range is 1995 - 2015.

The mean of policy\_annual\_premium is 1256.41 and the mode is 1257.20. The range is 433.33 - 2047.59 and IQR is 1089.61 - 1415.69.

**NOW AGAIN CHECKING MISSING VALUES FOR THIS CATERGORICAL DATA:-**

**"dx.isnull().sum()"**

onths\_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

dtype: int64

(There is no missing Data is present)

**DISTRUBUTION PLOT FOR CATERGORICAL COLUMNS:-**

**for i in dx[cont\_cols]:**

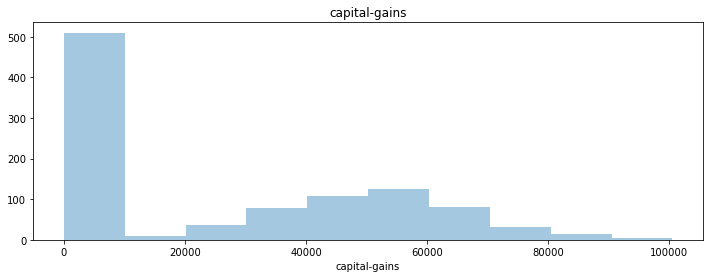
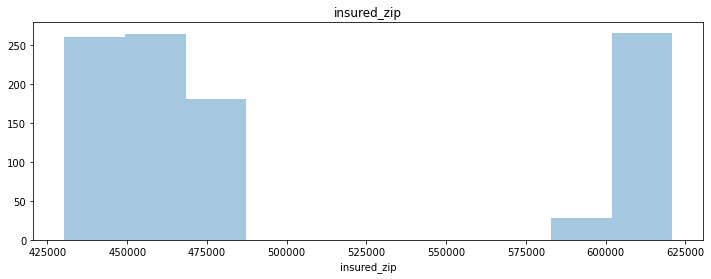
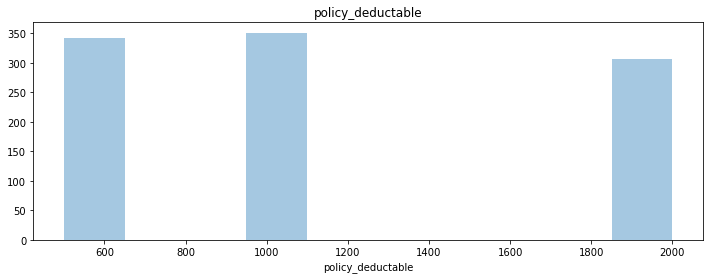
**plt.figure(figsize = (12, 4))**

**sns.distplot(dx[i], bins = 10, kde = False)**

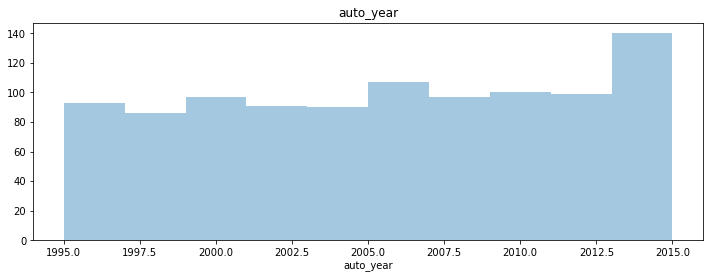
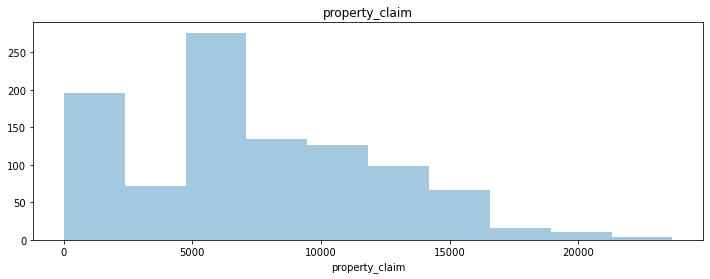
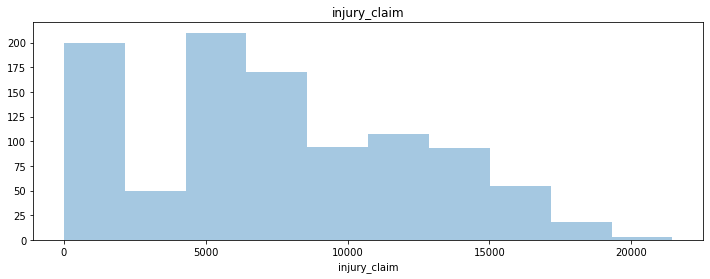
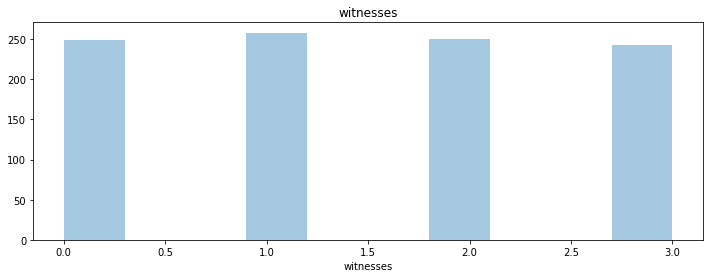
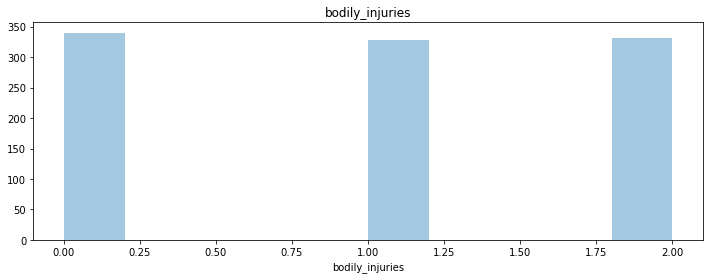
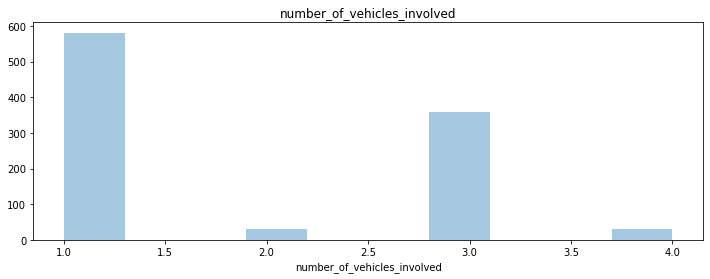
**plt.title(i)**

