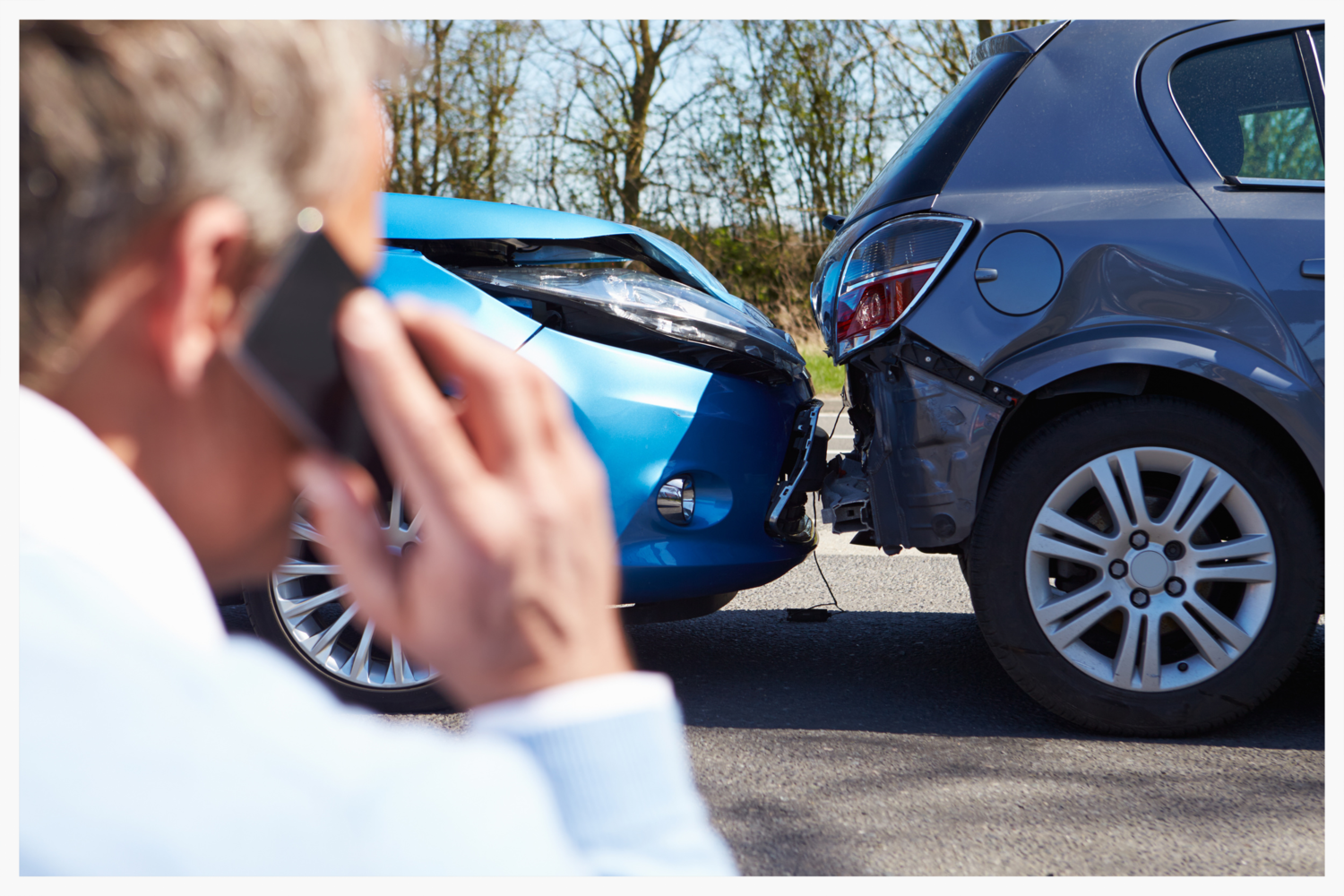
**Insurance Claims- Fraud Detection using Machine Learning**

Using Supervised Machine Learning Algorithms to identify if a particular insurance claim is fraudulent or not.

**PROBLEM DEFINITION:**



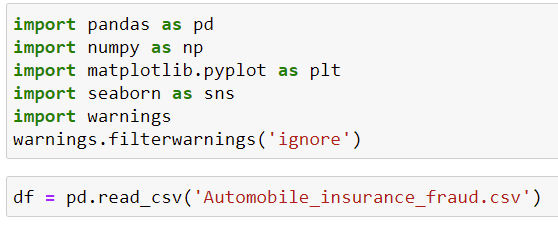
The Automobile Insurance Fraud is one of the main challenges for insurance companies. It is difficult to identify fraud claims. When most of the insurance cases are genuine there is equivalent chance of that case being fraudulent and that leads to greater financial losses.

Machine Learning is in a unique position to help the Auto Insurance industry with this problem.  We are provided a dataset which has the details of the insurance policy along with the customer details. We are required to predict if the particular insurance case is genuine or fraudulent when it receives an input.

**DATA ANALYSIS:**

**Overview of Data**

First, we need to import the necessary libraries and data.



Information of imported libraries:

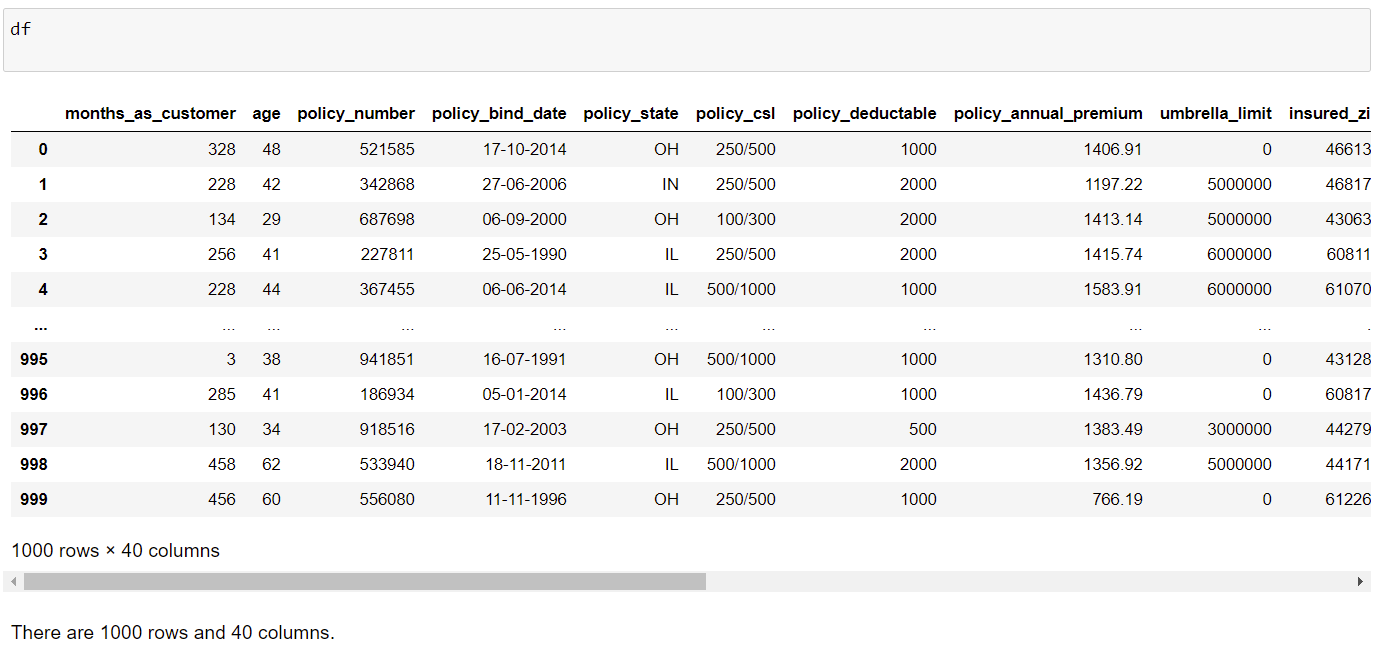
**pandas** to handle tabular data,

**numpy** to support large, multi-dimensional arrays and matrices, help to perform mathematical functions on those arrays.

