# Insurance Claim – Fraud Detection ML Project Explained

Hello Friends, My Name is Sanjay Panchal. I have been working on Machine Learning projects with Python since few months. I have worked on more than 20 projects. I got lots of knowledge about Machine Learning from my projects. Today I am going to talk about a Machine Learning project named Insurance Claim Fraud Detection. This projects is one of my favourite projects which I got great result. In this project we will see that what is the problem statement and how to solve it with Machine Learning using Python. I will tell you about this step by step and how to increase the final score. So, let's start…

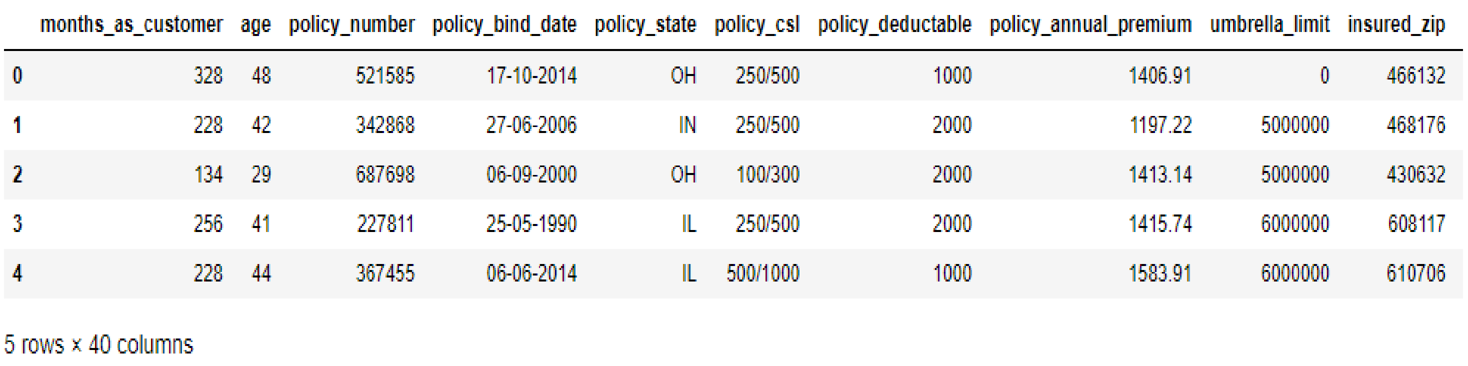
**Let's first see the problem statement and some details about dataset..**

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, we have 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, 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.

First we need to import the dataset. As we can see that we have successfully imported the dataset below:

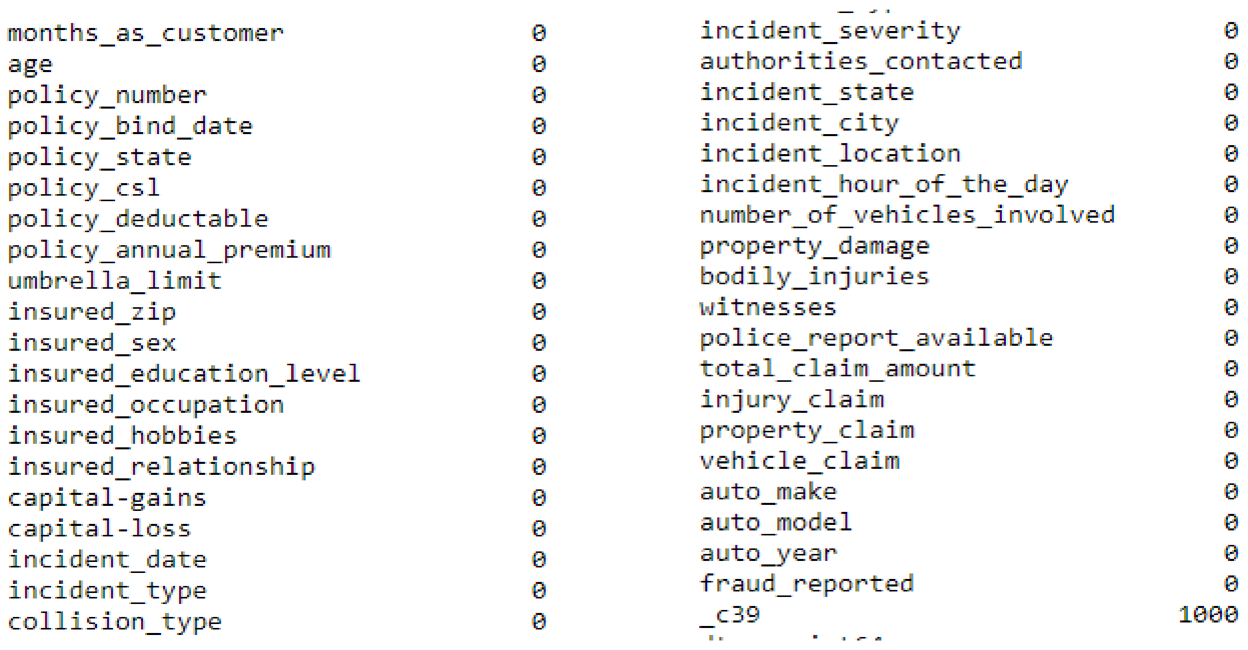


| | months_as_customer | age | policy_number | policy_bind_date | policy_state | policy_csl | policy_deductable | policy_annual_premium | umbrella_limit | insured_zip |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 328 | 48 | 521585 | 17-10-2014 | OH | 250/500 | 1000 | 1406.91 | 0 | 466132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

5 rows × 40 columns

This dataset contains 1000 Rows and 40 columns.

