

Insurance Claim Fraud Detection

Detecting Fraud Insurance Claims

An insurance claim is a formal request to an insurance company asking for a payment based on the insurance policy’s terms. The insurance company reviews the claim for its validity and then pays out to the insured or requesting party (on behalf of the insured) once approved.

# **Introduction**

Insurance fraud is an attempt to exploit an insurance contract. Insurance is meant to protect against risks, not serve as a vehicle to enrich the insured. Insurance fraud by the policy issuer does occur, although the majority of cases have to do with the policyholder attempting to receive more money by exaggerating a claim.

In this article, we will be studying auto insurance frauds -

In India, it is estimated that every year a sum of INR 30 Crore is lost to false claims and scams under car insurance. While people are aware of insurance frauds, they might not be aware of how it is done. Different methods are often used by scammers and frauds to squeeze out money from auto insurance companies without any ounce of verity to the situation.

## 

## Problem Statement :

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.

To solve this problem we are provided a dataset that has the details of the insurance policy along with the customer details. It also has the details of the accident based on which the claims have been made. By building a good model we will able to save the Insurance industry from the huge amount of losses.

## Dataset :

The dataset given can be found from the following link :

<https://github.com/dsrscientist/Data-Science-ML-Capstone-Projects/blob/master/Automobile_insurance_fraud.csv>

The dataset contains different features and information on the employee which are listed below :

1. months\_as\_customer - Total number of months of the customer
2. age - age of the policyholder
3. policy\_number- policy number
4. policy\_bind\_date - The date of policy binding
5. policy\_state - The state where the policy was done
6. policy\_csl - Combined single limit of the policy
7. policy\_deductable - Amount that customer is responsible for paying for an insured loss
8. policy\_annual\_premium - The total amount of premium paid annually
9. umbrella\_limit - Extra insurance that protects existing limits
10. insured\_zip - zip code of the insured
11. insured\_sex- gender of the insured
12. insured\_education\_level - Education level of insured
13. insured\_occupation - Occupation of insured
14. insured\_hobbies - Hobbies of insured
15. insured\_relationship - Relationship of insured
16. capital-gains - Profit made on the sale of capital assets
17. capital-loss - Loss encountered on the sale of capital assets
18. incident\_date - date of the incident
19. incident\_type - type of incident
20. collision\_type - collision type of the automobile
21. incident\_severity - the severity of the incident
22. authorities\_contacted - authorities contacted by the insured after the occurrence of the incident
23. incident\_state - State where the incident occurred
24. incident\_city - City of incident
25. incident\_location - location of incident
26. incident\_hour\_of\_the\_day - hour of incident
27. number\_of\_vehicles\_involved - number of vehicles involved in the incident
28. property\_damage - the property damage incurred by the insured
29. bodily\_injuries - number of people injured during the incident
30. witnesses - witnesses of the incident
31. police\_report\_available - If the police report is available for the incident or not
32. total\_claim\_amount - total amount claimed by insured
33. injury\_claim - a claim made for injuries
34. property\_claim - claim for the property damage
35. vehicle\_claim - a claim made for the vehicle
36. auto\_make - the manufacturing company of the vehicle
37. auto\_model - the model of the automobile
38. auto\_year - the model year of automobile
39. fraud\_reported - Fraud reported is the target feature using which we will predict if the claim is fraud or not

## Contents of the Article:

The article explains all the steps end to end to understand the attrition through exploratory data analysis and then build a machine learning model to predict the attrition of an employee.

Following are the steps that are taken to build this project which we will be covering in this article:

1. Exploratory Data Analysis
2. Feature Engineering and preprocessing
3. Feature Selection
4. Model Building and Evaluation
5. Hyperparameter Tuning
6. Saving the model for predictions and deployment
7. Conclusion

So let’s start with the exploration of the dataset and build a good prediction model.

# Exploratory Data Analysis

Exploratory data analysis is a statistical approach to analyze data and summarise the main characteristics of the data which gives us insight into the business or any use case or problem statement.

In EDA we generally explore the data for the below-mentioned point:

- Summary of the dataset

- Missing values

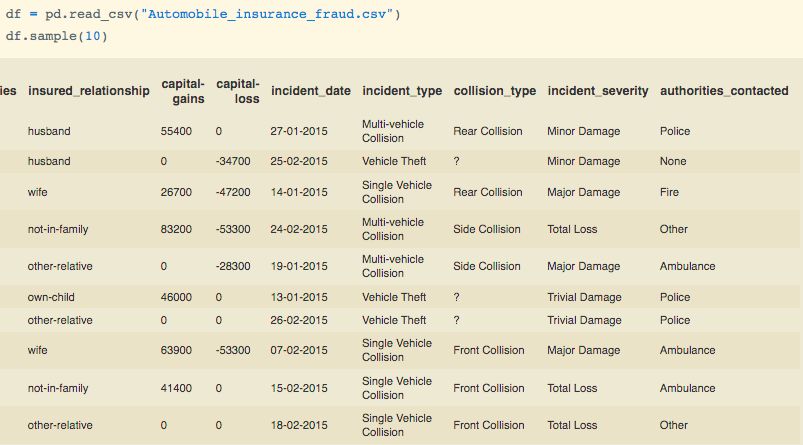
- Detecting outliers

- Analyzing types of variables

- Distributions of variables

- Relationship between independent and dependent features

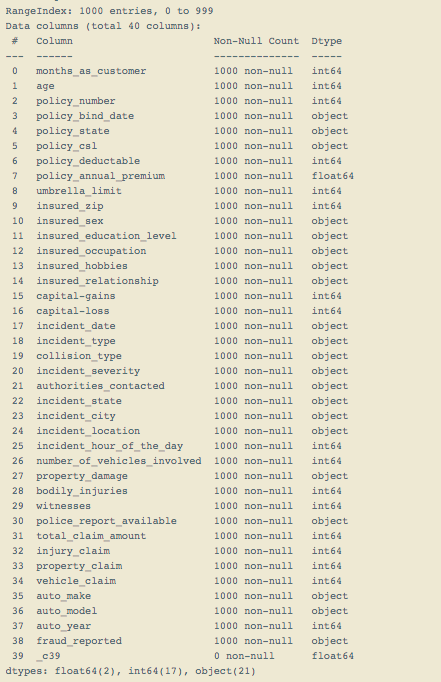
To begin with first will be importing libraries important for EDA and other important libraries required in this project and will load the dataset into jupyter notebook using pandas read\_csv method, and will have a look at the sample dataset.



After looking at the sample records of data we can identify the following points:

1. There are both categorical and numerical features
2. Some records in collision\_type column are missing
3. There is a date type variable as well

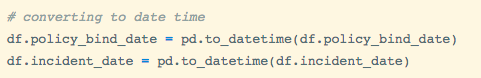
To get more detail we will check info about the dataset using info() method to get detailed information about the dataset.

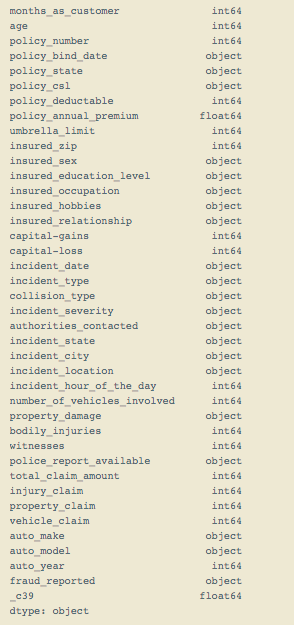


From the info of the dataset, we can see that there are a total of 40 features, fraud\_reported being the target feature and 17 variables are integer type variables, 2 float type variables, and 21 are object type variables.

Also, there date type variable which is policy\_bind\_date and incident\_date but has been classified as object type so will be converting them to the date-time variable. In addition to that, we can see that “\_c39” variable has 0 non-null values which means that the column is empty.

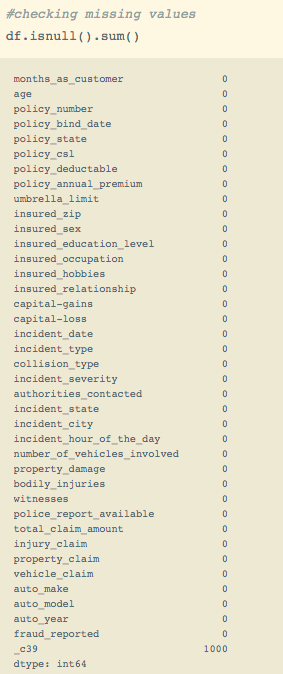
Will be converting policy\_bind\_date and incident\_date to date-time type variable using pandas to\_datetime method and will be dropping incident\_location feature as it does not give any relevant information for prediction.