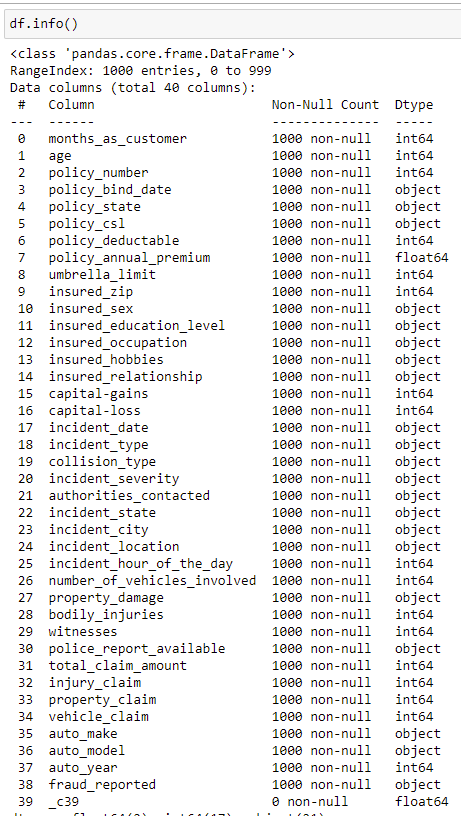
**Matplotlib** and **seaborn** for data visualizations.

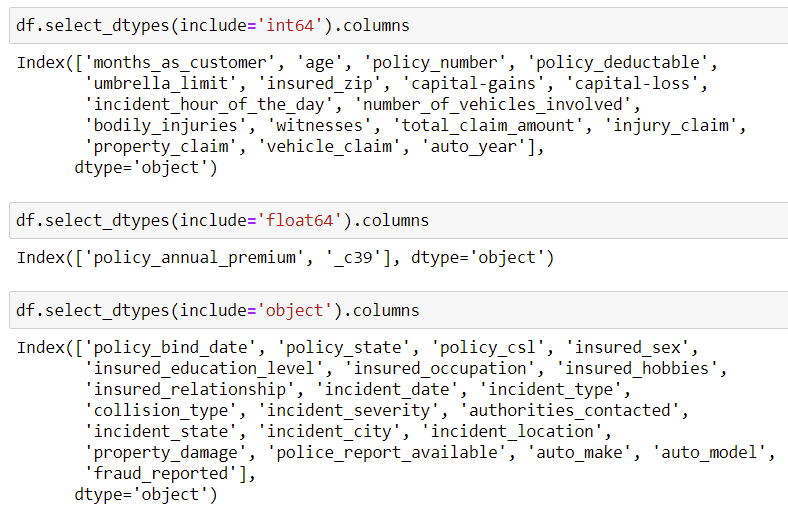
**warnings** help us not show unnecessary warning while we continue working on the project.

Now, we will look at our data.



There are 1000 rows and 40 columns.





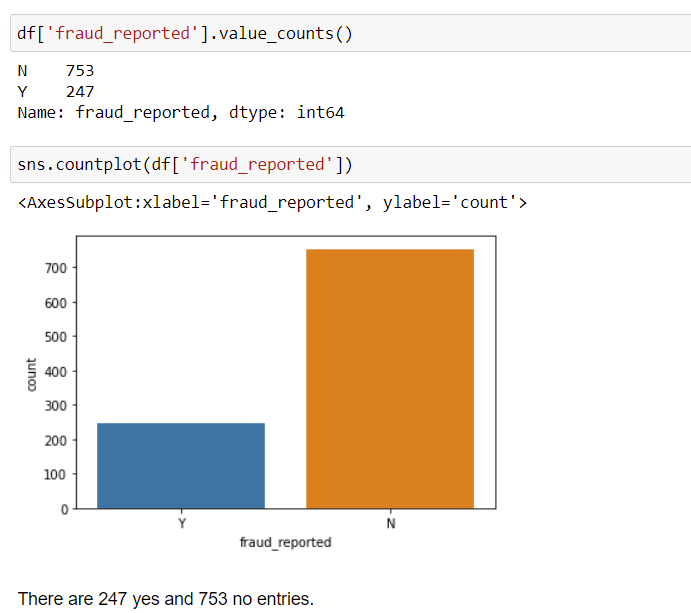
We have displayed integer, float and object datatypes for our convenience.

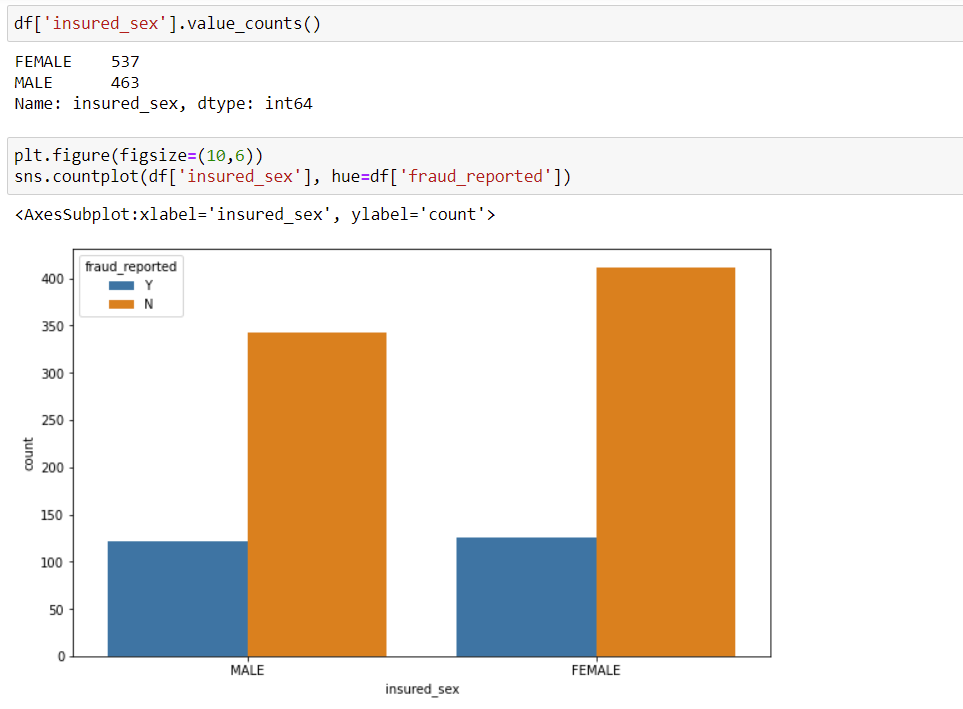
There are Null values in the dataset specified as '?'. Column \_c39 has all null values so we will need to drop that column.