**plt.show()**

****

****

****

****

**Observations:-**

In some columns Data is Normally distributed and in some columns there is skweness is present in Data.

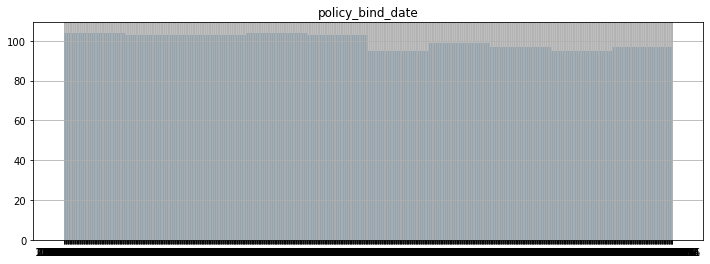
**NOW CHECK THROUGH HISTOGRAM:-**

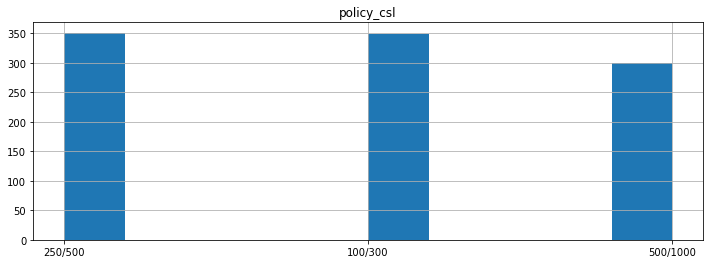
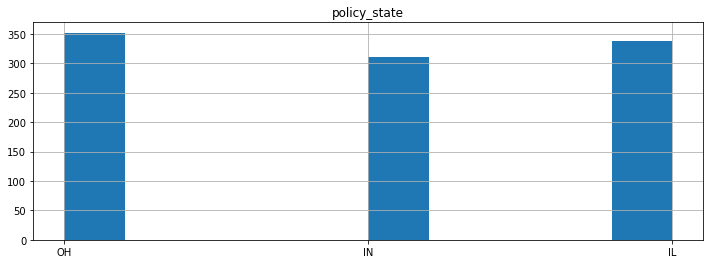
**for i in cate\_cols:**

**plt.figure(figsize = (12, 4))**

**dx[i].hist(grid = True)**

**plt.title(i)**

**plt.show()**



**Observations:-**

(Here for Columns only Visualization for getting Basic Idea about How the Data present).