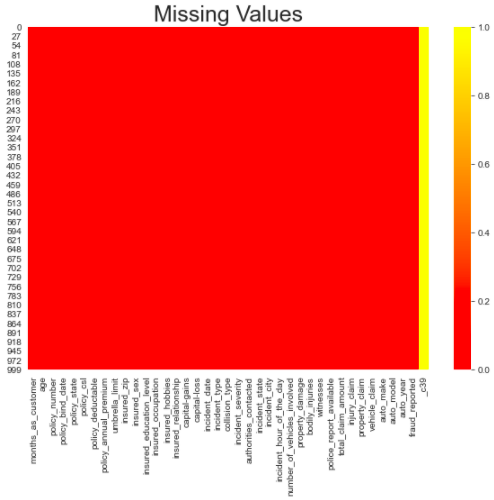


We can see that the data type of policy\_bind\_date and incident\_date has changed from object to date-time type.

Moving further will check if the dataset has missing values.



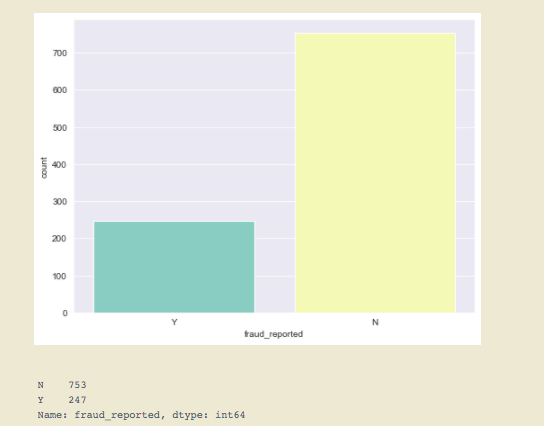
We can also visualize the missing values using heatmaps -



As we can see that ‘\_c39 is missing so we will drop it

Analyzing Target Variable -

Here I will be having a look at the distribution of classes in the feature -

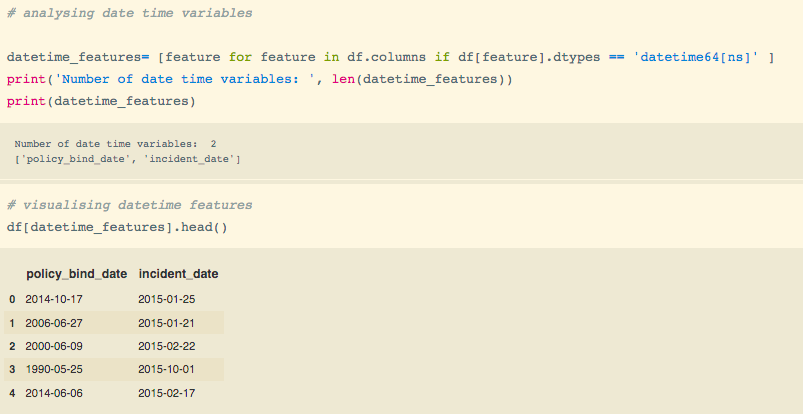


From the above visual we can conclude that the classes have a quite larger difference in their ratios of distribution, it looks like an imbalance in the classes.

Moving ahead since the target feature “fraud\_reported” is a categorical variable and we have to analyze relationships with it so will encode its categories by mapping them with integer values, due to which we will also be able to check its correlation with numerical features, also we can do some more analysis with it.

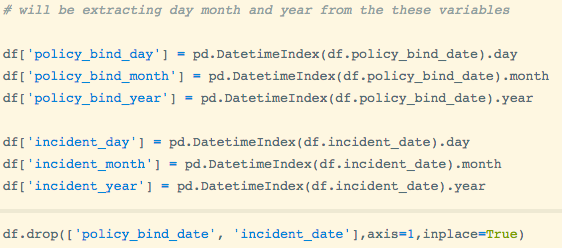


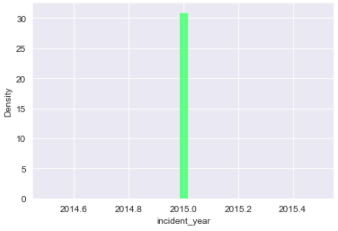
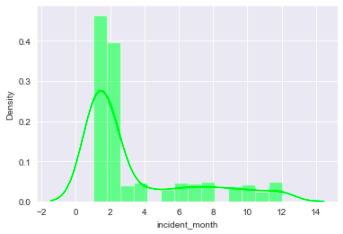
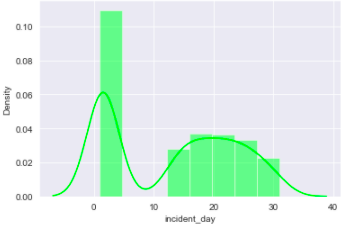
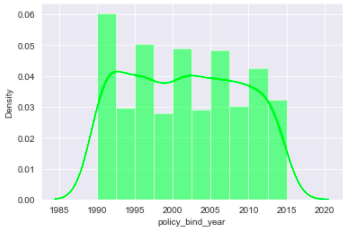
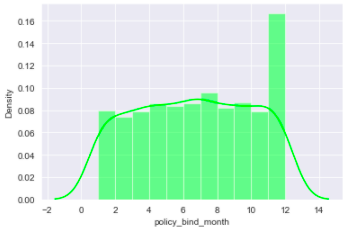
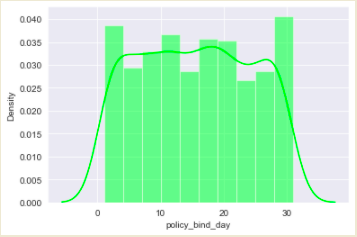
Now let’s do some analysis with the date-time variables.



Here I am checking some samples of the date-time variables policy\_bind\_date and incident\_date. We, Will, extract month, year, and day from the features so that we can analyze the frauds reported better.

To extract I have used the DateTimeIndex method from the pandas library to extract the month year and day and then will drop variables policy\_bind\_date and incident\_date.

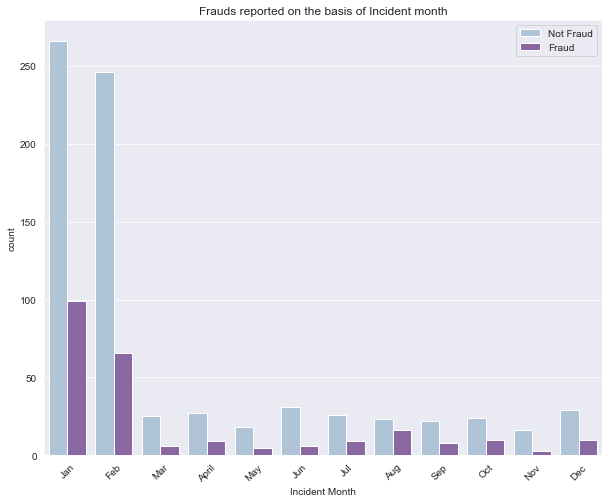


Now let’s check the distribution of extracted elements day, month and year - 

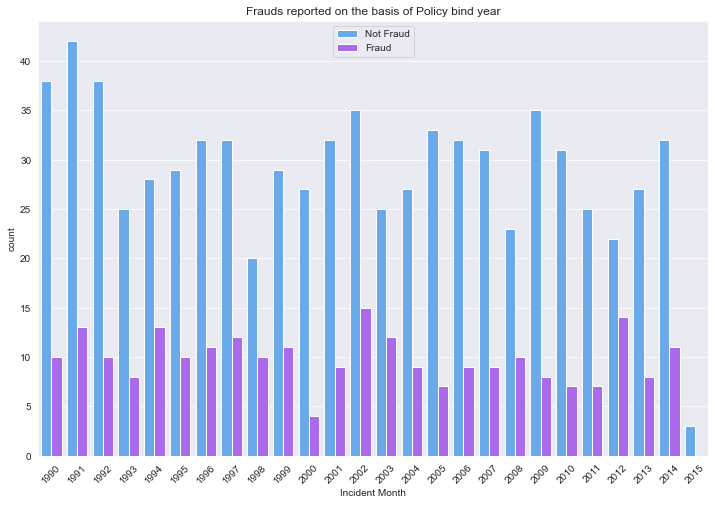
From the above plots we can conclude following points:

* Policy\_bind day is somewhat uniformly distributed, implying that policy binding can be done on any day of the month.
* Policy\_bind\_month is not normally distributed, as we can see that a major number of policies are bound in November and December
* Ploicy\_bind\_year has the highest frequency in the year 1990 and is not normally distributed
* Incident\_day is highly skewed to right, which can imply that major incidents occur on the initial days of the month
* Incident\_month is also right-skewed for the initial months
* Incident\_year has only one value 2015