‘fraud\_reported’ is the target column. Y and N are to classes in the column. This is **Bi-Classification** problem and we will consider this as a **Data Imbalance** problem. We will need to deal with data imbalance problem.

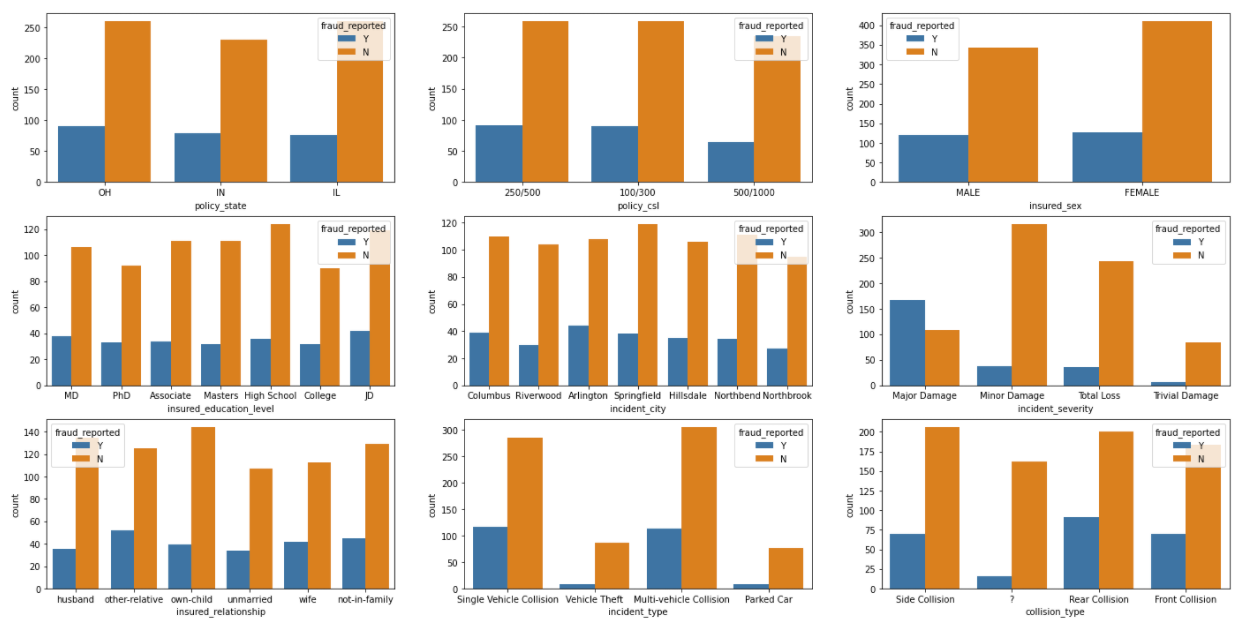
**EDA**

We will now visualize data to understand the data.

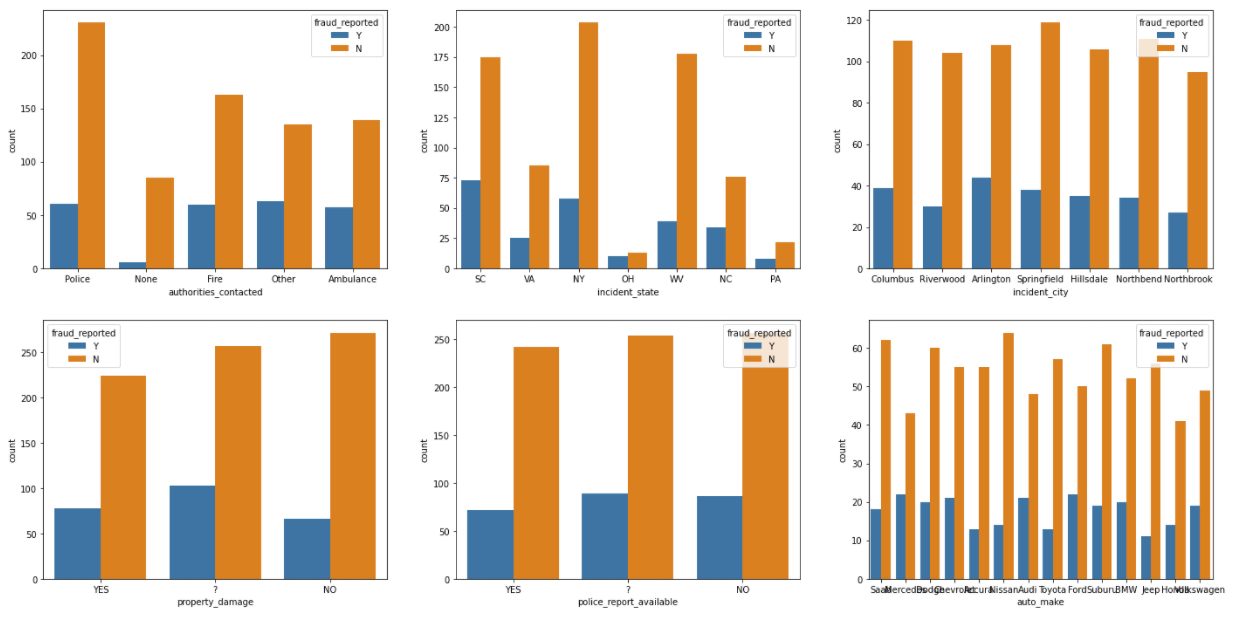




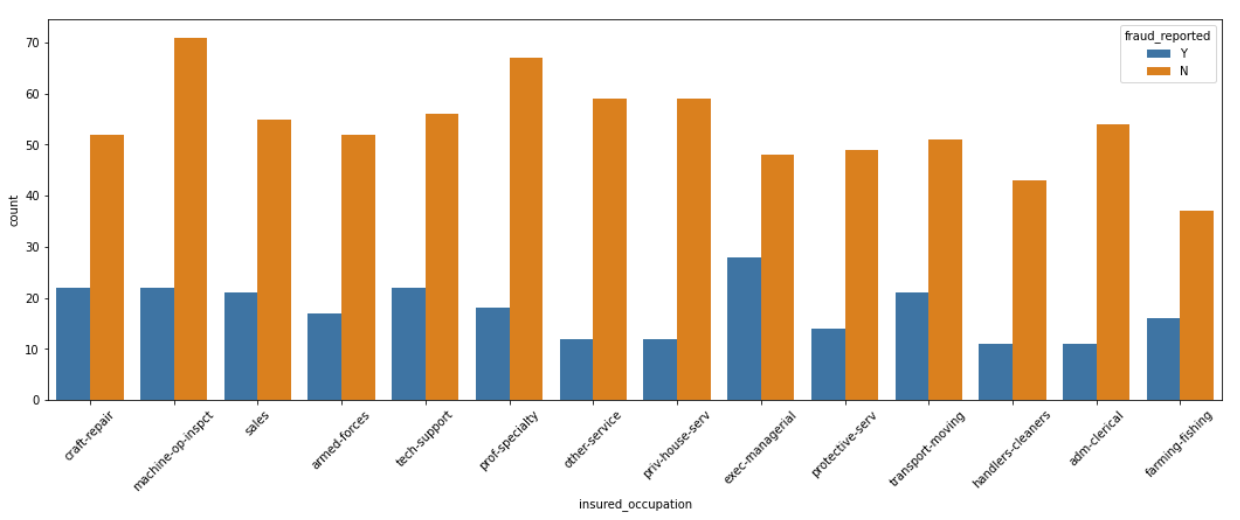
There are a greater number of females, however, same amount of two classes who insured the vehicle.