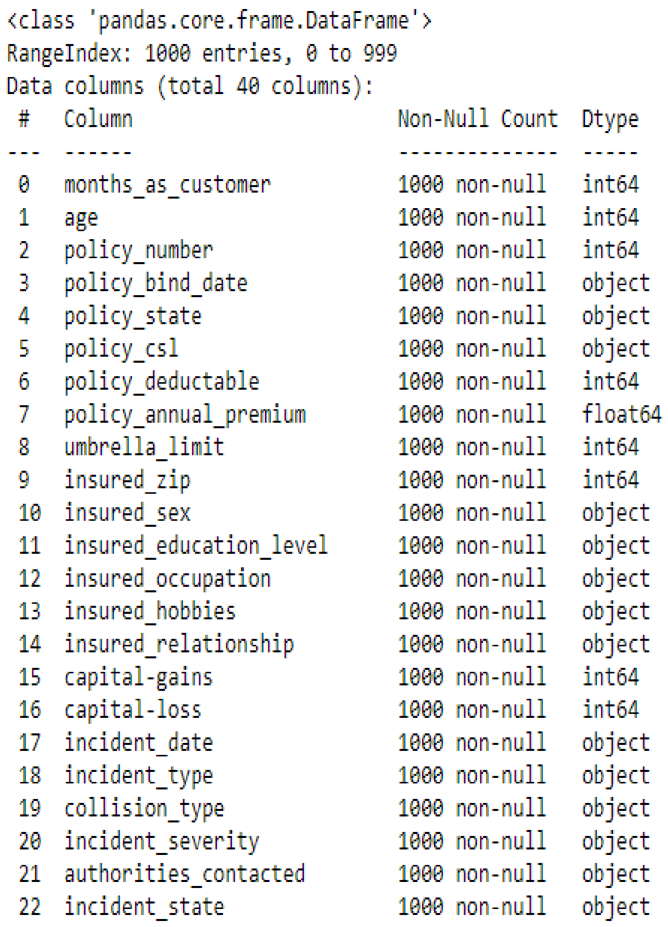
**Now let's check the all columns name and null values if any present in the dataset.**



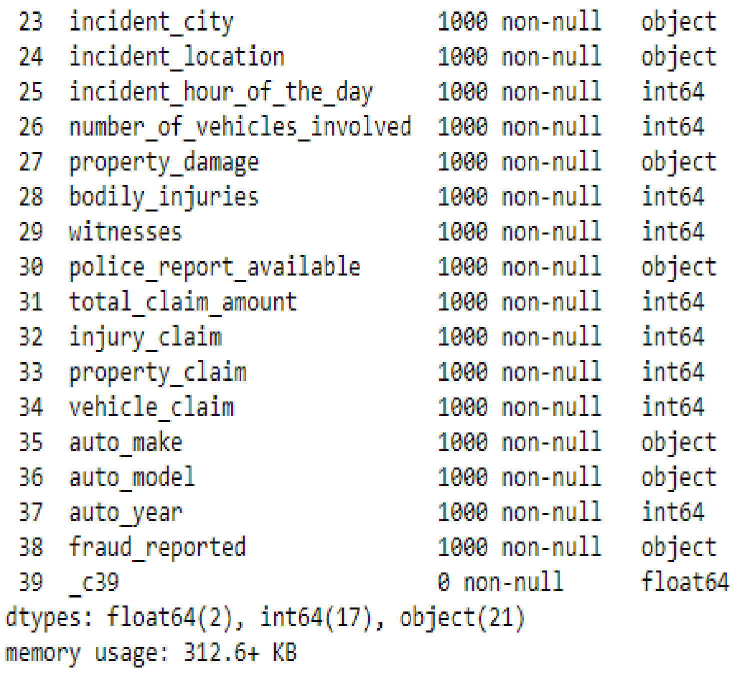
```
months_as_customer             0      incident_severity            0
age                            0      authorities_contacted        0
policy_number                  0      incident_state               0
policy_bind_date               0      incident_city                0
policy_state        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

Now we can see the all columns name and its null values. Now it’s clear that only one column has missed 1000 Rows data means 100 %. So we don’t need this columns and have to drop it because it does not have any data.

**Now let’s check the some necessary information about dataset :-**



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 40 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   months_as_customer           1000 non-null   int64
 1   age                          1000 non-null   int64
 2   policy_number                1000 non-null   int64
 3   policy_bind_date             1000 non-null   object
 4   policy_state                 1000 non-null   object
 5   policy_csl                   1000 non-null   object
 6   policy_deductable            1000 non-null   int64
 7   policy_annual_premium        1000 non-null   float64
 8   umbrella_limit               1000 non-null   int64
 9   insured_zip                  1000 non-null   int64
 10  insured_sex                  1000 non-null   object
 11  insured_education_level      1000 non-null   object
 12  insured_occupation           1000 non-null   object
 13  insured_hobbies              1000 non-null   object
 14  insured_relationship         1000 non-null   object
 15  capital-gains                1000 non-null   int64
 16  capital-loss                 1000 non-null   int64
 17  incident_date                1000 non-null   object
 18  incident_type                1000 non-null   object
 19  collision_type               1000 non-null   object
 20  incident_severity            1000 non-null   object
 21  authorities_contacted        1000 non-null   object
 22  incident_state               1000 non-null   object
```



```
 23  incident_city                1000 non-null   object
 24  incident_location            1000 non-null   object
 25  incident_hour_of_the_day     1000 non-null   int64
 26  number_of_vehicles_involved  1000 non-null   int64
 27  property_damage              1000 non-null   object
 28  bodily_injuries              1000 non-null   int64
 29  witnesses                    1000 non-null   int64
 30  police_report_available      1000 non-null   object
 31  total_claim_amount           1000 non-null   int64
 32  injury_claim                 1000 non-null   int64
 33  property_claim               1000 non-null   int64
 34  vehicle_claim                1000 non-null   int64
 35  auto_make                    1000 non-null   object
 36  auto_model                   1000 non-null   object
 37  auto_year                    1000 non-null   int64
 38  fraud_reported               1000 non-null   object
 39  _c39                         0 non-null      float64
dtypes: float64(2), int64(17), object(21)
memory usage: 312.6+ KB
```

Now we can see that we have 2 float64, 17 int64 and 21 objects columns. And our target is to predict the columns name fraud_reported.

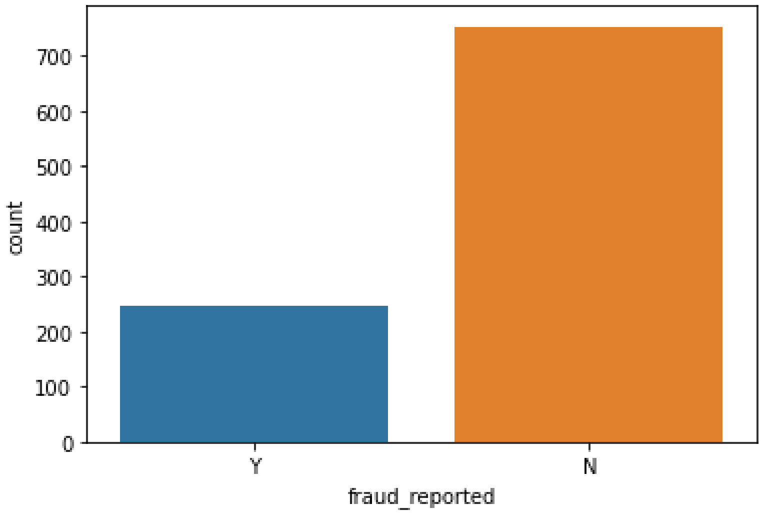
**And after checking the values counts of each columns, I found that some columns is not useful for model building and need to drop it. Columns name to be drop : -**

Policy_number = It is the unique of every customer.
_C39 = It has all null values.
Policy bind date = Not required for prediction.
Incident_date = Not required for prediction.
Insured_zip = Almost unique for every customer.
Incident_location = It has each unique value.