Moving further let’s visualize the frauds\_reported with year, month, and day to get more insight into the data -



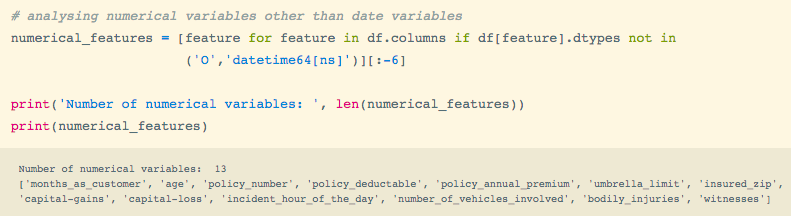




From the above visualizations, we can conclude that :

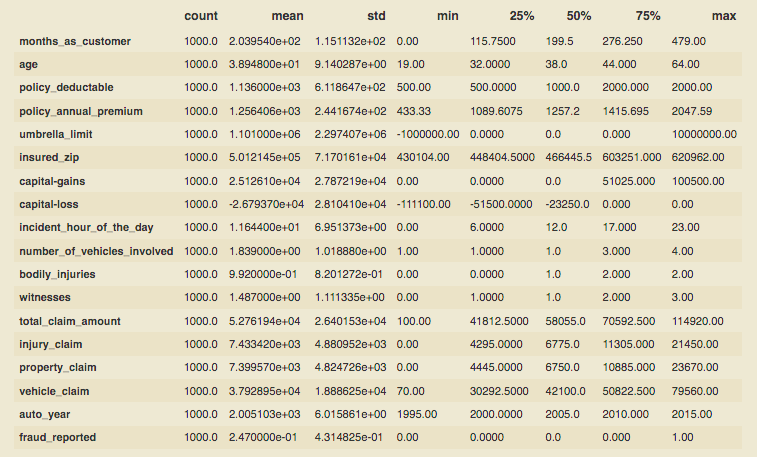
* Majorly frauds based on incident month occur most in January and February as compared to other months
* There is no specific relation of fraud with policy bind month, which means frauds who get a policy can get the policy in any month there is no specific month when mostly frauds get the policy.
* There is no specific relation between frauds and incident year

Moving ahead let’s analyze the numerical features -



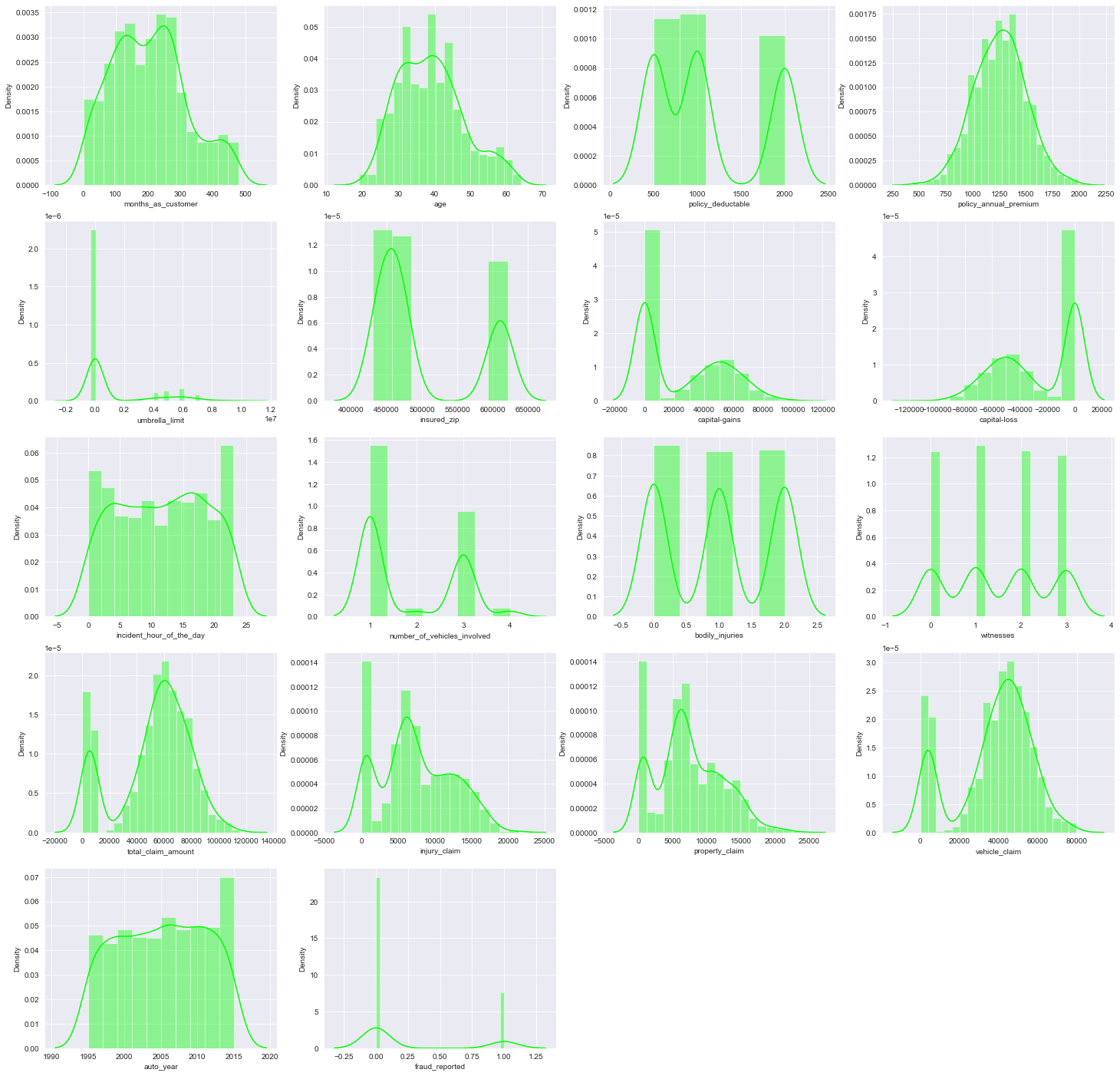
We can see from above that there are 13 numerical features so let’s explore these features more.

We can check the statistical summary with the help of command data.describe() -



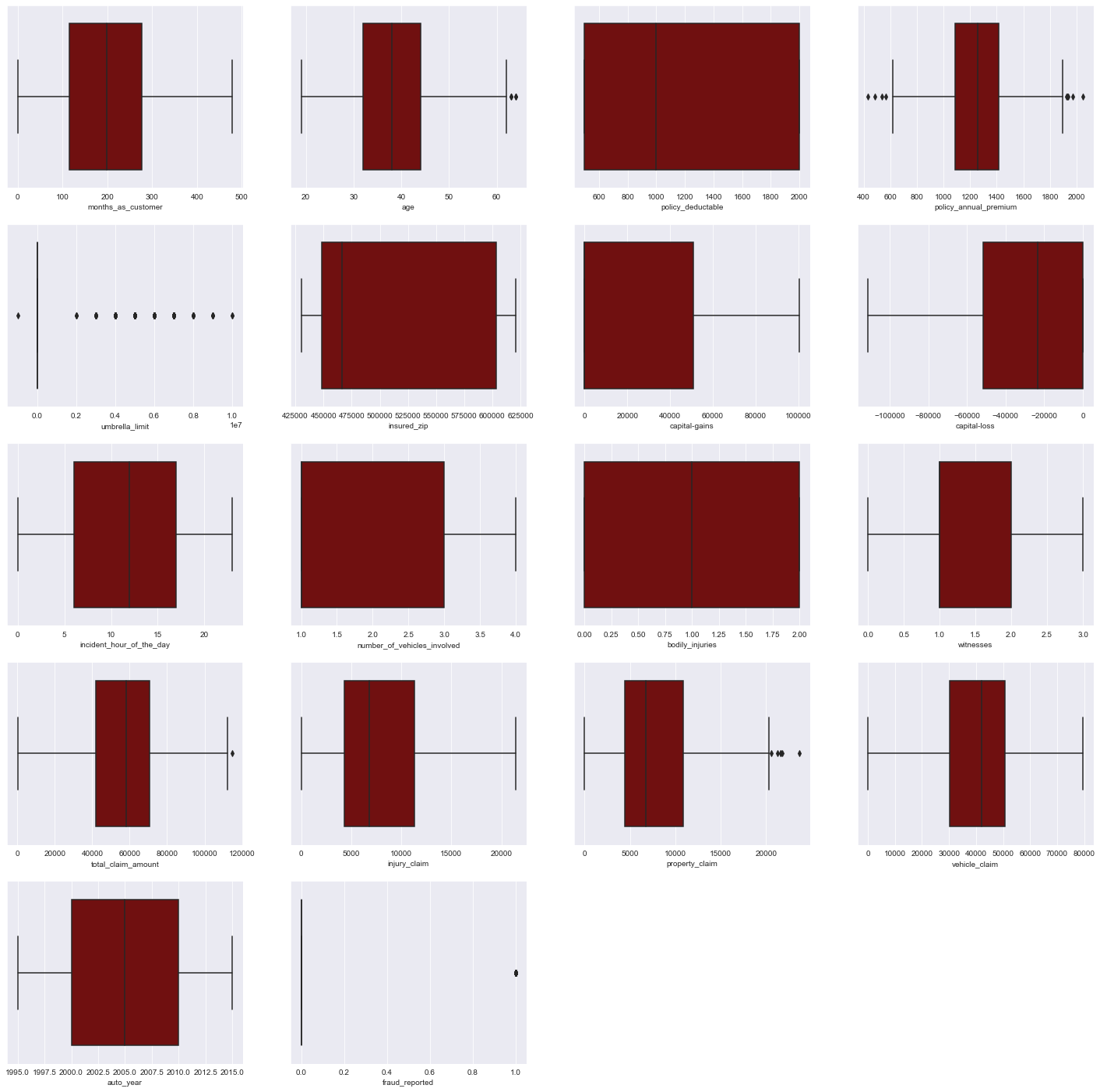
* From the above table, we can see the statistical summary of the variables
* The standard deviation for some features is quite high which implies the data is widely spread with some abnormality in the distribution, they might not be symmetrically distributed.
* Also, some of the variables are somewhat normally distributed with less skewness since the mean and median do not have much difference