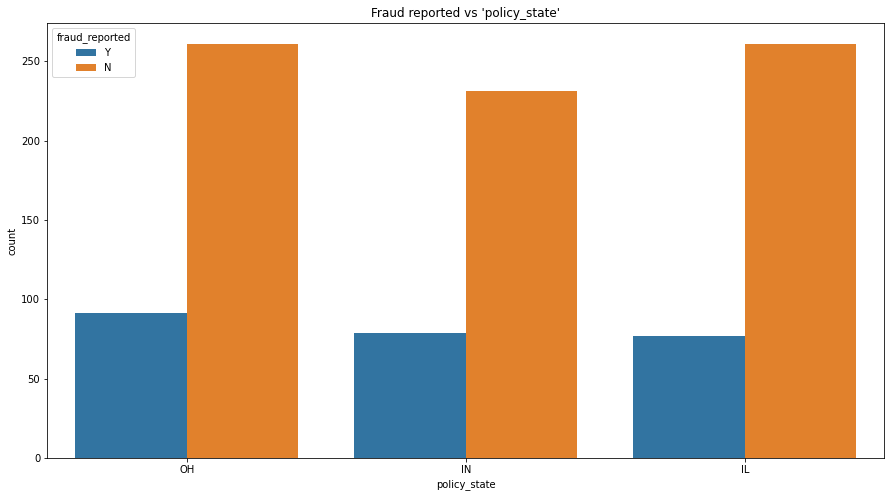
**NOW CHECKING THROUGH COUNT PLOT:-**

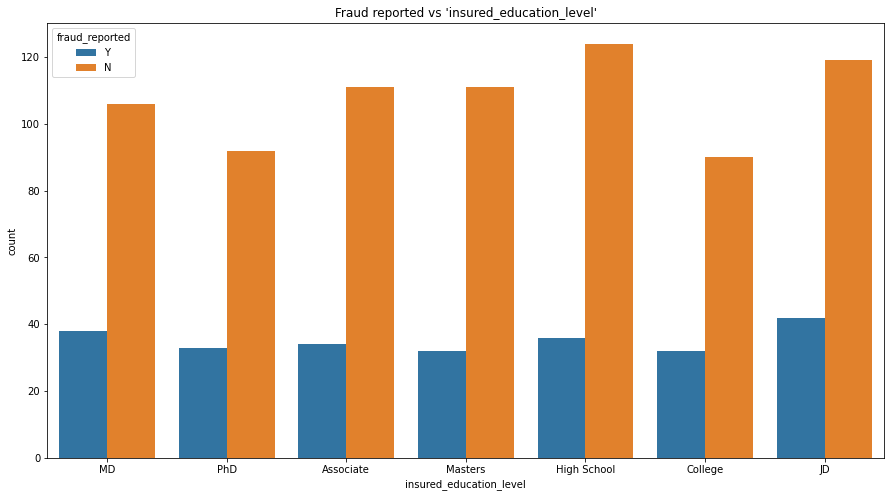
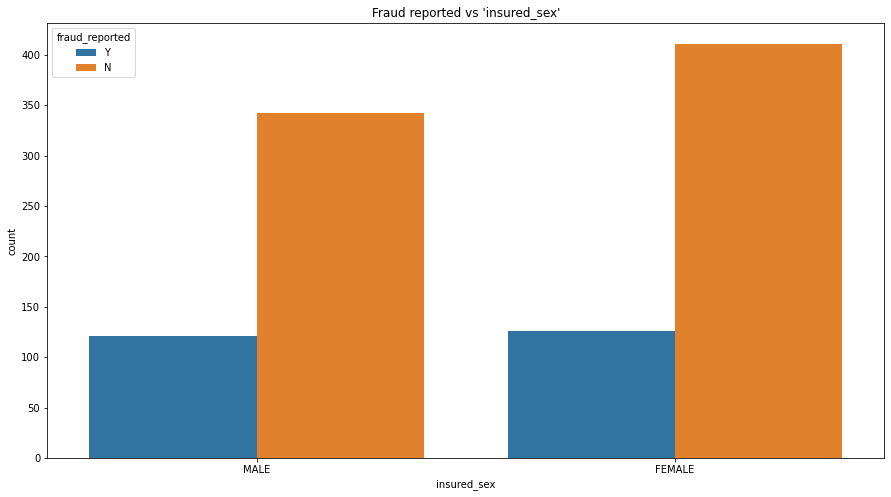
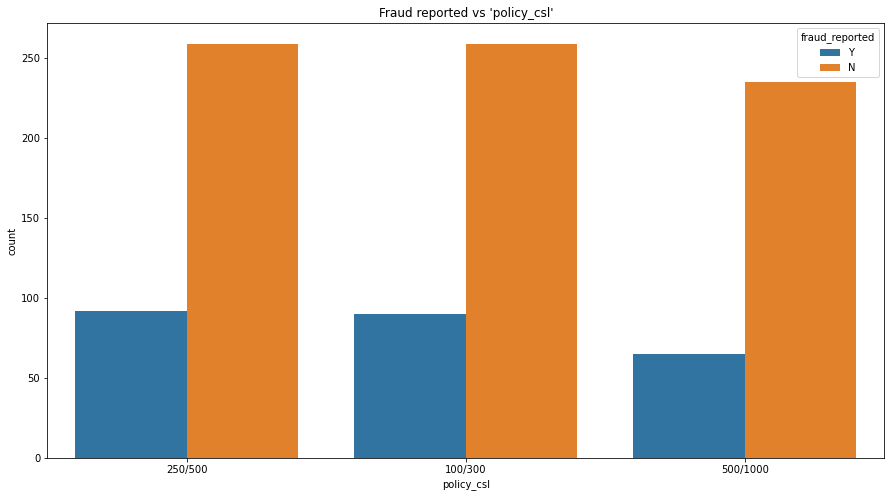
**"for i in cate\_cols[:-1]:**

**plt.figure(figsize=(15, 8))**

**plt.title("Fraud reported vs '%s'"%i)**

**sns.countplot(dx[i],hue=dx['fraud\_reported'])"**





**NOW CHECKING THROUGH STRIP PLOT:-**

for i in cont\_cols:

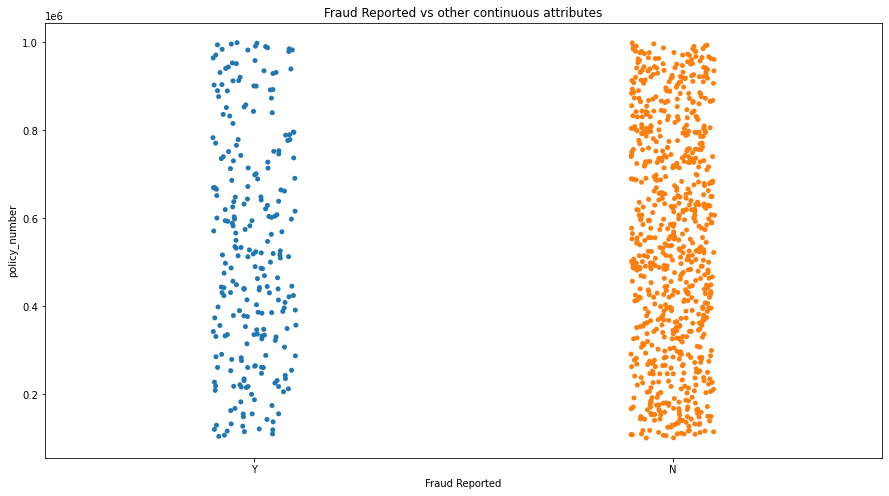
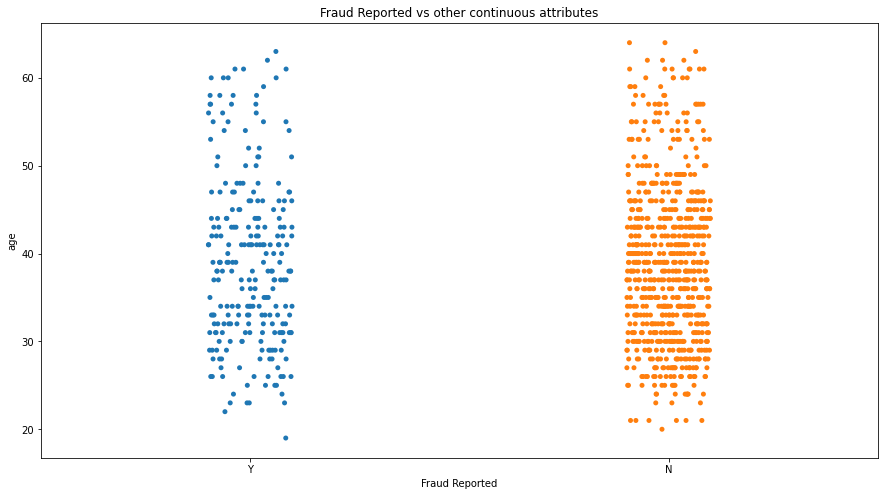
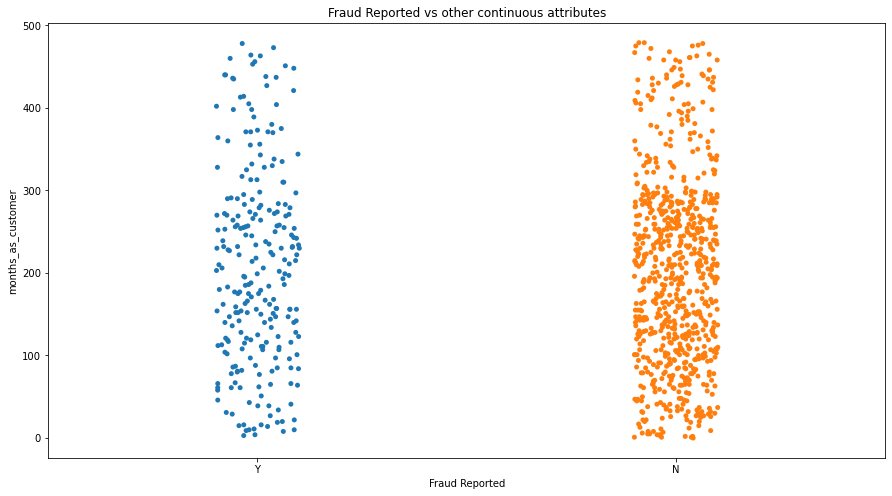
plt.figure(figsize = (15, 8))

ax = sns.stripplot(dx['fraud\_reported'], dx[i])

ax.set(xlabel = 'Fraud Reported', ylabel = i)

plt.title('Fraud Reported vs other continuous attributes')

plt.show()



Note:-

Hence the Target Column " fraud\_reported" is catergorical column it will replaced by Some Data.