**Now it time to visualize the target columns.**

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

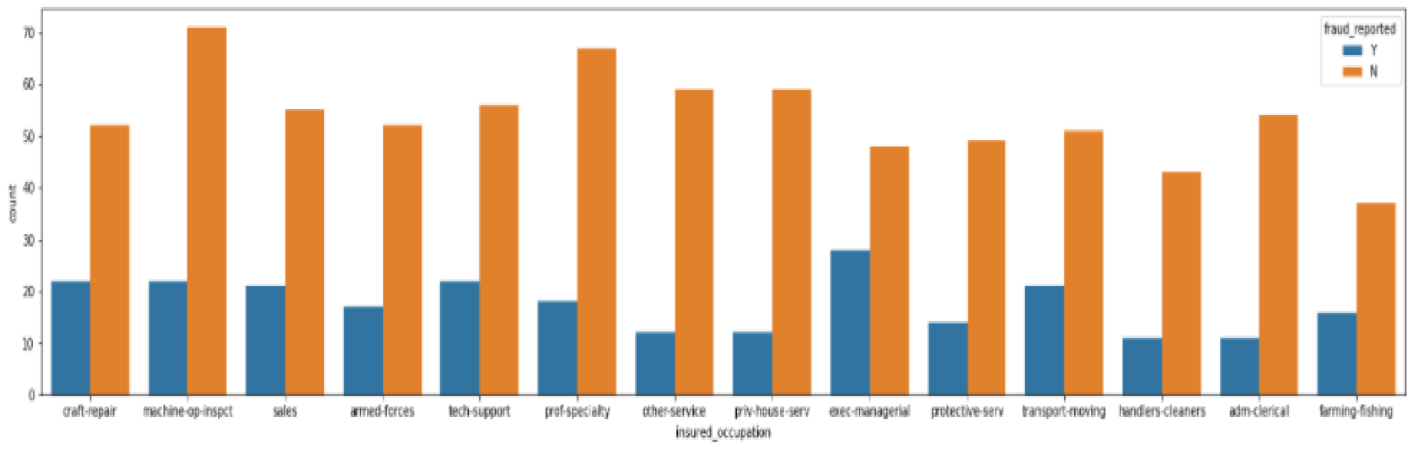
```
<AxesSubplot:xlabel='fraud_reported', ylabel='count'>
```



Our target columns is fraud_reported and after visualize I found only 2 outputs means there is classification problem and column have Y & N means Yes Or No. Y has 247 entries and N has 753 entries means there is class imbalance issue exists and need to balance the classes for model building. So, let’s visualize the whole dataset and then we will balance the classes.

**Now let’ s check how insured_occupation affects the fraud_reported:-**

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

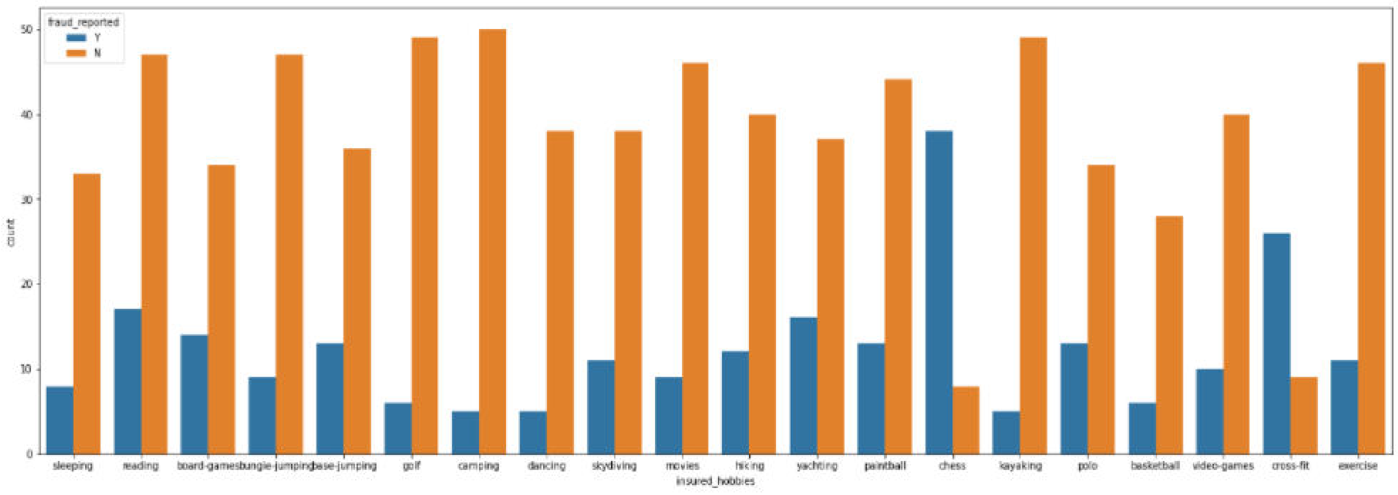


It seems that most of exec-managerials are doing fraud than others. Otherwise all are doing fraud and can't say anything.

Let's observe another things...