Moving further let’s check the distribution to get ore statistical summary on the features visually -



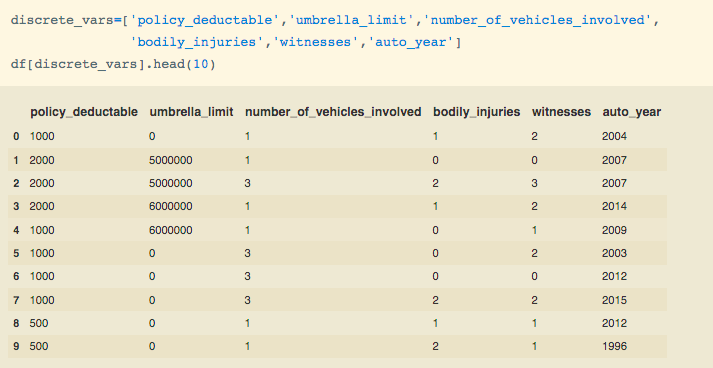
* From the above plot, we can see that age is normally distributed
* Some features are having multiple peaks that imply that they are discrete
* Some of the features are not normally distributed they are skewed

Moving further let’s visualize the outliers -

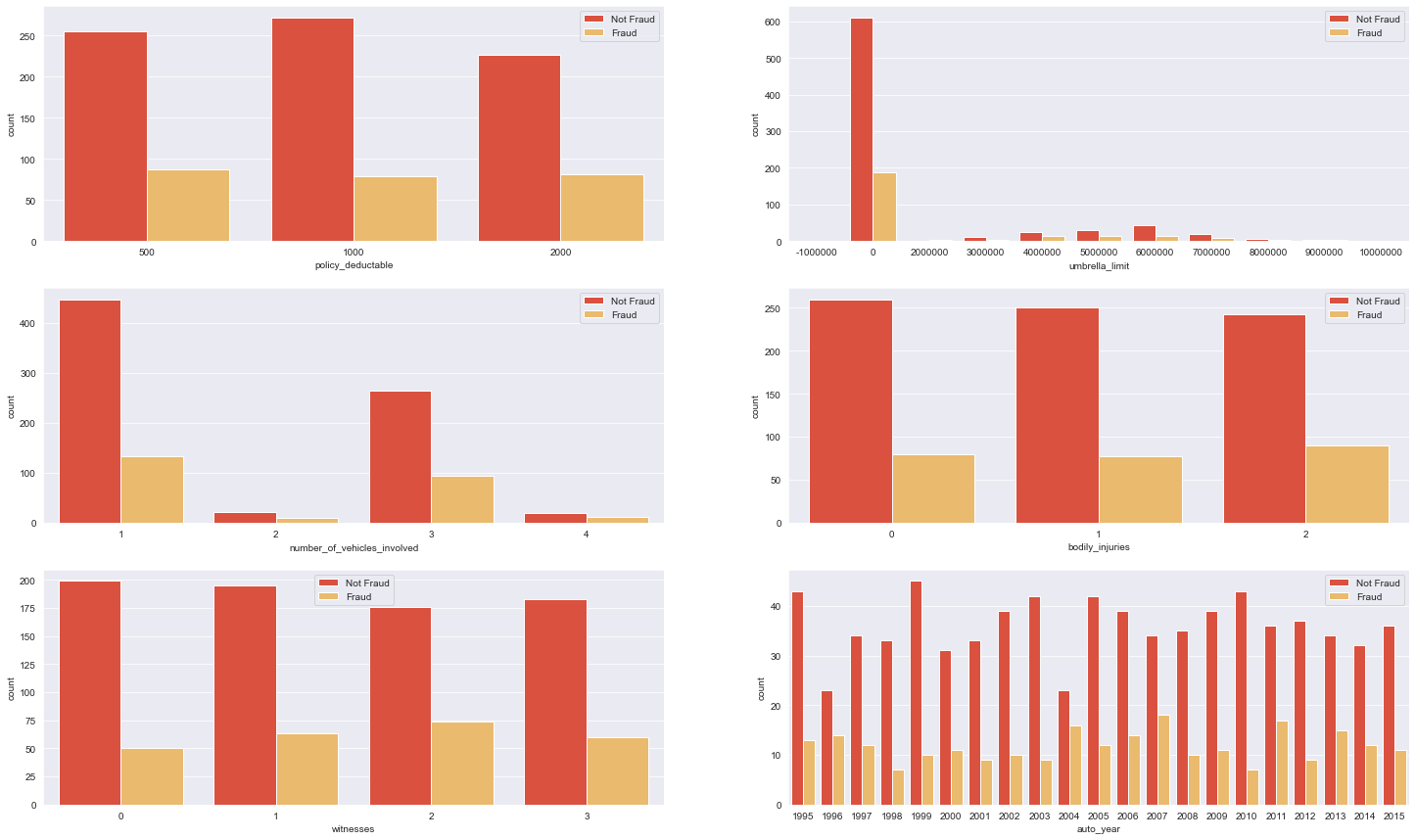


* Can see some outliers in the dataset from the above plot
* We, Will, be keeping the outliers in our dataset as the problem statement is for fraud detection, and outliers can represent frauds, therefore will keep the outliers and will scale down the abnormal values in feature engineering

The numerical features we analyzed were continuous let’s have a look at the discrete variables -