**"dx['fraud\_reported'].replace('N', 0, inplace = True)**

**dx['fraud\_reported'].replace('Y', 1, inplace = True)"**

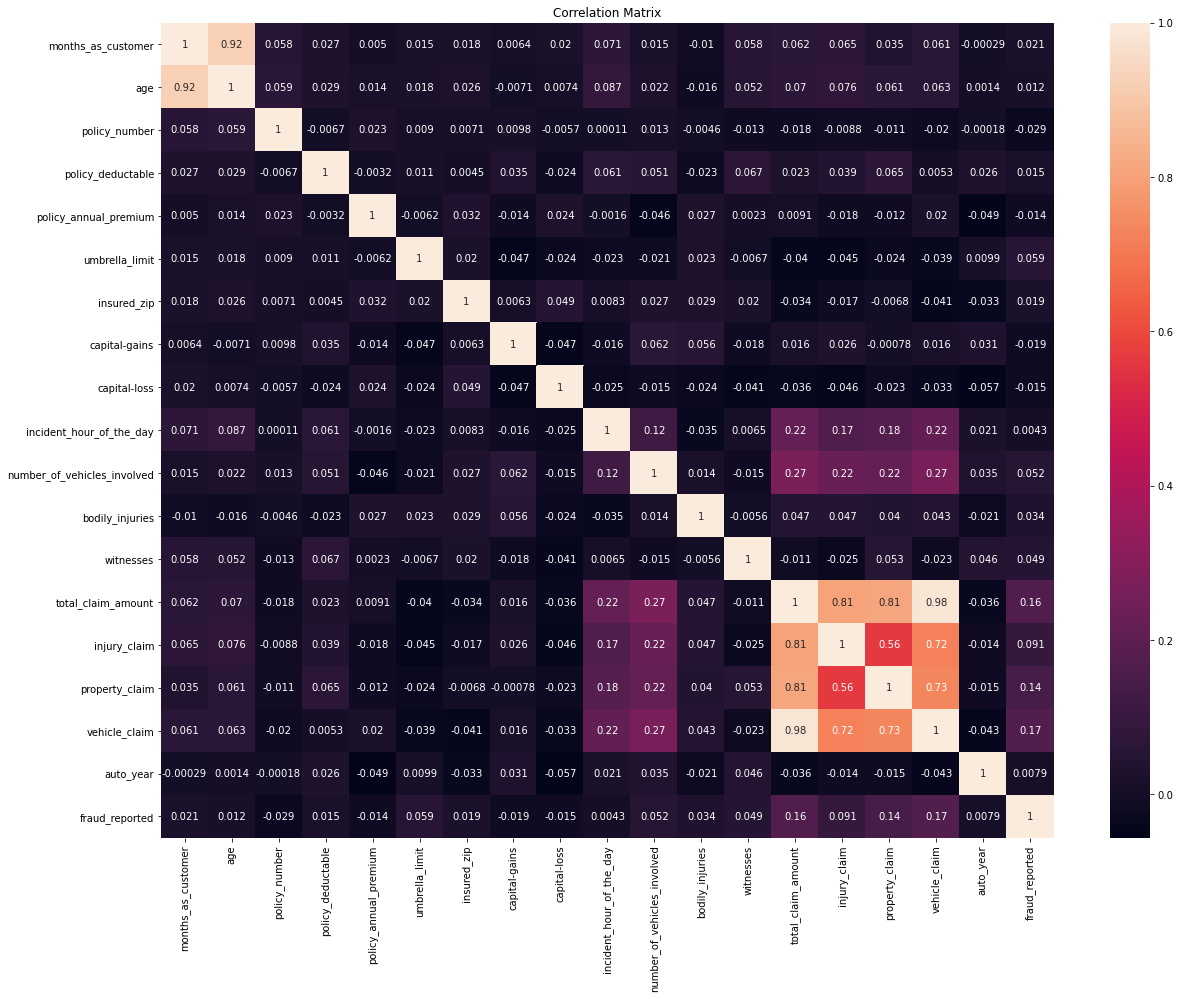
**NOW CHECKING THE CORRELATION OF DATA:-**

plt.figure(figsize = [20, 15])

sns.heatmap(dx.corr(), annot = True)

plt.title('Correlation Matrix')

plt.show()