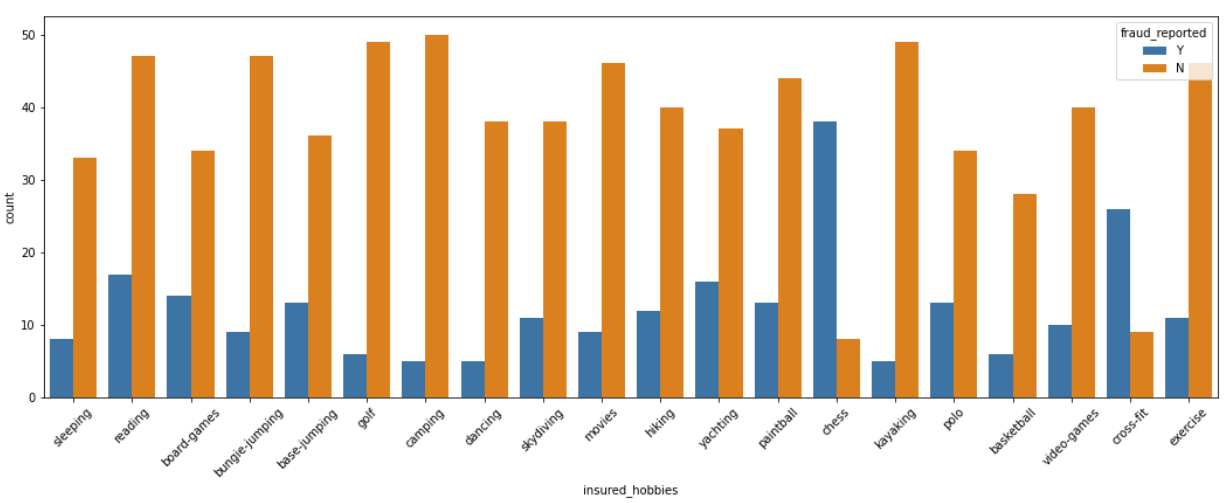
1) OH Policy\_state has more number of insurance and slighly more fraud reported numbers.  
2) There are same number of three categories in policy\_csl.  
3) insured\_relationship and insured\_education\_level has same amount of frauds reported.  
4) Same goes with incident\_city.  
5) Customers claimed to have Major\_damage in inciden\_severity has more frauds reported.  
6) In incident\_typr Single\_vehicle and Multi-vehicle collision categiry has more frauds reported.  
7) There are null values in the column collision\_type. Rear\_collision has more frauds reported.



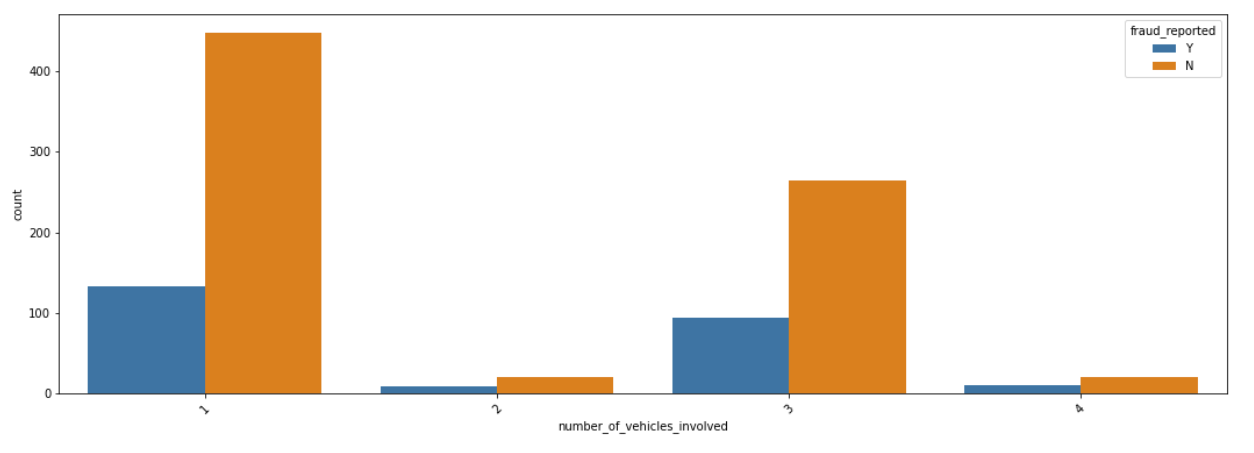
1) authorities\_contacted all the categories have equal amount of frauds detected, however, Police has been concated more.  
2) SC, NY and WV has more incidents reported.  
3) incident\_city, property\_damage, police\_report\_available and auto\_make have comparetively equal number of frauds reported.  
4) police\_report\_available and propert\_damage has missing values as "?".



exec-managerial has more customers reported as fraud.



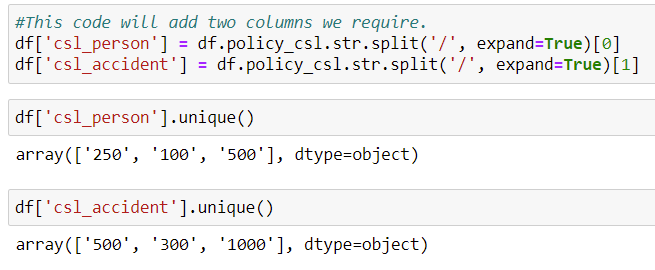
Customer with chess hobbies has significantly more frauds detected. "kayaking" has very less frauds detected.



number\_of\_vehicles\_involved with 1 has more fraud detected entries.

**PREPROCESSING**

CSL is a single number that describes the predetermined limit for the combined total of the Bodily Injury Liability coverage and Property Damage Liability coverage per occurrence or accident.  
So let's make two new columns that give us per person and per accident results.



These are categorical values; I can consider them as ordinal data.

auto\_year columns have year of manufacture of the vehicle. We will transform this column to the total age of the vehicle.