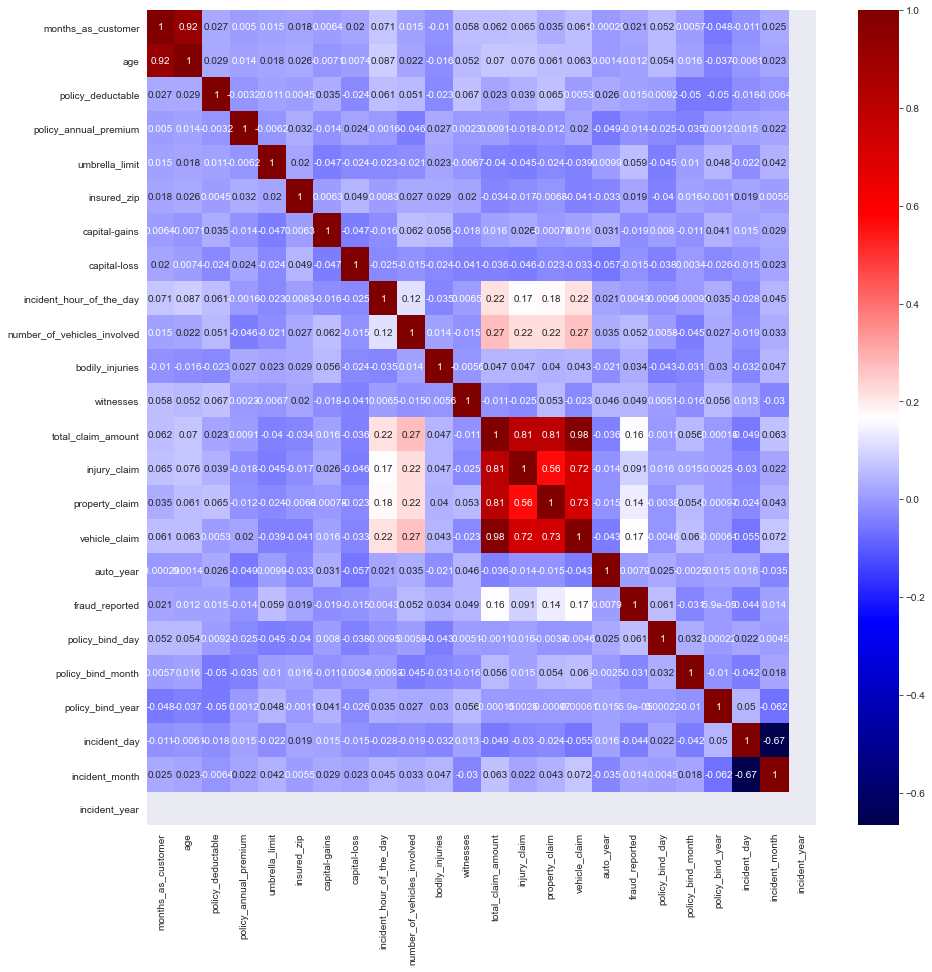


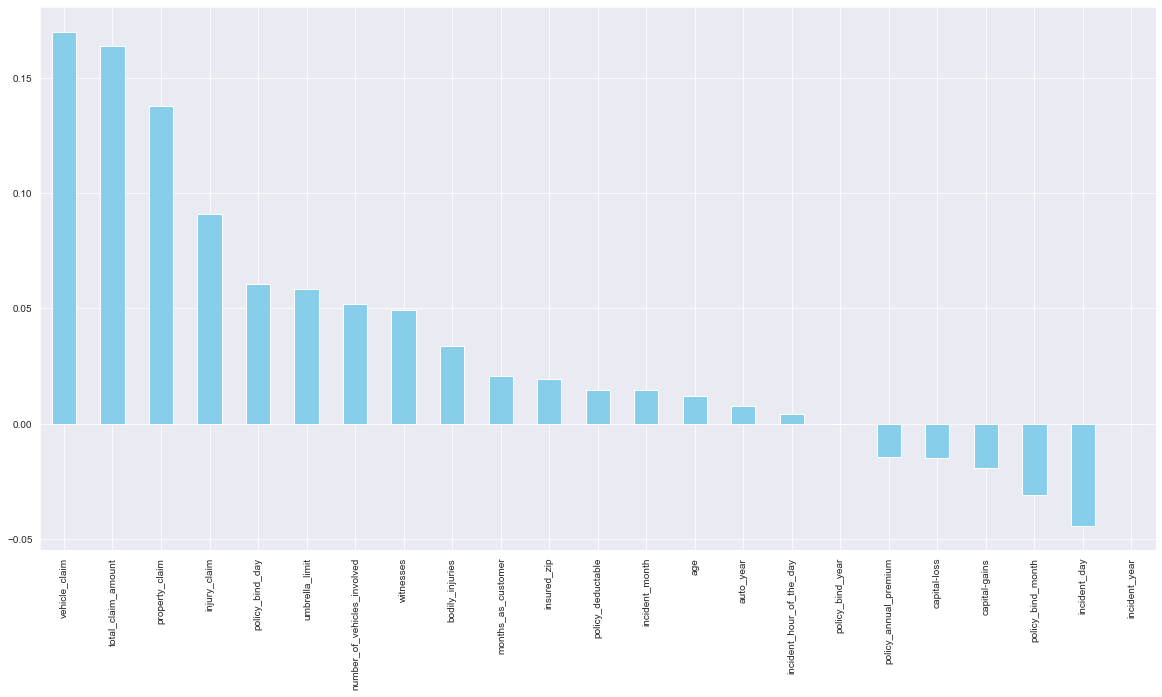
We can see from above the discrete variables let’s try to visualize them with target variable -



* There is no specific relation between fraud reported and policy deductible
* Usually, Frauds have an Umbrella limit of 0
* If the insured claims that the number of vehicles involved in 1 or 3 there are quite good chances of him or she being a fraud

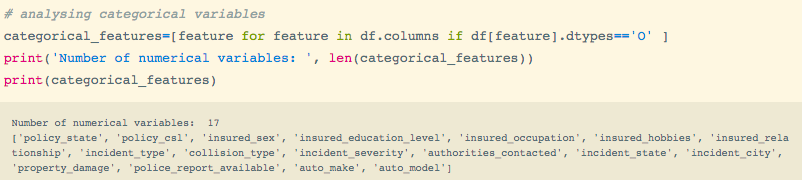
Proceeding further will check the correlation of the variables and correlation of target variable with independent features -

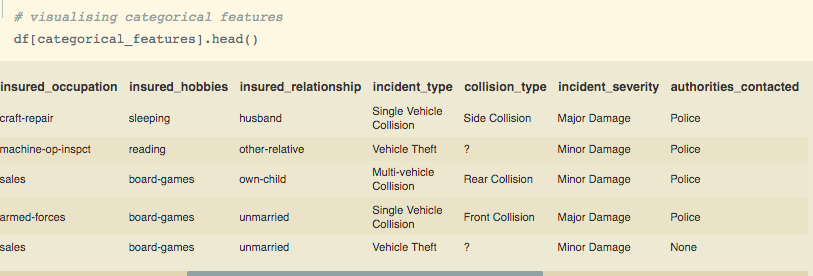




* From above we can see the correlation between the variables
* Can see higher multicollinearity between independent variables
* Also, policy\_bind year and incident\_year do not correlate with the target variable

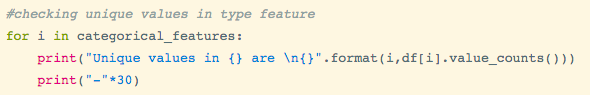
Now let’s analyze the categorical variables

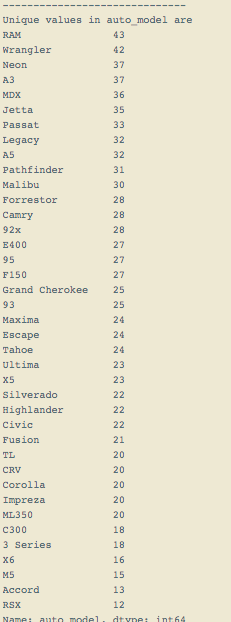
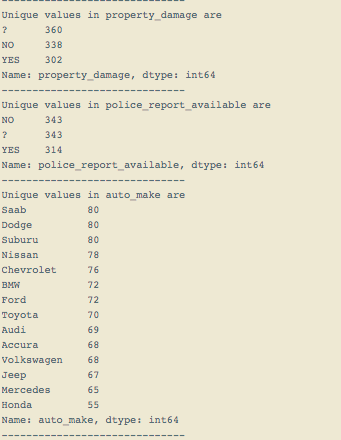
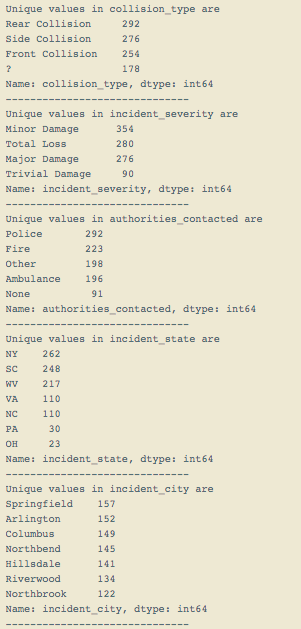
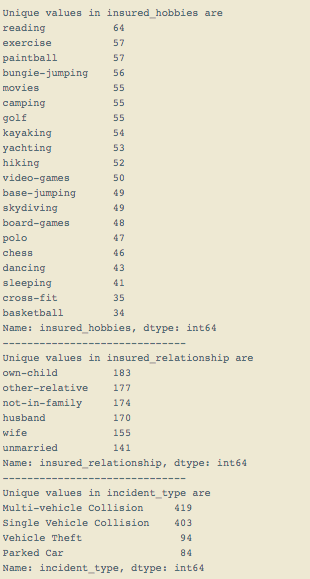
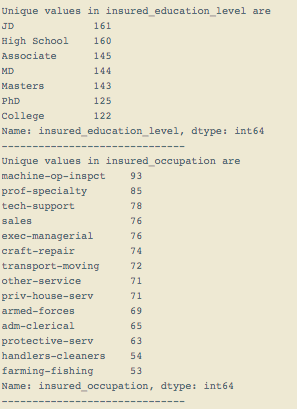
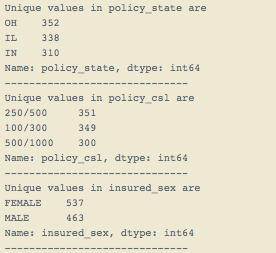




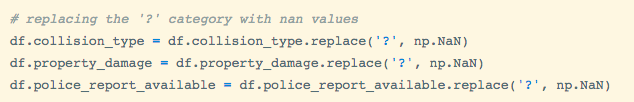
We can see that there are 17 categorical features in the dataset and by looking at the head we can see that there are some unknown values represented as ‘?’ which means the values are missing so will check the unique values in all the categorical features and will replace these types of characters by nan values which we treat further.