**Observations:-**

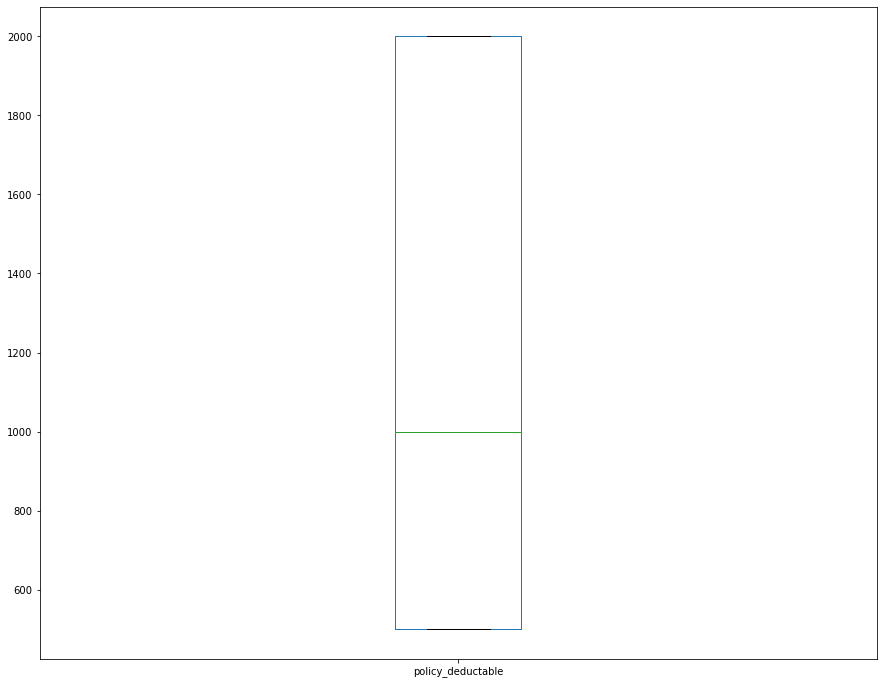
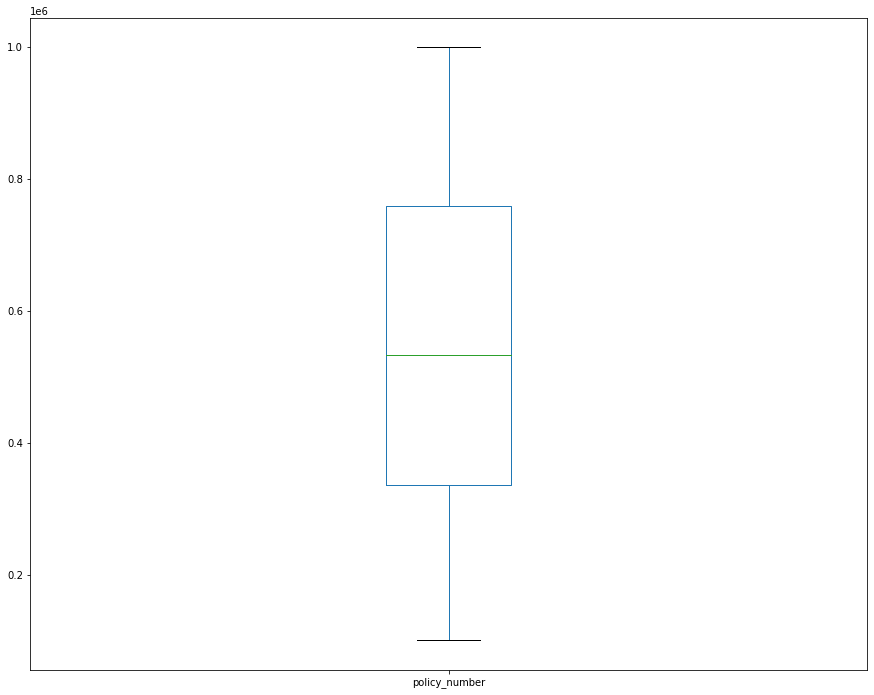
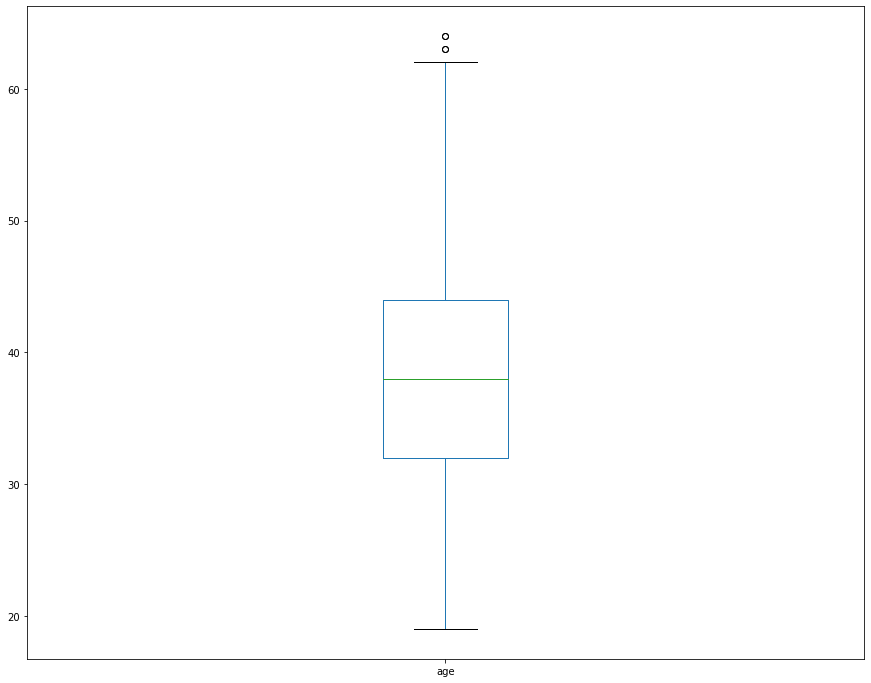
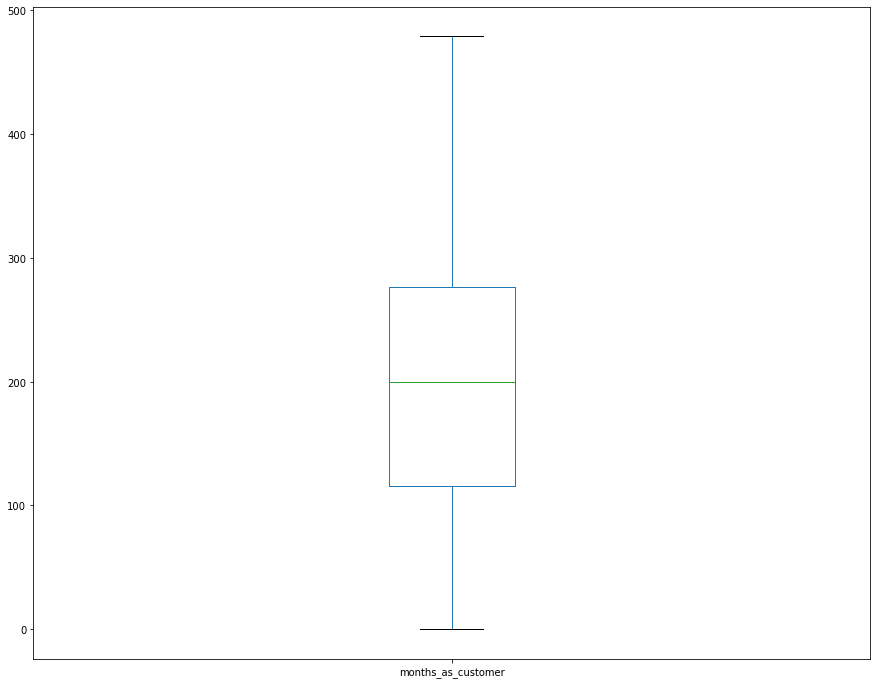
This will be used for checking for any multicollinearity exists in this Data or Not. So there is No of No issue with this Data set.