**Now let' s check how insured_hobbies affects the fraud_reported:-**

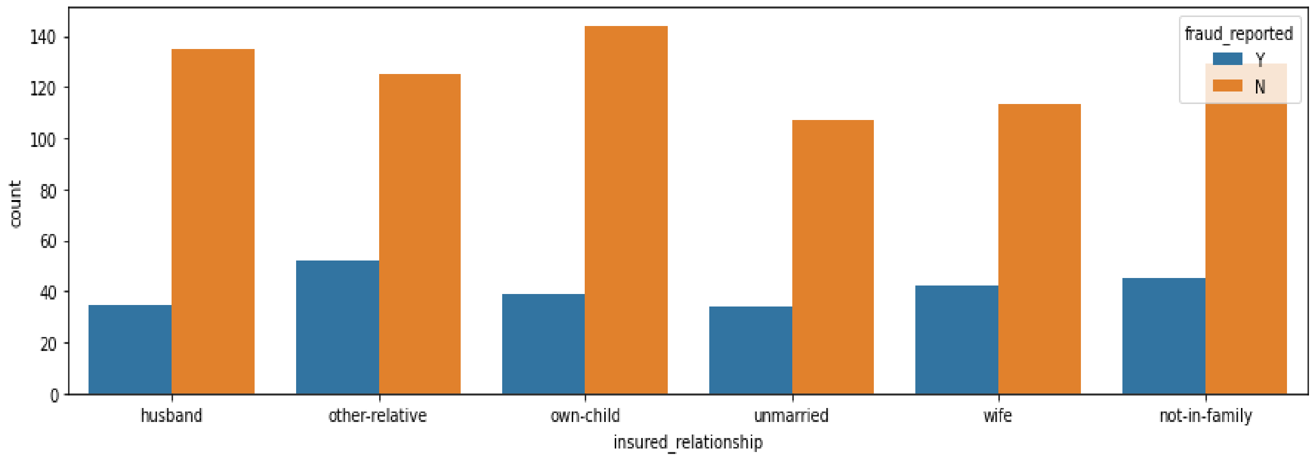
```
<AxesSubplot:xlabel='insured_hobbies', ylabel='count'>
```



Most of the chess player and cross-fit are doing fraud...

**Now let' s check how insured_relationship affects the fraud_reported:-**

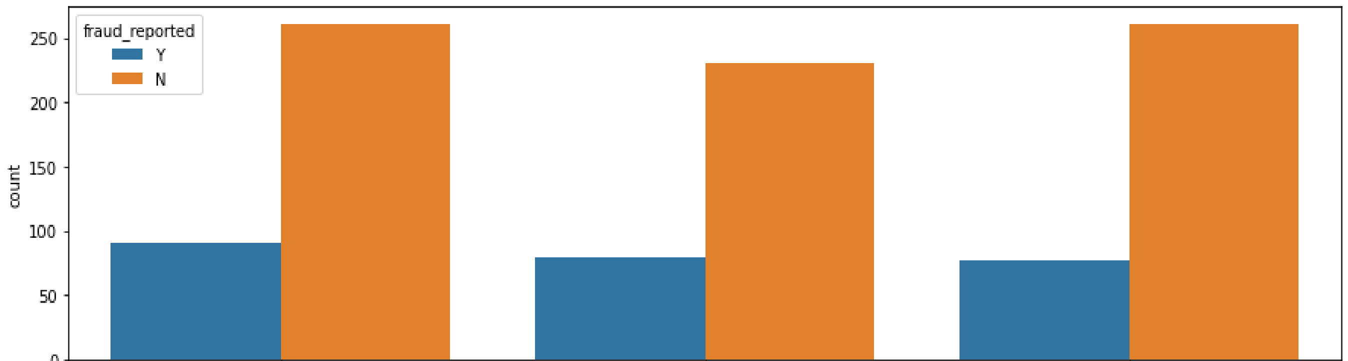
```
<AxesSubplot:xlabel='insured_relationship', ylabel='count'>
```



In relationship all are almost equal in fraud report.

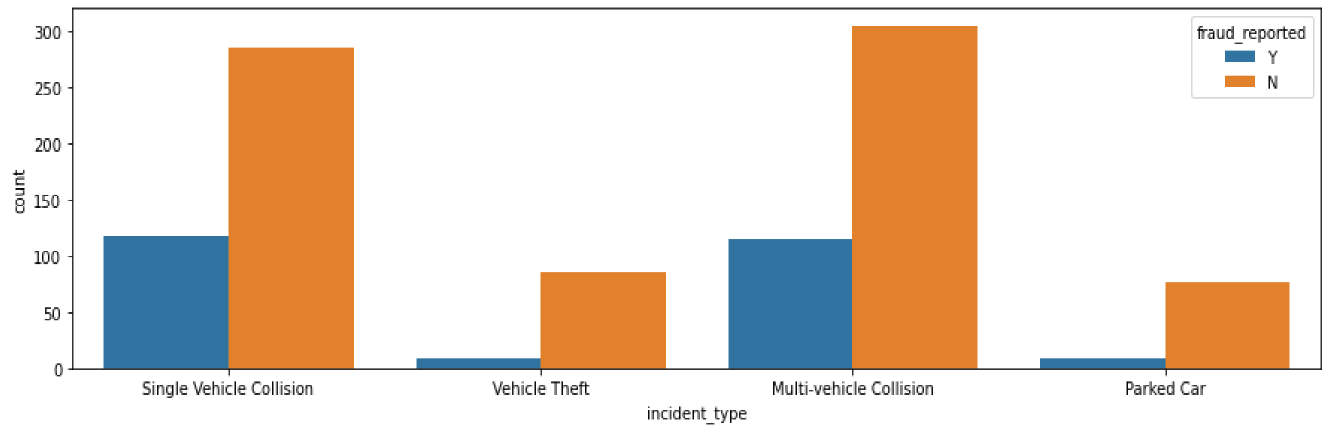
**Now let' s check how policy_state affects the fraud_reported:-**

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



**Now let' s check how incident_type affects the fraud_reported:-**

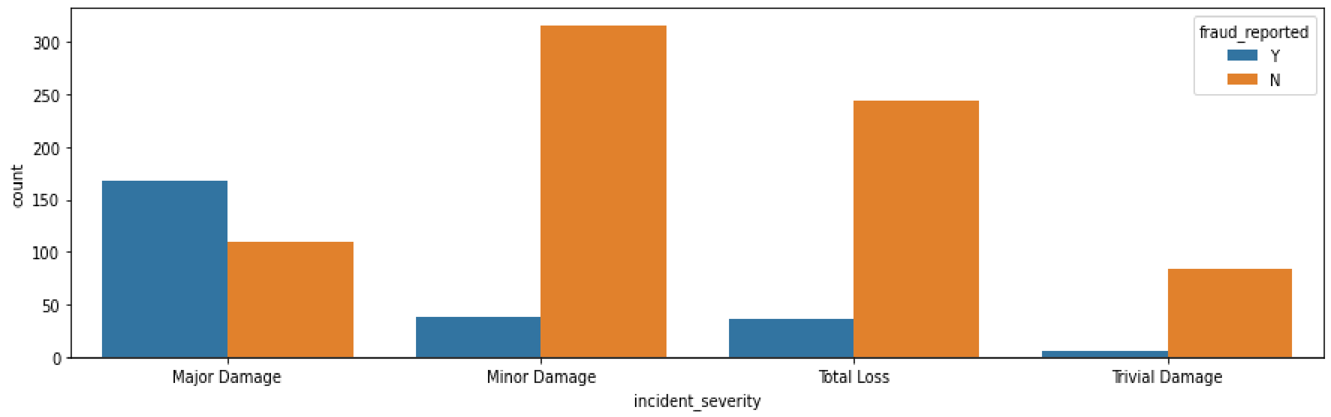
```
<AxesSubplot:xlabel='incident_type', ylabel='count'>
```



Most of the Single Vehicle Collision and Multi-Vehicle Collision detect in fraud report. Now let's analyse the collision type in fraud report...