Checking Value counts of the features -

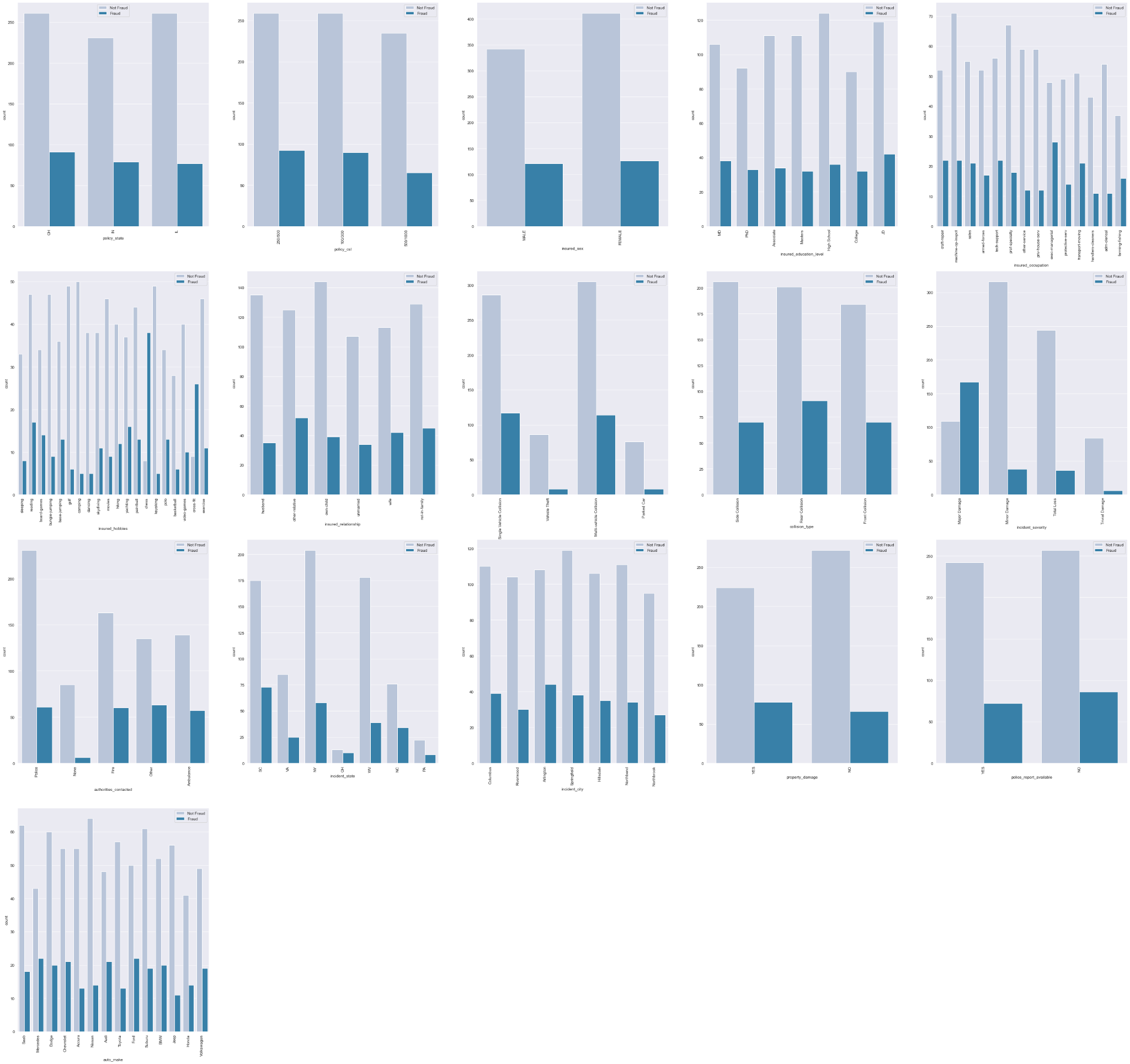




From above analysis we can see that that there are ‘?’ in collision\_type, property\_damage, police\_report\_available which we will replace by nan values.



Moving further let’s analyze the categorical variables with fraud\_reported -



From the above analysis, we can conclude that :

* Policy state, gender, and policy\_csl does not have any specific relationship with fraud reported cannot specifically say for these features
* Majorly non-frauds are only high school qualified
* Majorly non-frauds have machine operation inspectors and majorly frauds have a managerial executive as occupation
* Most frauds have hobbies as playing chess or Crossfit
* Mostly frauds claim to be other relatives as relationship
* Nonfrauds generally claim to have multi-vehicle collision
* Frauds generally claim that they had a rear collision
* People who claim minor damage are more likely non-frauds but people claiming major damage can have chances of being a fraud
* Majorly Non-frauds tend to contact police
* Mostly frauds claim the incident state like South Carolina
* Mostly frauds claim the incident city as Arlington
* Mostly for frauds, there is no property damage
* Majorly Frauds do not have any police reports registered
* Frauds can claim his or her vehicle from any of the automobile makers

As we have explored the dataset from all aspects and have analyzed their distribution and relation with target variables, so we will now move to feature engineering.

# 

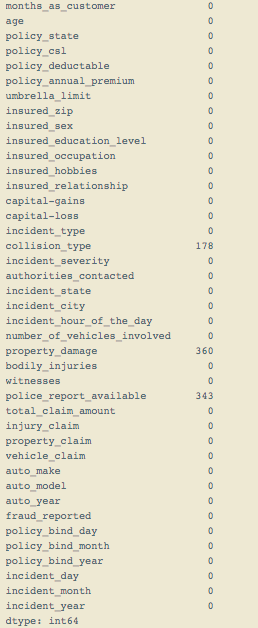
# Feature engineering

In this process we will :

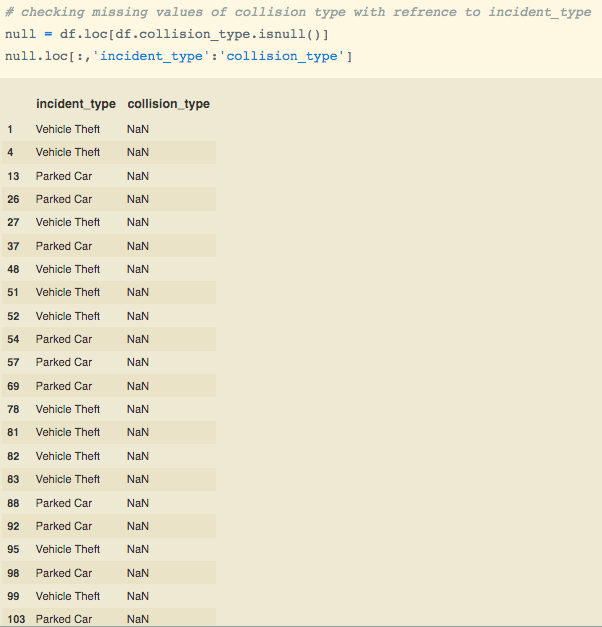
* Treat Missing values
* Transform the skewed variables
* Encode the categorical variables

So Let’s proceed with treating missing values.

We treat the missing values which we got from the previous step where we replaced ’?’ with ‘np.nan’ -



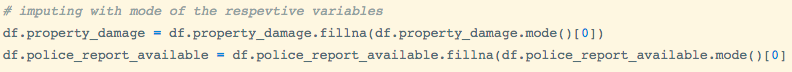
We will be replacing nan values collision\_type first -



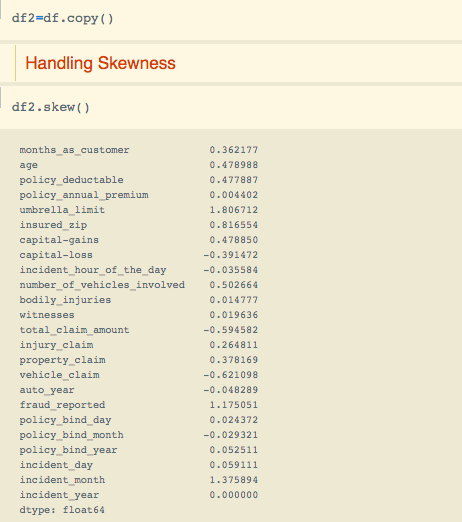
As we can see that collision type is missing only for incident\_type being either theft or parked car so will be categorizing these values as 'No Collision'.