**NOW CHECKING OUTLIERS FOR THE DATA:-**

for i in cont\_cols:

plt.figure(figsize = (15, 12))

dx[i].plot.box()



**Observations:-**

So there are Outliers are present in the data. But it is not an issue.

**NOW CHECKING SKEWNESS:-**

dx[cont\_cols].skew()

months\_as\_customer 0.362177

age 0.478988

policy\_number 0.038991

policy\_deductable 0.477887

umbrella\_limit 1.806712

insured\_zip 0.816554

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\_annual\_premium 0.004402

dtype: float64

**Observations:-**

Here skweness is not a big issue.

**NOW CHANGING REPLACING CATEOGORICAL VALUES:-**

Now importing the LableEncoding from Sklearn library.

**"from sklearn.preprocessing import LabelEncoder**

**label\_encoder = LabelEncoder()**

**for col in cate\_cols:**

**dx[col] = label\_encoder.fit\_transform(dx[col])"**

**Observations:-**

Using LabelEncoder to convert object data types into float or int, so we can use these values in models.

\*\*\*\*END OF EDA PROCESS\*\*\*\*.

**APPLYTING MACHINE LEARNING MODEL**

**NOW SPLITING THE DATA INTO X & Y:-**

x = dx.drop('fraud\_reported', axis = 1)

y = dx['fraud\_reported']

**NOW PERMFORMING BEST MODELS FOR DATA TO PREDICT WELL:-**

# a)Best Random State:-

maxAccu = 0

maxRS = 0

for i in range (1, 500):

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.30, random\_state = i)

LR = LogisticRegression()

LR.fit(x\_train, y\_train)

pred = LR.predict(x\_test)

acc = accuracy\_score(y\_test, pred)

if acc>maxAccu:

maxAccu = acc

maxRS = i

print ('Max Accuracy obtained is', maxAccu, 'on Random State', maxRS)

**Note:**

after splitting the Data in X & Y applied the test\_size of 0.30 with random state in i which Ranges from (1-500).

Observations:

Max Accuracy obtained is 0.81 on Random State 16

# **CREATING TRAIN SPLIT TEST:-**

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.30, random\_state = 373)

LR = LogisticRegression()

DTC = DecisionTreeClassifier()

svc = SVC()

rf = RandomForestClassifier()

models = [LR, DTC, svc, rf]

for i in models:

print (i)

i.fit(x\_train, y\_train)

pred = i.predict(x\_test)

print (accuracy\_score(y\_test, pred))

print (confusion\_matrix(y\_test, pred))

print (classification\_report(y\_test, pred))

**Note:-**

As it's an classifier problem imported Logistic Regression Assigned with LR variable. In that Assigned other DTC Variable with Decision Tree Classifier, SVC () as support vector classifier, Random Forest Classifier with variable name rf. respectively. And then fit these in y\_train to predicts the Accuracy scores ,Confusion\_matrix, Classification\_report .