**Now let' s check how incident_severity affects the fraud_reported:-**

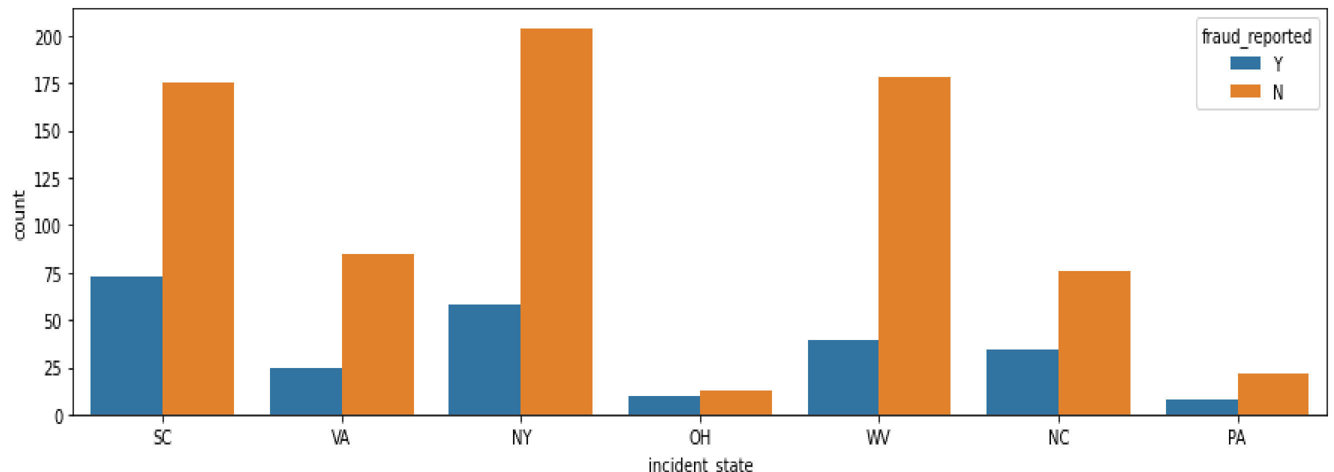
```
<AxesSubplot:xlabel='incident_severity', ylabel='count'>
```



Most of the Major Damage issues detect in fraud report..

**Now let' s check how incident_state affects the fraud_reported:-**

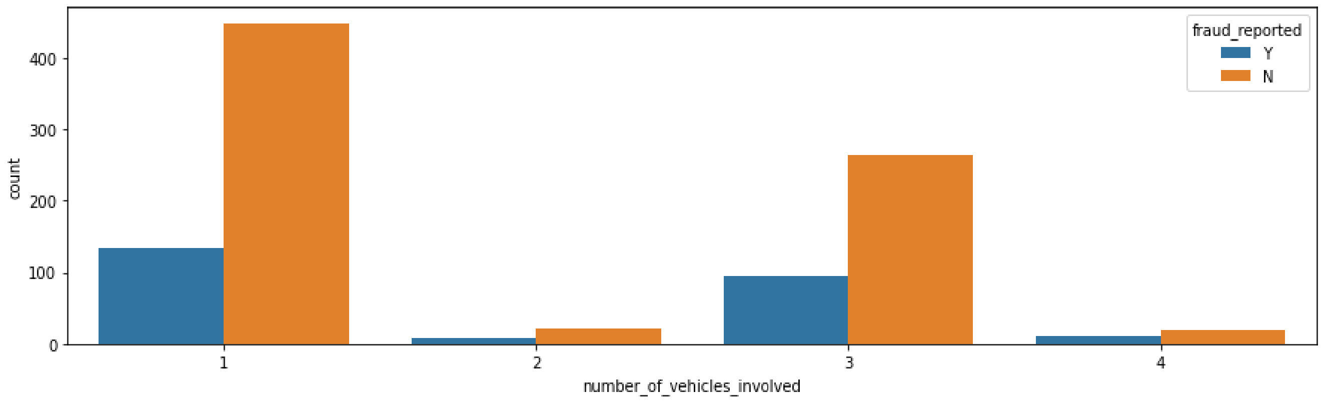
```
<AxesSubplot:xlabel='incident_state', ylabel='count'>
```



Most of the fraud case are from SC & NY states.

**Now let' s check how number_of_vehicle affects the fraud_reported:-**

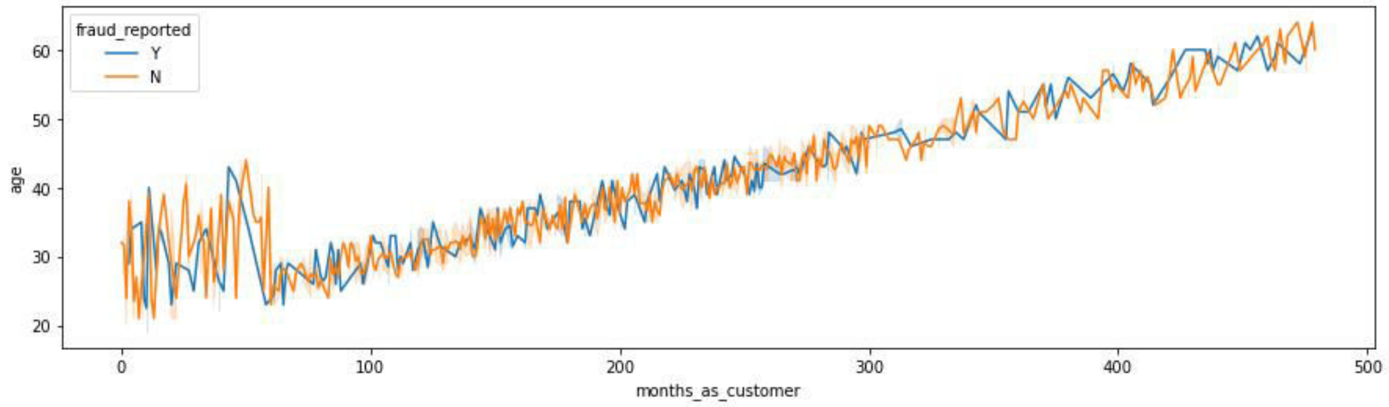
```
<AxesSubplot:xlabel='number_of_vehicles_involved', ylabel='count'>
```



Most of 1 and 3 vehicle reported in fraud cases.