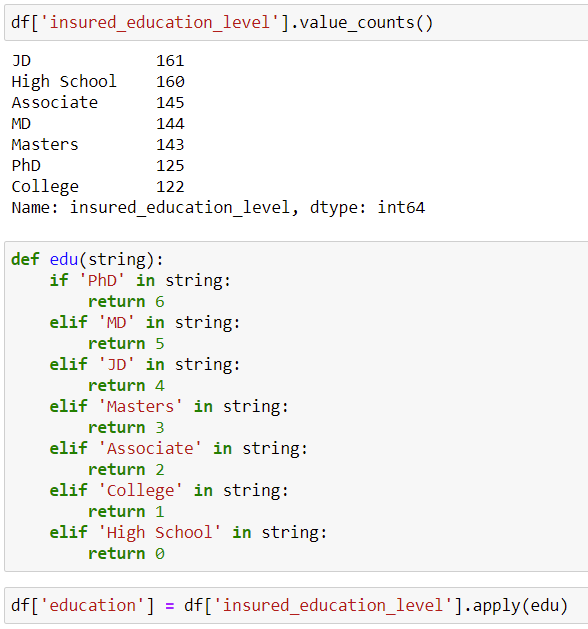
We will drop vehicle\_age column and use auto\_year.

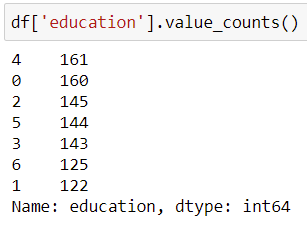
incident\_hour\_of\_the\_day has 24 categories with each hour. We will segregate them into 6 categories depending upon the part of the day.



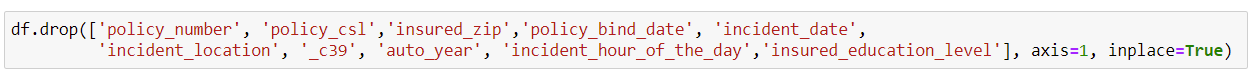
we have converted int64 column to object type.

insured\_education\_level column has categories and there is a sense of order to the education titles. So, we will map the categories starting with lowest education with 0 and highest education with high integer value. This will help our model to understand that there is a level of importance to consider while training and test of the model. Model will consider higher number as important.





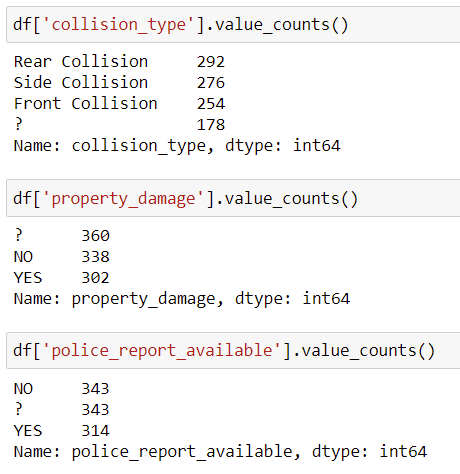
We have now converted education column into ordinal column as there is sense of order to this feature. Now we will drop the features that aren't helpful for model creation.



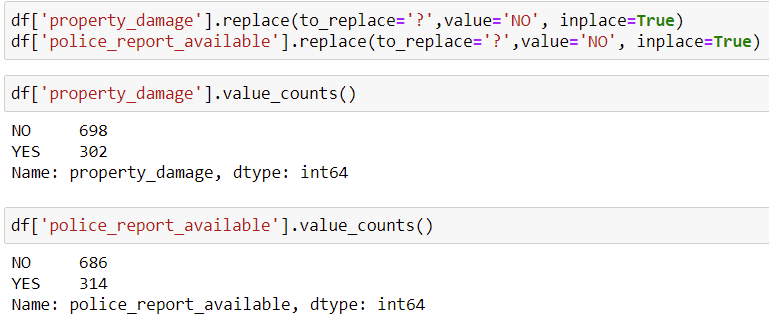
We have now dropped the features which are not useful and those we created new columns from.

**Treating missing values**

We have missing values in collision\_type, property\_damage and police\_report\_available columns with value '?'



As Property Damage and Police Report are normally yes or no, we will assume that the missing values (i.e., "?") are NO.



We treated missing values for property\_damage and police\_report\_available. We will now consider missing value in Collision Type column as a seperate category.



**Encoding**

We encode categories with two classes with LabelEncoder and columns and categorical columns with more classes with OneHotEncoder or get\_dummies method.

If you need help with LabelEncoder please follow the link below:

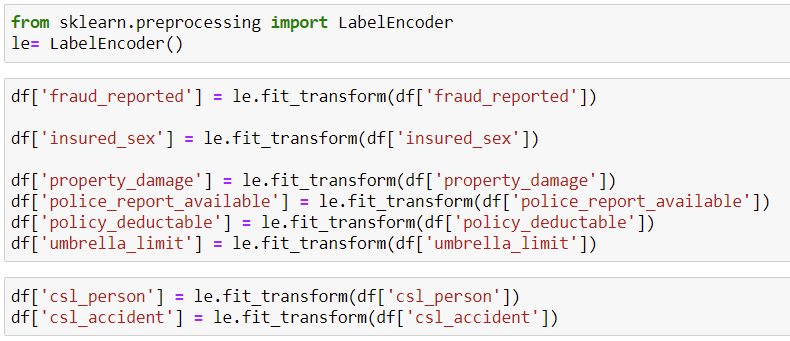
<https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html>

If you need help with, get\_dummies please follow the link below:

<https://pandas.pydata.org/docs/reference/api/pandas.get_dummies.html>

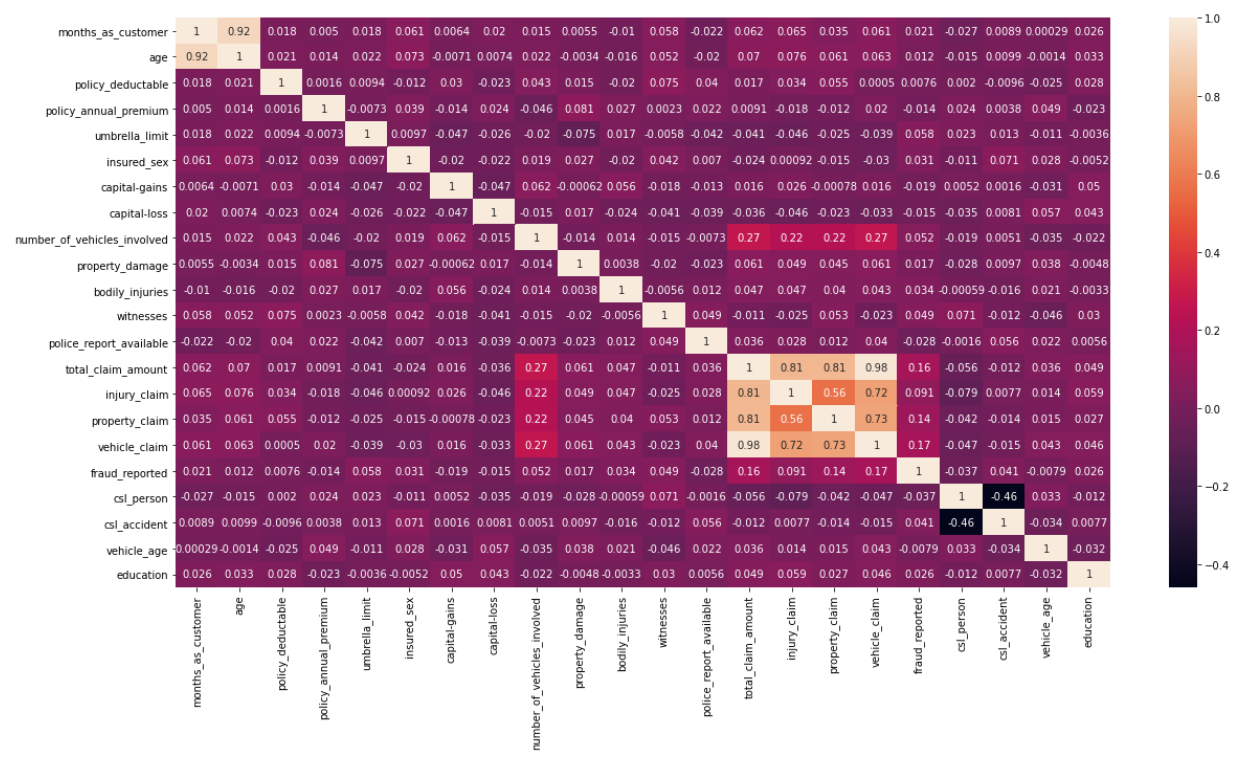
There are many types of encoding categorical data. To know more please follow below link:

<https://www.analyticsvidhya.com/blog/2020/08/types-of-categorical-data-encoding/>



1) Converted insured\_sex, police\_report\_available, property\_damage and fraud\_reported(target variable) into binary variables.  
2) Included insured\_sex, property\_damage and police\_report\_available as the features has only 2 categories.  
3) Converted csl\_person and csl\_accident using label enocoding assuming the values has a ordinal trend.

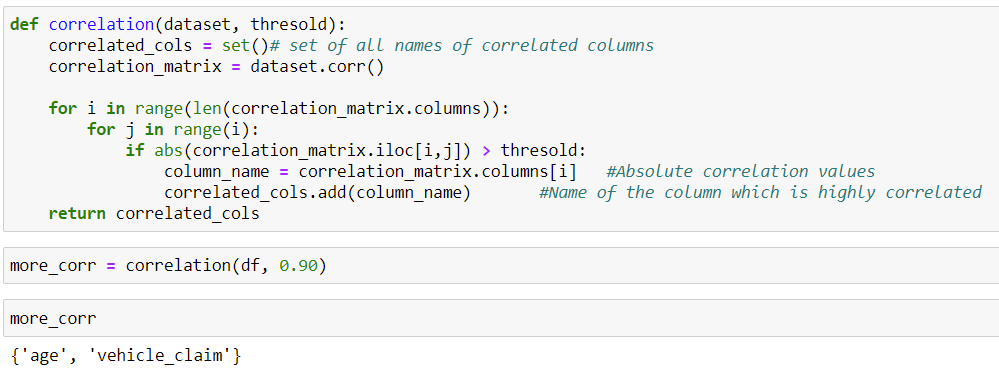
**Collinearity and Multicollinearity**

****

**What is Multicollinearity and why is it a problem?**

**Multicollinearity** exists whenever an independent variable is highly correlated with one or more of the other independent variables in a multiple regression equation. **Multicollinearity** is a **problem** because it undermines the statistical significance of an independent variable. Being said it is a significant problem in linear regression however it can seriously affect our bi classification problem. Please refer to the link below to clarify more:

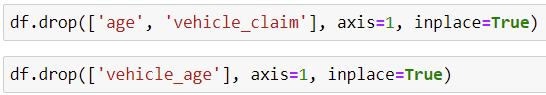
<https://stats.stackexchange.com/questions/266267/should-one-be-concerned-about-multi-collinearity-when-using-non-linear-models>



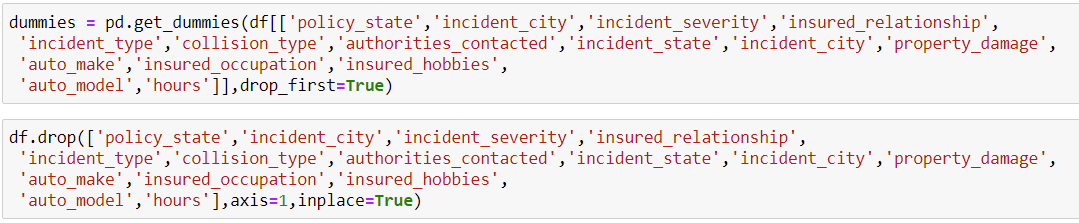
The above code will help us provide the features or columns which are highly (>90) with each other.

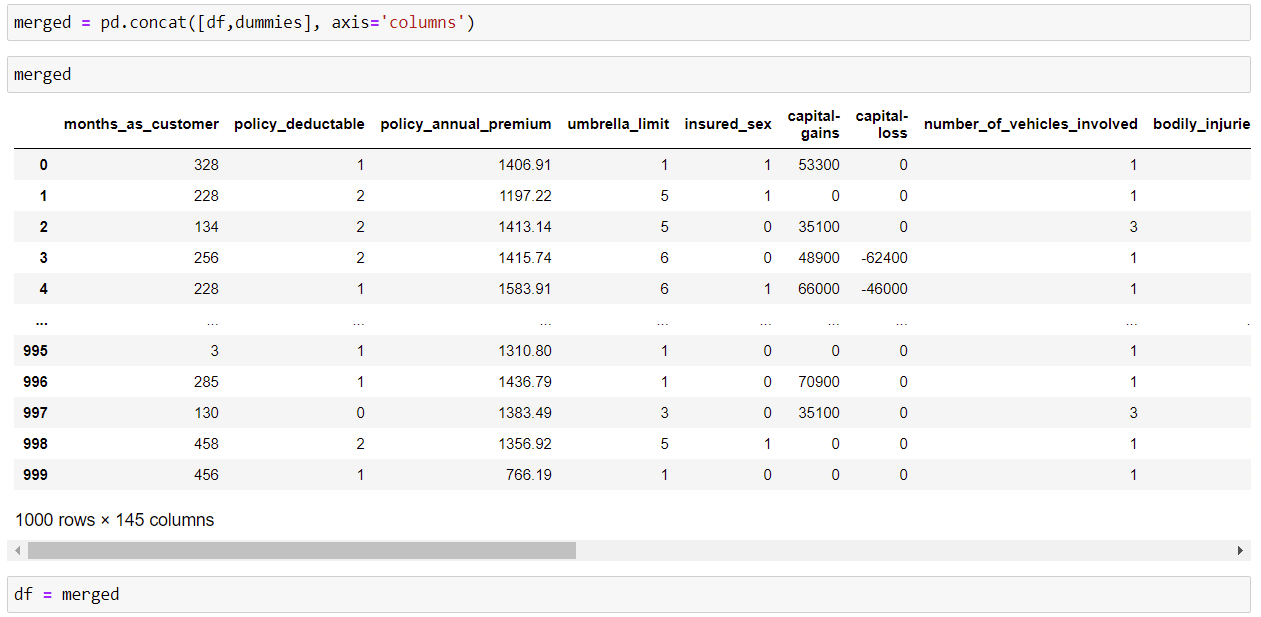
1) months\_as\_customer and age are highly correlated with each other and one of them can be processed as duplicate.  
2) total\_claim\_amount and vehicle\_claim are highly correlated with each other.  
we will drop any 2 of the above 4 columns (1 from the two mutually high correlated columns) as this would decrease the dimensions of the data set and also remove duplicity of columns which will help improve ML accuracy.