LogisticRegression()

0.81

[[241 2]

[ 55 2]]

precision recall f1-score support

0 0.81 0.99 0.89 243

1 0.50 0.04 0.07 57

accuracy 0.81 300

macro avg 0.66 0.51 0.48 300

weighted avg 0.75 0.81 0.74 300

DecisionTreeClassifier()

0.83

[[214 29]

[ 22 35]]

precision recall f1-score support

0 0.91 0.88 0.89 243

1 0.55 0.61 0.58 57

accuracy 0.83 300

macro avg 0.73 0.75 0.74 300

weighted avg 0.84 0.83 0.83 300

SVC()

0.81

[[243 0]

[ 57 0]]

precision recall f1-score support

0 0.81 1.00 0.90 243

1 0.00 0.00 0.00 57

accuracy 0.81 300

macro avg 0.41 0.50 0.45 300

weighted avg 0.66 0.81 0.72 300

RandomForestClassifier()

0.8233333333333334

[[222 21]

[ 32 25]]

precision recall f1-score support

0 0.87 0.91 0.89 243

1 0.54 0.44 0.49 57

accuracy 0.82 300

macro avg 0.71 0.68 0.69 300

weighted avg 0.81 0.82 0.82 300

**Observations:-**

Above the Predictions of Accuracy Score, Random Forest, Support vector machine, Logistic regression, Decision Tree Classifier are present in Data.

**CROSS VALIDATION REPORT:-**

**for i in models:**

**cvs = cross\_val\_score(i, x, y, cv = 5)**

**print ('Cross Validation Score for ',i, ' model is :', cvs.mean())**

**print (' ')**

Cross Validation Score for LogisticRegression() model is : 0.751

Cross Validation Score for DecisionTreeClassifier() model is : 0.784

Cross Validation Score for SVC() model is : 0.7529999999999999

Cross Validation Score for RandomForestClassifier() model is : 0.771

**Observations:-**

This is used to check the individual score for Logistic Regression, Decision Tree Classifier, Support vector Machine, Random Forest Classifier respectively. Here i got Cross Validation Score LogisticRegression() is 0.751 i.e 75.1%

# **HYPERTUNING:-**

from sklearn.model\_selection import GridSearchCV

param\_grid = { 'criterion':['gini','entropy'],'max\_depth': np.arange(3, 15)}

gdtc = GridSearchCV(estimator= DTC, param\_grid=param\_grid, cv= 3)

gdtc.fit(x\_train, y\_train)

GridSearchCV(cv=3, estimator=DecisionTreeClassifier(),

param\_grid={'criterion': ['gini', 'entropy'],

'max\_depth': array([ 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14])})

gdtc.best\_params

{'criterion': 'entropy', 'max\_depth': 3}

gdtc1 = DecisionTreeClassifier(random\_state = 373, max\_depth = 3, criterion = 'entropy')

gdtc1.fit(x\_train, y\_train)

predgdtc1 = gdtc1.predict(x\_test)

print (accuracy\_score(y\_test, predgdtc1))

print (confusion\_matrix(y\_test, predgdtc1))

print (classification\_report(y\_test, predgdtc1))

0.8366666666666667

[[202 41]

[ 8 49]]

precision recall f1-score support

0 0.96 0.83 0.89 243

1 0.54 0.86 0.67 57

accuracy 0.84 300

macro avg 0.75 0.85 0.78 300

weighted avg 0.88 0.84 0.85 300

cvs = cross\_val\_score(gdtc1, x, y, cv = 5)

print ('Cross Validation Score for ',gdtc1, ' model is :', cvs.mean())

Cross Validation Score for DecisionTreeClassifier(criterion='entropy', max\_depth=3, random\_state=373) model is : 0.865

**Observations:-**

After hypertuning with gridsearchcv our decision tree model's accuracy and cross val scores increased from 0.81 to 0.84 and 0.79 to 0.86 respectively.

**SAVING A MODEL :-**

import pickle

filename = 'Insurancegdtc.pkl'

pickle.dump(gdtc1, open(filename, 'wb'))

**Observations:-**

The Library Pickle is used for saving the model which was predicted.

**CONCLUSION:-**

**This is final part of the Machine Learning Model.**

loaded\_model = pickle.load(open('Insurancegdtc.pkl', 'rb'))

result = loaded\_model.score(x\_test, y\_test)

print (result)

0.8366666666666667.

**Here after saving in to pickle got the prediction score of 83.6%**

conclusion = pd.DataFrame([loaded\_model.predict(x\_test)[:], predgdtc1[:]], index = ['Predicted', 'Original'])

conclusion

| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **...** | **290** | **291** | **292** | **293** | **294** | **295** | **296** | **297** | **298** | **299** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Predicted** | **0** | **1** | **1** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **...** | **1** | **1** | **0** | **0** | **0** | **0** | **1** | **0** | **0** | **0** |
| **Original** | **0** | **1** | **1** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **...** | **1** | **1** | **0** | **0** | **0** | **0** | **1** | **0** | **0** | **0** |

**2 rows × 300 columns**

So, this is the final conclusion of the Classification Data of "**INSURANCE FRAUD CLAIM**".

\*\*\*\*END OF THE PROJECT\*\*\*\*