**Now let' s check how age and months_of_customers affects the fraud_reported:-**

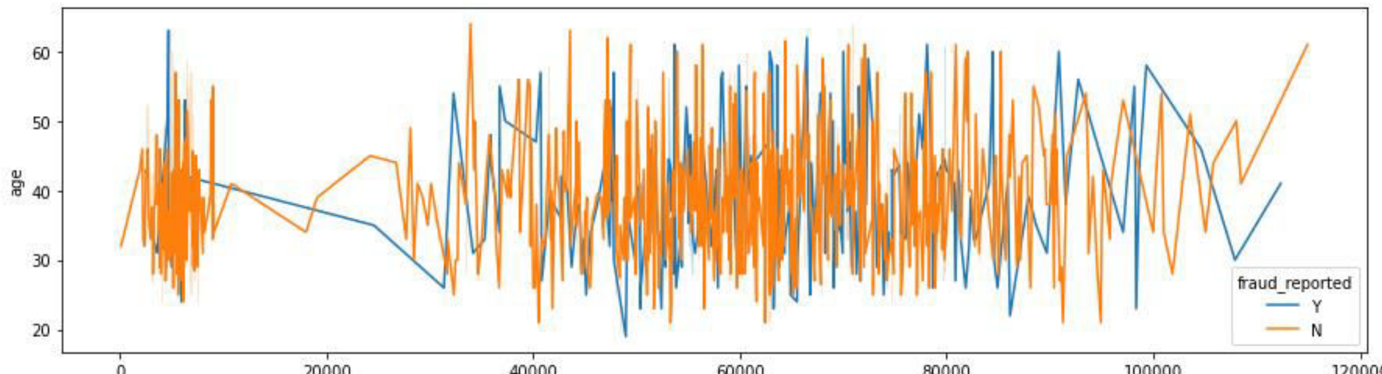
```
<AxesSubplot:xlabel='months_as_customer', ylabel='age'>
```



Most of higher age customers and new customers are detected in fraud report. Low age customer and old customers is rarely found in fraud report.

**Now let' s check how total_claim_amount affects the fraud_reported:-**

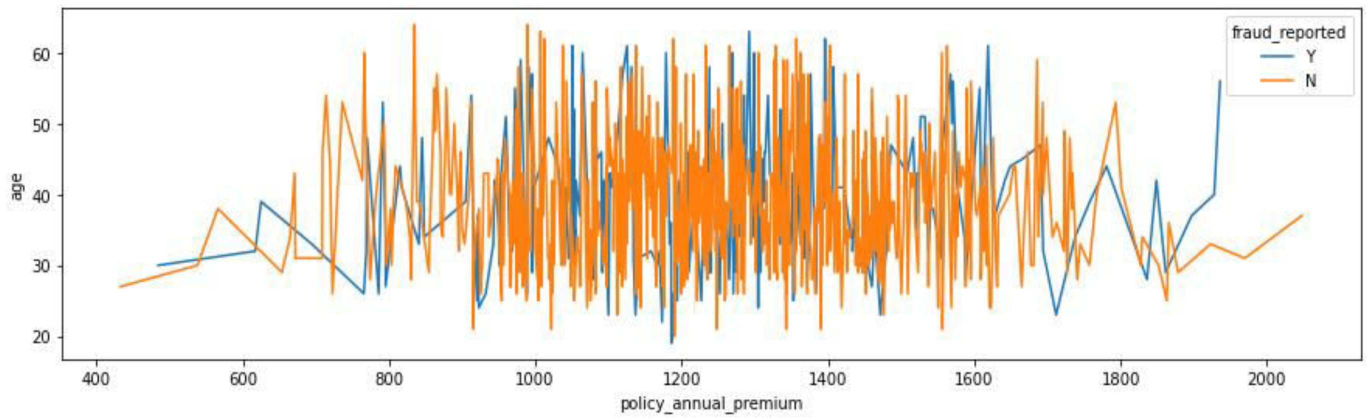
```
<AxesSubplot:xlabel='total_claim_amount', ylabel='age'>
```



Most of the customers who claimed the amount more than 40000. they generely detected in fraud report. And in most cases it happened.

**Now let' s check how policy annual premium affects the fraud_reported:-**

```
<AxesSubplot:xlabel='policy_annual_premium', ylabel='age'>
```



The customer who has policy annual premium between 1000 to 1600 mostly detected in fraud report. Mostly customer suffered from their annual premium. So maybe that's why they do it..

**And after analysing the dataset it is observed that total_claim_amount is the some of injury_claim, property_claim & vehicle_claim. So we don't need this column and have to drop it.**

**After analysing the correlation matrix, most of the columns is not highly related to another column but in some cases there is 4 to 5 columns which is significantly correlated to another columns.**

Age and months of customer is 92% correlated to each other.
Vehicle claim and injury claim is 72% correlated to each other.
Vehicle claim and property claim is 73% correlated to each other.
Property claim and injury claim is 56% correlated to each other.

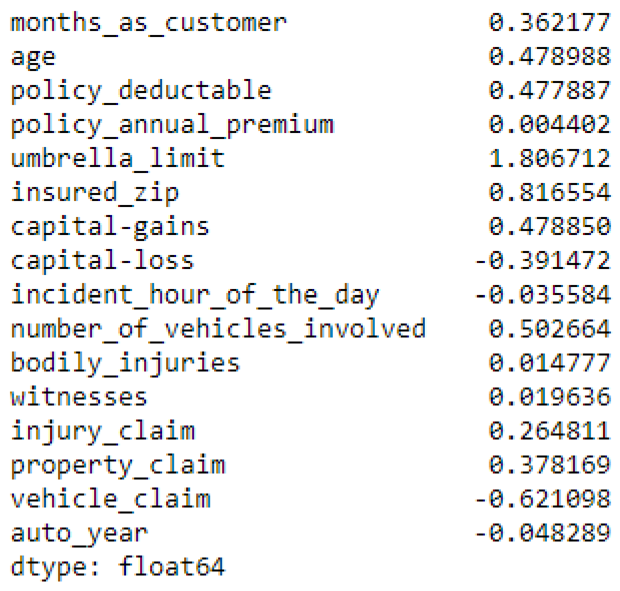
And 5-8 columns is nearly 15-30% correlated to each other. I didn't show the correlation matrix here because that matrix graph is high in size so I explained here what is observed in that.