We will also drop vehicle\_age as this column is least correlated with our target column.



**Encoding with get\_dummies()**





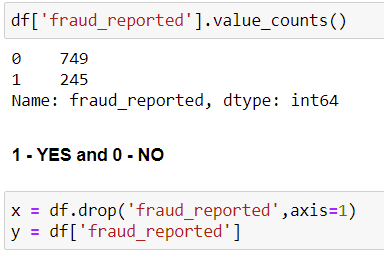
**Outliers and skewness**

We will need to treat outliers and skewness of columns which are not categorical. months\_as\_customer, capital-gains, capital-loss, total\_claim\_amount, injury\_claim, and property\_claim are the columns which needs to be treated with outliers and skewness.

As we have multiple categorical columns, we can use IQR method to treat outliers and try different transformation to treat skewness.

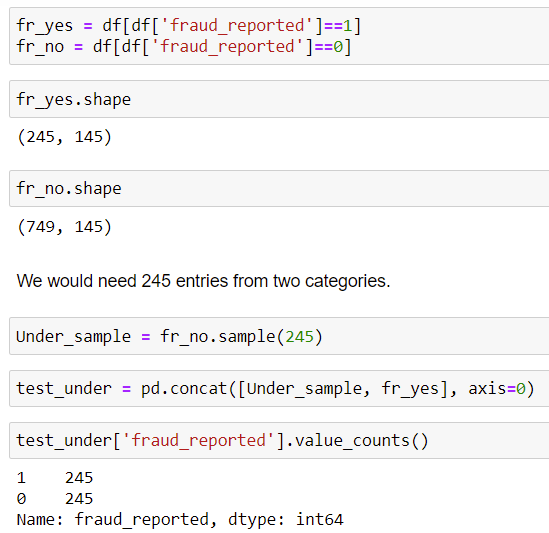
**BUILDING MACHINE LEARNING MODEL**

**Dealing with Data Imbalance Problem**

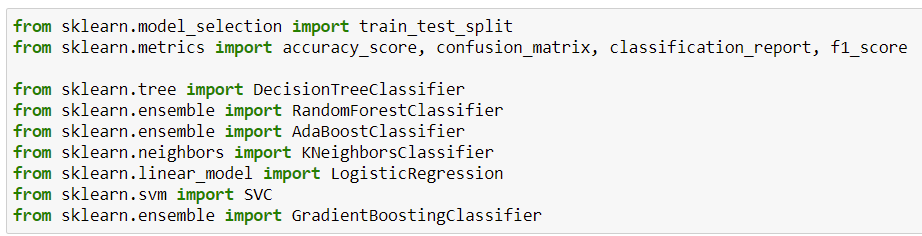
****

There are multiple techniques to deal with data imbalance problem. We will try 3 methods:

**Method1: Undersampling**

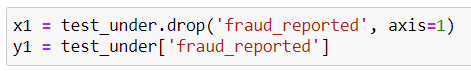


We have created a new DataFrame with same amount of two classes for this data.



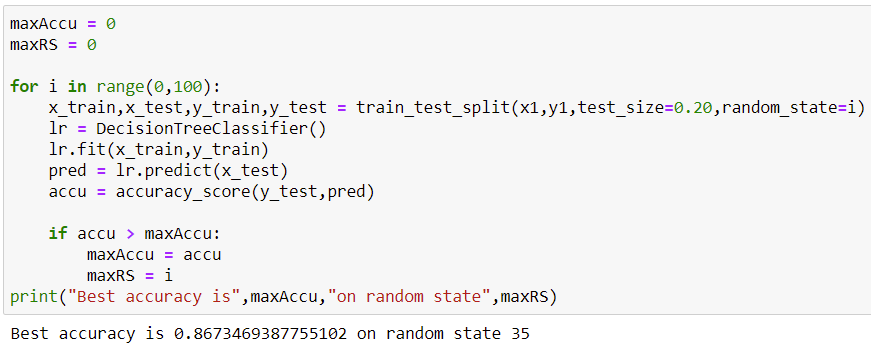
We imported:

1. train\_test\_split to split our data to train and test.
2. accuracy\_score, confusion\_matrix and classification report to understand our model performance.
3. Different Classifier algorithms.



**Finding the best random state for the model**

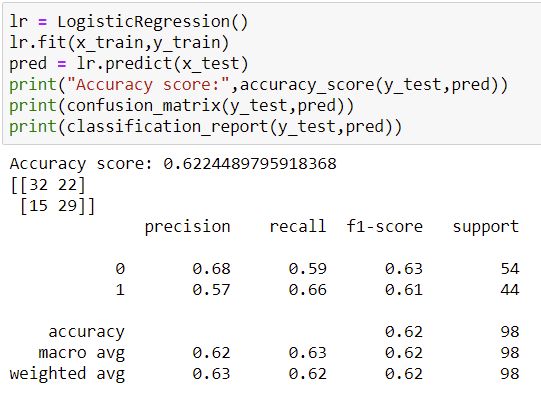
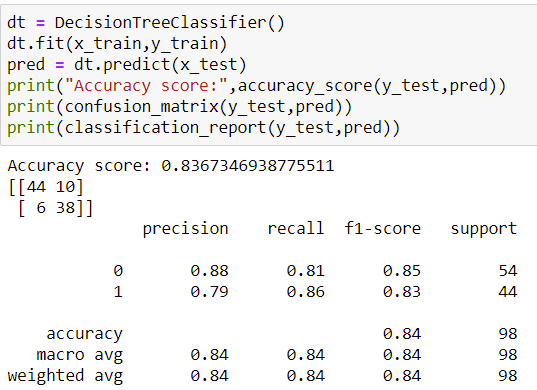
Finding a good random state for our model is important as this may increase our model performance.

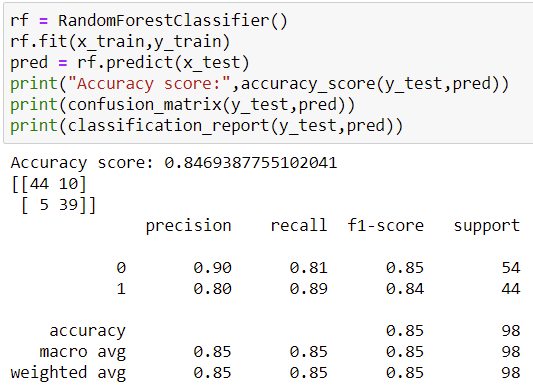
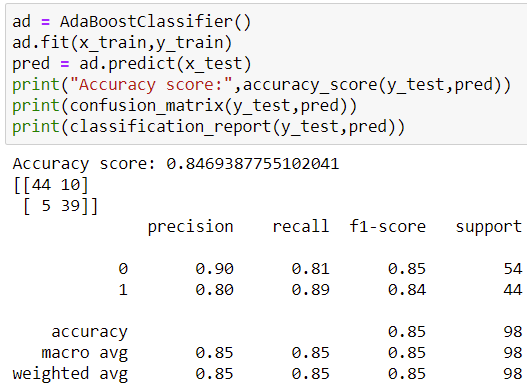