For property\_damage and police\_report\_available will be replacing missing values with the modes of the respective columns.

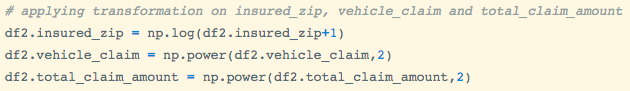


As we have treated the missing values let’s move to the transformation step where we will be transforming the variables to reduce the skewness. For that, we will check the skewness of the dataset.



As we can see that insured\_zip is positively skewed and vehicle\_claim, total\_claim\_amount are negatively skewed.

So I will apply log transformation on insured\_zip and power transformation on vehicle\_claim and total\_claim\_amount.

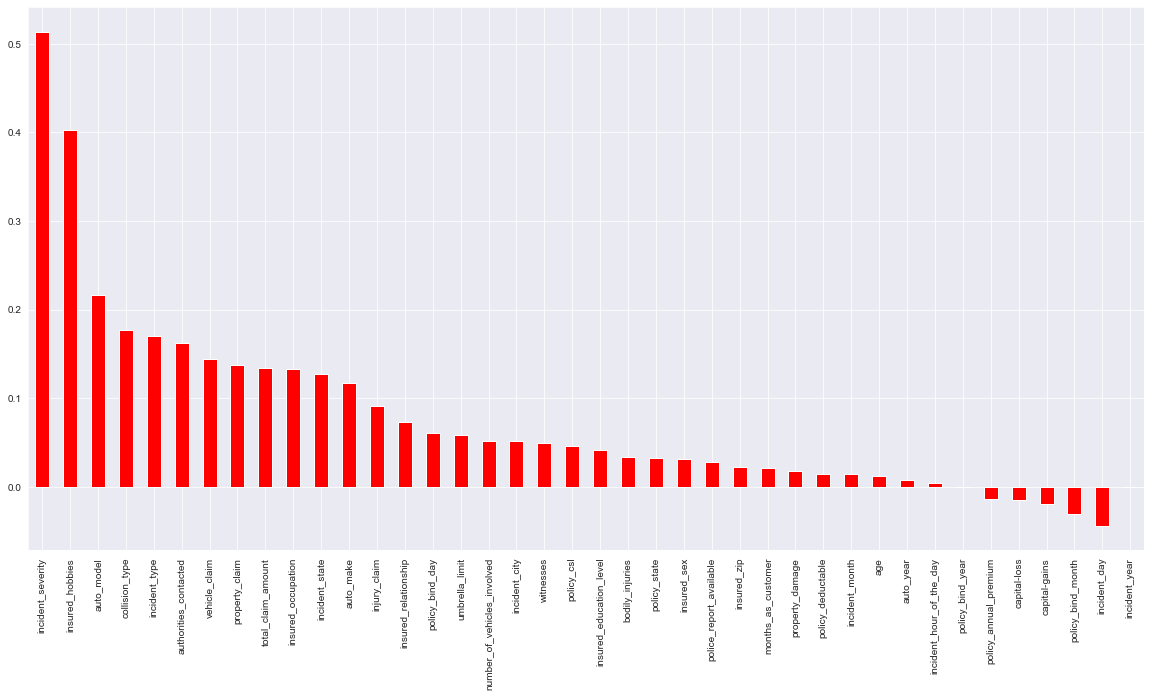
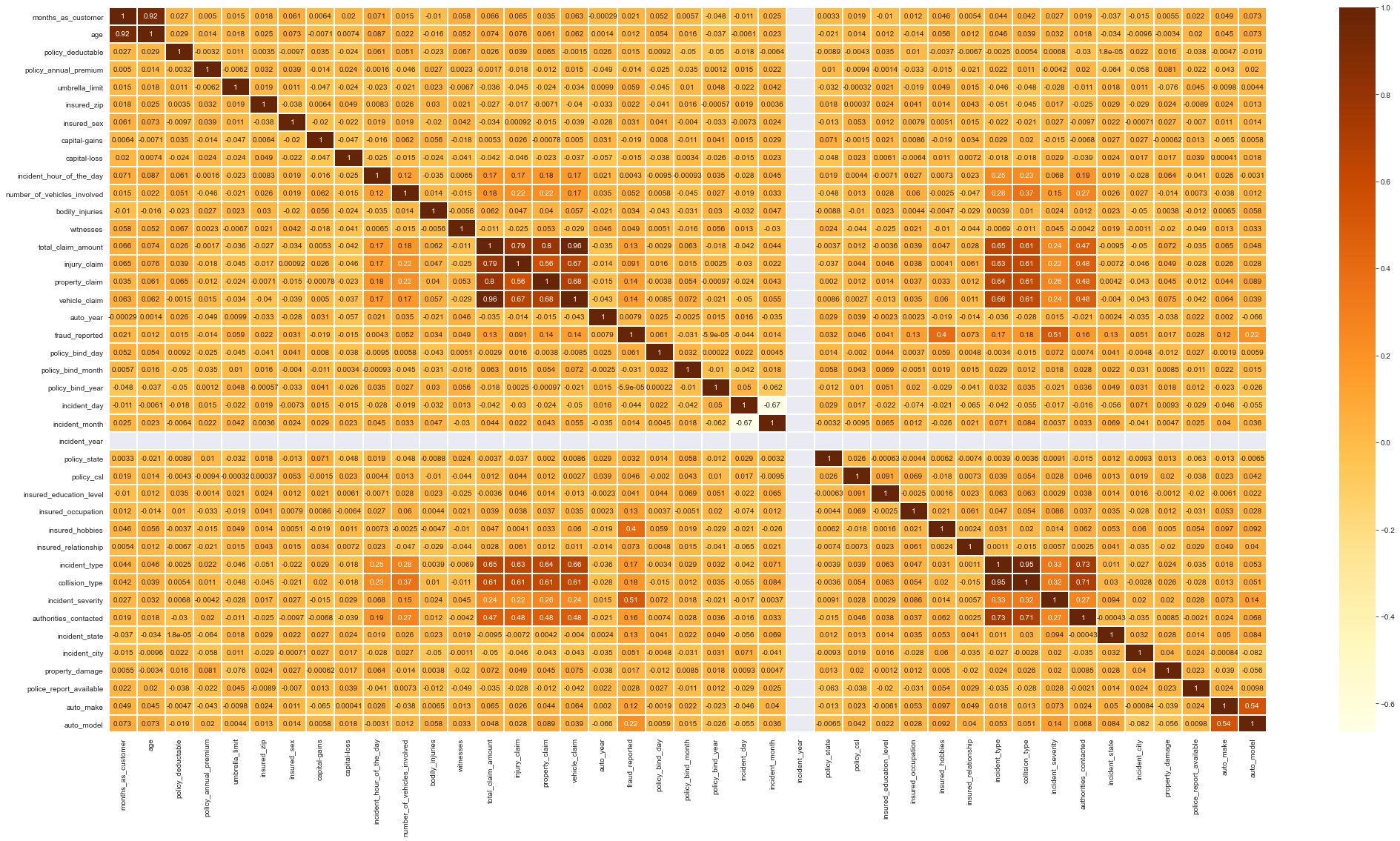


Now we will handle the categorical variable, where we will convert the categorical features into numerical features since the models do not work on object type datatype, however, we have a few models which do accept categorical variables but even they have the algorithm to convert the categorical features to numerical inbuilt. There are several ways to convert the categorical features to numerical and this process of conversion is known as ‘Encoding Categorical Data’.

For this problem statement since I have 17 categorical variables so, will be binary encode the gender column, and since other categorical variables have multiple classes and if we dummy encode it will result in a ‘curse of dimensionality so to prevent it I will target encode whare we replace the classes with the mean concerning target variables.



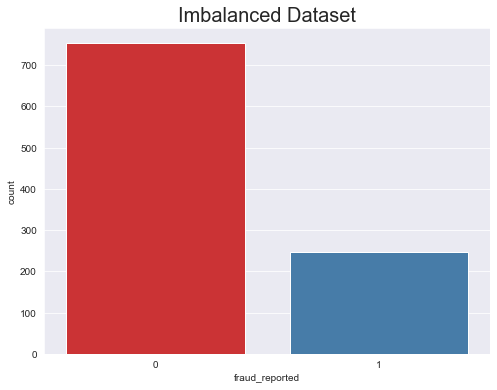
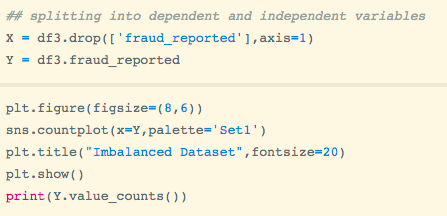
Now since we have created some new features while encoding let’s check the correlation to identify if there is multicollinearity between the variables, and will drop those independent variables which have weaker strength with the target variables.