**Skewness :-**

Now it's time to check the skewness, Skewness is the major part of every machine learning model. Without handle the skewness you cannot make a perfect predicted machine learning model.

So our first priority is to deal with skewness. If anyone know how to handle with it then they can make perfect model. And keep one thing in mind that Skewness works on numerical columns only and you will find only numerial column while checking the skewness.

So let’s check the skewness of every column  and consider a skewness range between +0.5 to -0.5. If any features found between this then it ok if not found then we need to adjust between this range.

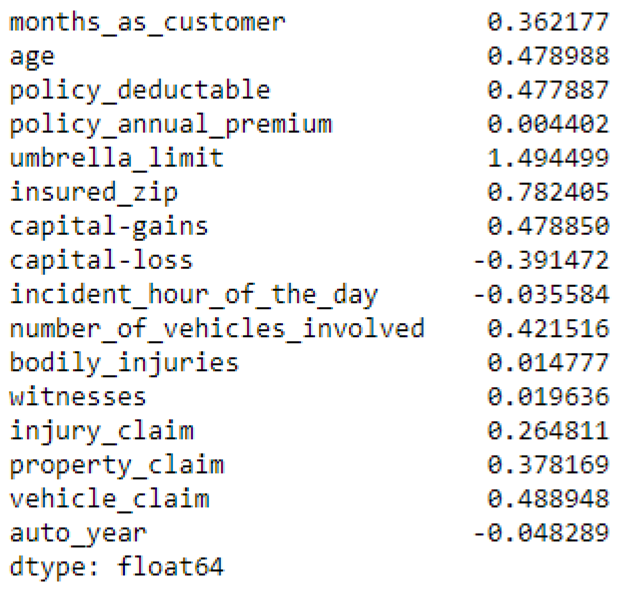


```
months_as_customer             0.362177
age                            0.478988
policy_deductable              0.477887
policy_annual_premium          0.004402
umbrella_limit                 1.806712
insured_zip                    0.816554
capital-gains                  0.478850
capital-loss                  -0.391472
incident_hour_of_the_day      -0.035584
number_of_vehicles_involved    0.502664
bodily_injuries                0.014777
witnesses                      0.019636
injury_claim                   0.264811
property_claim                 0.378169
vehicle_claim                 -0.621098
auto_year                     -0.048289
dtype: float64
```

Now I can see that most of the columns are between skewness range. But there are 4 columns out of range which is umbrella limit, insured zip, number of vehicles involved and vehicle claim.

So let’s the the skewness technique to adjust these 4 columns between skewness range. There are lots of technique, you can choose according to you but you have to perfect in that techniques because some technique causes lose of data and data is very important. So, choose such a technique in which you don’t lose the data.

I applied some technique, So let’s check the skewness again…



```
months_as_customer             0.362177
age                            0.478988
policy_deductable              0.477887
policy_annual_premium          0.004402
umbrella_limit                 1.494499
insured_zip                    0.782405
capital-gains                  0.478850
capital-loss                  -0.391472
incident_hour_of_the_day      -0.035584
number_of_vehicles_involved    0.421516
bodily_injuries                0.014777
witnesses                      0.019636
injury_claim                   0.264811
property_claim                 0.378169
vehicle_claim                  0.488948
auto_year                     -0.048289
dtype: float64
```

The skewness of all columns looks good but ther is one column named umbrella limit has high skewness. But if I will treat it then there is 100% chance to lose the 100% data because I tried. So, Now let's drop this column for better result. This is hit and trial method. You can keep according to you but according to me it is not good for best model.

As already discussed that our target column has only Y and N. and in classification model you need to make you target column in numerical form. So, change the value of Y to 1 and N to 0.

**Let’s covert the object column to numerical form:-**

In the quick information we already discussed that some columns are in object form and any model does work on any object column, So we need to convert the object column to numerical form before proceeding to the model.

So there are many technique like Label encoding, One hot encoding, Dummy variable encoding. You can use according you but I generally use Dummy variable encoding techniques because this is very easy to covert and don’t need to import any library for this.

Before converting the object column to numerical form we had only 40 columns but after converting we have 144 columns in this dataset in which 1 is our target and 143 is the features.

**Let’s solve class imbalance issue with target column:-**

Now our dataset is ready for proceeding to the algorithms but we already discussed In the beginning that our target column has class imbalance issue. So, first we need to balance it. So, let’s use the SMOTE technique to balance it.

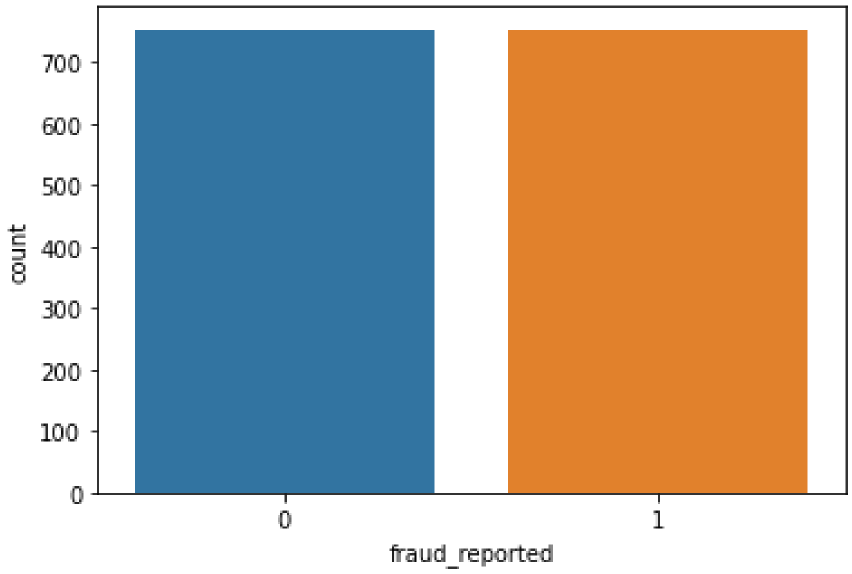
SMOTE technique is the famous technique for balancing the classes. Most of the ML engineer use this one. But before using the SMOTE technique we need to separate the features and target column.