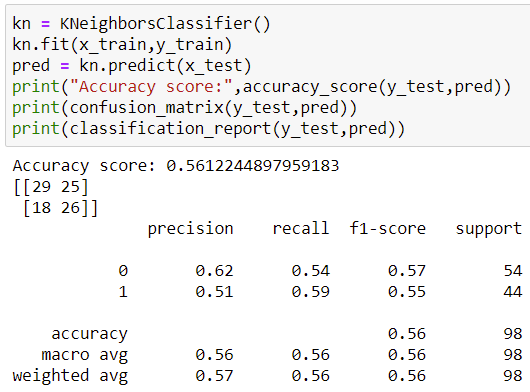
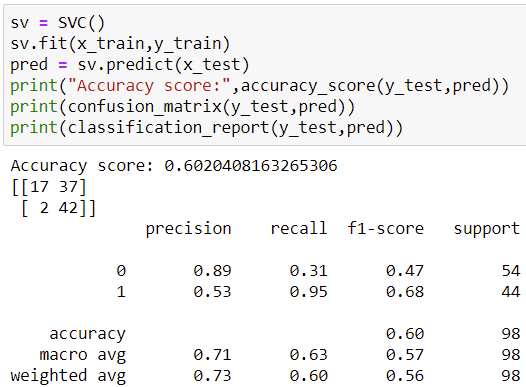
We got a good accuracy of 86.73% at random state 35. Let's test models.

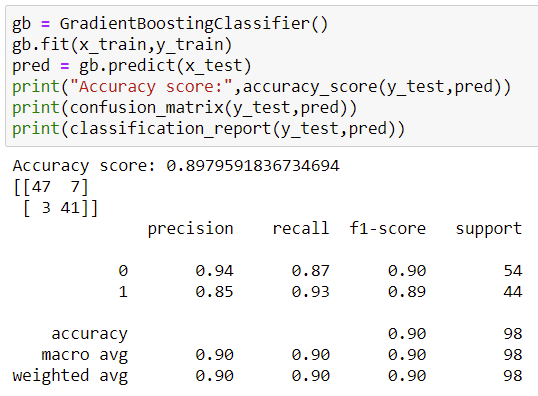


We will now test different models.



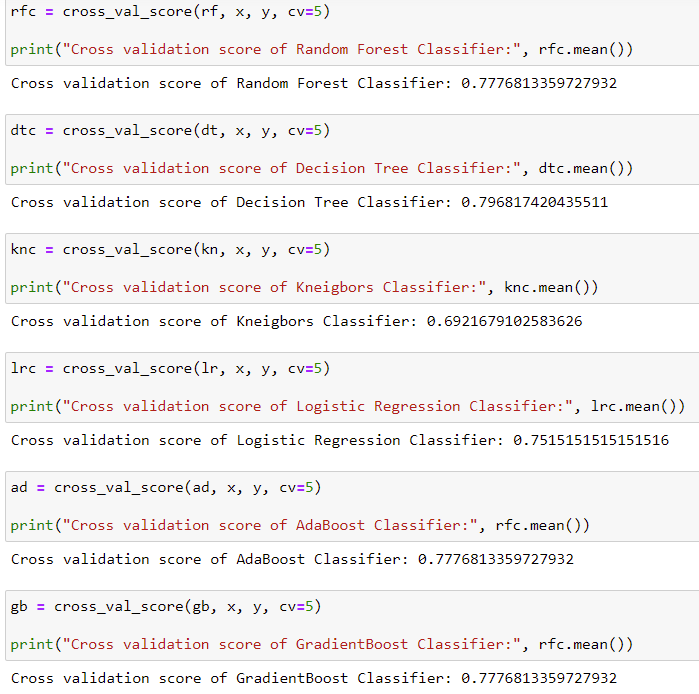
Gradient Boosting Classifier performed well. However, Let's crossvalidate.

**Crossvalidation**

**We will import** cross\_val\_score from sklearn to cross validate our model performances. We will set the cv to 5 so that our model runs 5 times and we will take the mean of the five model performances to understand the results. For more details:

<https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html>

****



Looking at the values we will consider gradient boosting algorithm for model building.

**Hyperparameter Tuning**

To do this we will need to import RandomisedSearchCv to perform hyperparameter tuning. To know more please visit:

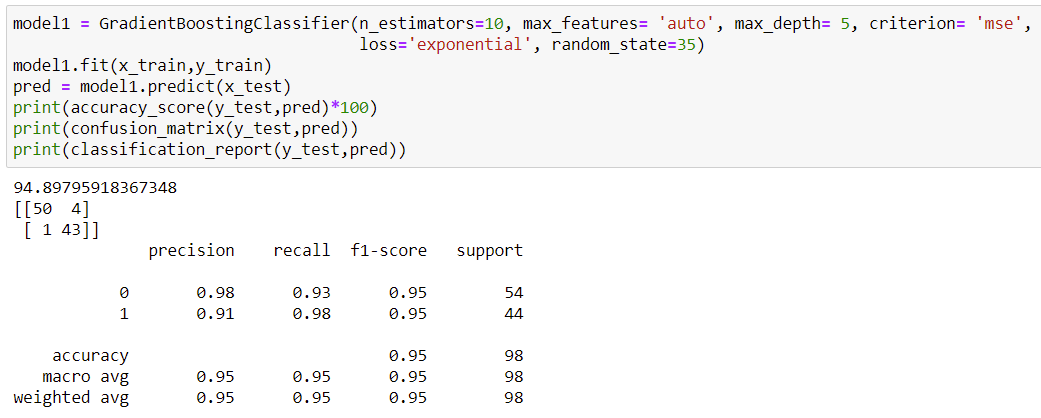
<https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html>





We give in parameters for the Gradient boosting Classifier and fit RandomisedSearchCv and retrieve the best parameters for the classifier. Using the output classifier parameters, we will build our final model.

**Model**



We got an accuracy of 94.90%, this is a very good accuracy.

We will continue the above steps using **Oversampling and SMOTE techniques** to check if we can get even more better accuracy.

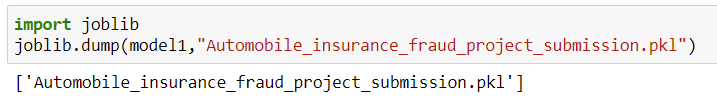
To learn about SMOTE please follow:

<https://glemaitre.github.io/imbalanced-learn/generated/imblearn.over_sampling.SMOTE.html>

**Saving the best model out of the 3 techniques.**

We will save the model which has best performance of all the three techniques. The best performance was with Undersampling technique with 94.90% accuracy.

We can use joblib to save our model.

****

**CONCLUSION**

We successfully trained a model that can identify if a particular insurance claim is fraudulent or not, based on the data, we have on the claim. To do this at scale would be futile for us humans, but the performance of the classifier shows us just how powerful these techniques can be. Classification is but one of the multitude of techniques available as part of the Predictive Modelling toolbox. I hope that this has proven to be an informative and engaging introduction to the topic.