SMOTE technique use the over sampling behind this and over sampling make balance lower class to higher class.

```
Shape of the features before over_sampling :  (1000, 143)
Shape of the features after over_sampling  :  (1506, 143)

Value Counts of Target Column --

: 1    753
  0    753
  Name: fraud_reported, dtype: int64
```

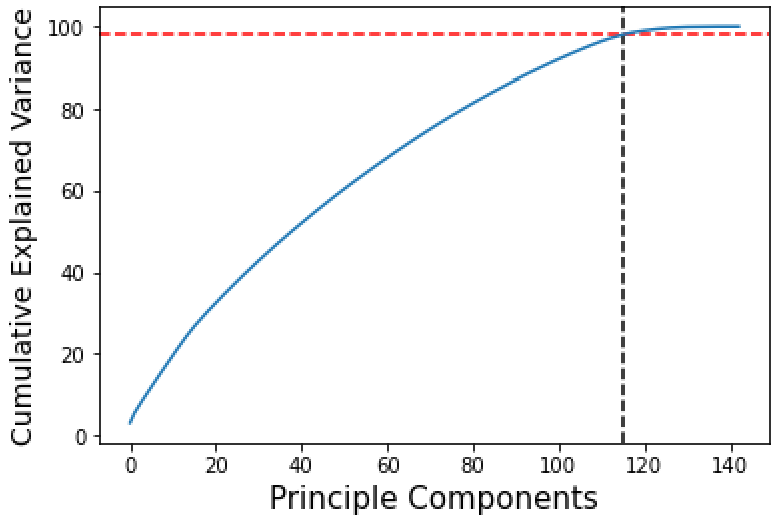


Before using the SMOTE technique we had only 1000 Rows but after balance the classed we have 1506 Rows because before proceeding to this technique, In target columns Y(1) had 753 data and N(0) had 247 data. And after this Y(1) and N(0) both have 753 data means now they have equal data.

**Let's apply the PCA technique:-**

As we now have 133 columns and due to this our model can take enough time for prediction. So, let's use the PCA technique to check how many columns we can give us 98% variance and generally ML engineer required only 95% but I always use 98% variance. That's why I will check 98% variance.

```
Number of components explained 98% variance :  115
```



Out of 143, only 115 features is giving the 98% variance. So, let's use the PCA technique for model building.

**Import the necessary libraries for model buildings:-**

Now we have done the all pre-processing steps and it's time to import the necessary libraries for model buildigs like:

Train test split = for test and train data
Accuracy score = to find the accuracy score
Cross validation score =  to find the corss validation scores
Plot roc curve  =  to plot the roc auc curve on test data

As we know that our target column has class imbalance issue so we will not use the classification report here.

**Now It's time to find the best random state:-**

Before apply any classification algorithm we have to find the best random state because It will help to find the best accuracy score and after find the best random state you can apply any classification algorithm.

And please make sure that test size range should be between 0.20 to 0.25. For very big dataset you can set the range upto 0.30.

**Select the classifier algorithms:-**

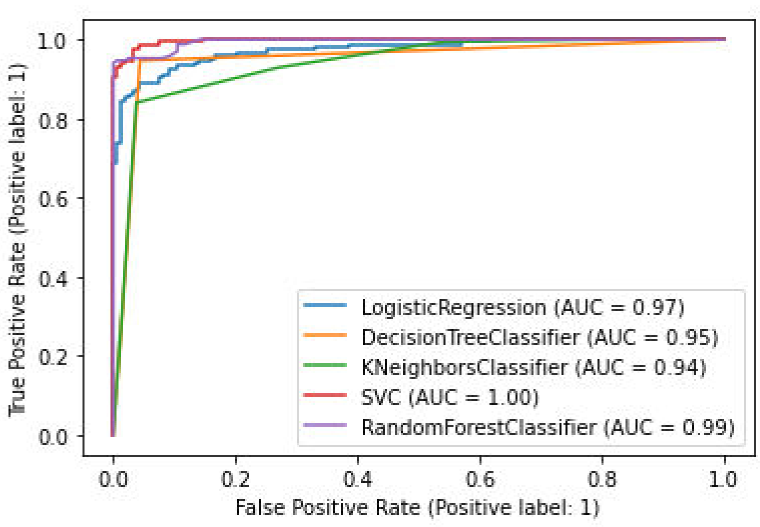
In this dataset I applied 5 classification algorithm like:

Logistic Regression
Decision Tree Classifier
K-Neighbors Classifier
Support Vector Machine
Random Forest Classifier

**Let's check applied classifier algorithm scores:-**

| | | |
|---|---|---|
| **Logistic Regression:-** | Best Accuracy Score | : 0.888 |
| | Cross Validation Score | : 0.834 |
| **Decision Tree Classifier:-** | Best Accuracy Score | : 0.822 |
| | Cross Validation Score | : 0.757 |
| **K-Neighbors Classifier :-** | Best Accuracy Score | : 0.662 |
| | Cross Validation Score | : 0.617 |
| **Support Vector Machine:-** | Best Accuracy Score | : 0.900 |
| | Cross Validation Score | : 0.829 |
| **Random Forest Classifier:-** | Best Accuracy Score | : 0.885 |
| | Cross Validation Score | : 0.831 |

**Let' plot ROC AUC Curve:-**



After analyzing the accuracy score, cross validation score and ROC AUC curve. Now it is clear that Support Vector Classifier is giving the best score. So, let's try to increase the accuracy score using Hyperparameter Tuning.

Hyper parameter tuning is the most popular technique to improve the accuracy score. Every ML engineer use it. If you want to increase the score then you must use it.

**Let’s check the final score after hyperparameter tuning:-**

After applying the hyper parameter tuning with Support Vector Classifier Our accuracy score is : 0.901

Now you can see that our score increased with 0.001%.

Most of the cases you score will increase after this technique but very rare chances that you score will not increase. And keep one thing in mind that If you score will not increase then please choose the high scored model only.

**Save the Final Model:**

Save your higher scored model.

**Conclusion:**

**Do EDA as much as you can.**

**Machine Learning required practice more and more.**

**Do correct Practice as much as you can.**

**Correct practice makes you perfect.**