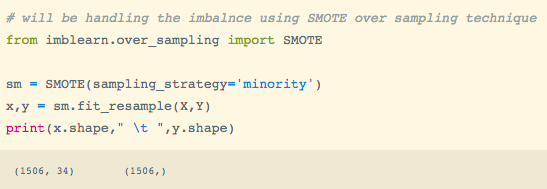
From the above plots, we can see some of the independent variables are highly correlated. So will be removing the variables with the weakest strength with the target variable. Also, incident\_year is not correlated with any feature so will be removing it as well.

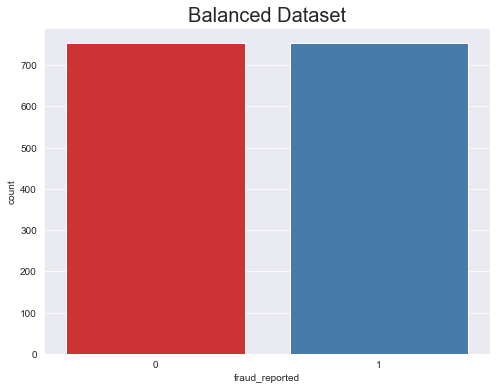
While exploring the Target variable we found that the data is imbalanced due to which there can be misleading predictions because the model will get biased towards the majority class. So will be handling the imbalance in the dataset by applying an oversampling technique SMOTE (Synthetic Minority Oversampling Technique), which creates synthetic data points of the minority class to match the distribution of the majority class.

Before Oversampling the dataset -

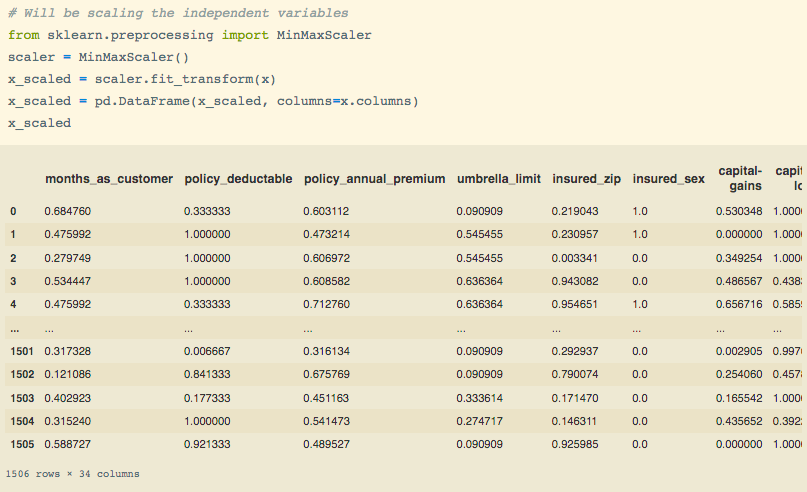


Applying oversampling technique SMOTE:-





Now our dataset is balanced, so we will move further to normalizing the independent features using the MinMaxScalar method from the sci-kit learn library.



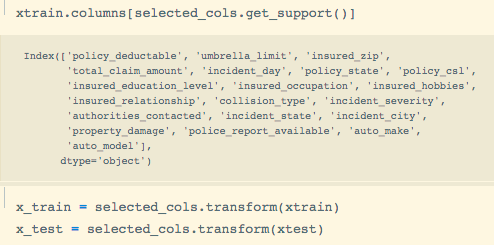
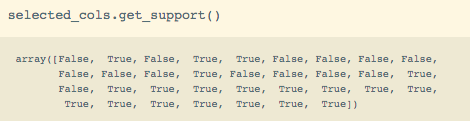
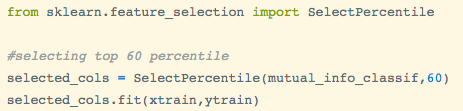
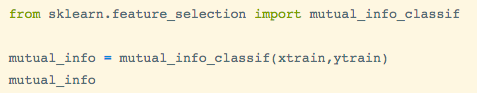
All the values have been normalized and brought down in the range of 0 -1.

# Feature Selection

In this process as suggested by the name, we analyze the feature importance for the machine learning model to take into account the features that have valuable importance or are relevant for prediction.

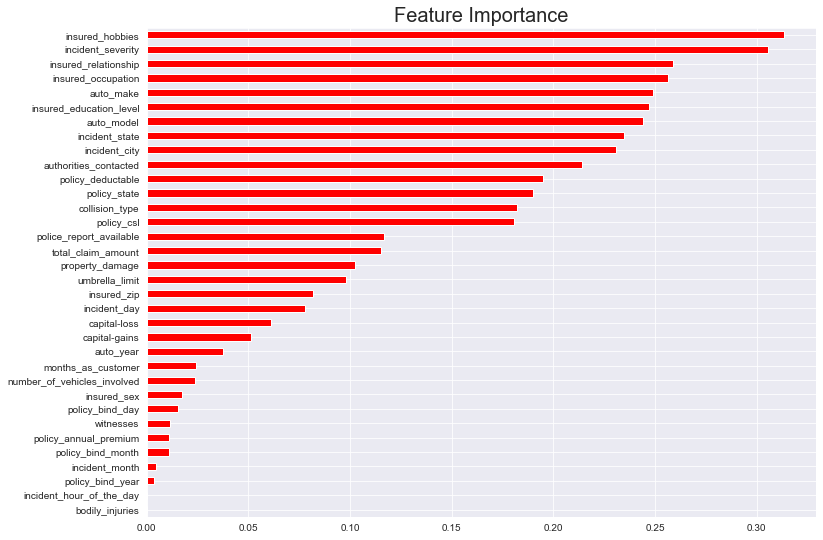
Here I will be using the mutual\_info\_classif method from sklearn.feature\_selection class to get the best features. Mutual \_info\_classif works on the principle of information gain or we can say it's similar to information gain.

And I will also plot a visual bar plot to analyze the feature importance and then will select the top 60 percentile features using the select percentile method.



As we can we have selected the top best features using the percentile method.

Let us analyze feature importance -



We can visualize the most important feature, as we can see that insure\_hobbies is the feature with the highest importance and bodily\_injured is least important, so we selected the best out of these features.

# Model Building and Evaluation

We will now proceed to the main step of our machine learning where we will fit the model and will get predictions as output.

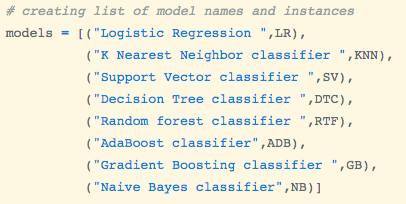
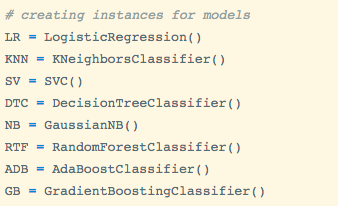
Also, we will evaluate the model on different evaluation metrics -

* Accuracy score
* Confusion matrix
* Precision
* Recall
* F1 score
* Auc-Roc curve

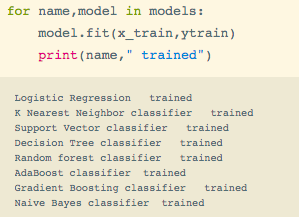
After this, we will validate the performance of the model using the cross-validation technique based on accuracy score to get unbiased evaluation results.

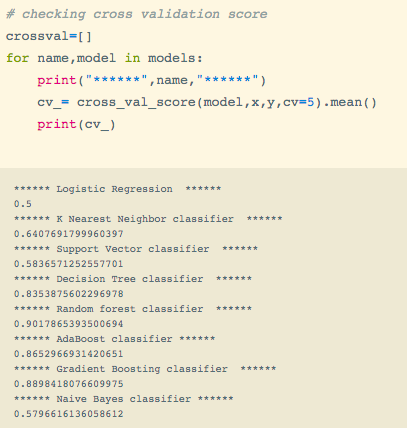
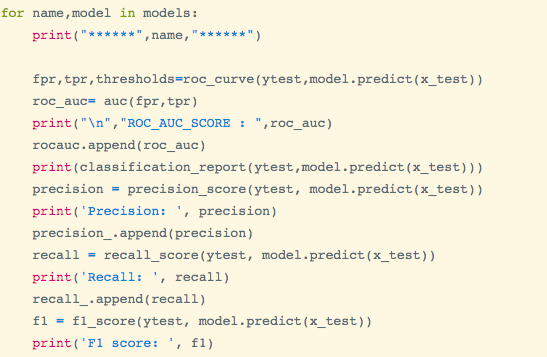
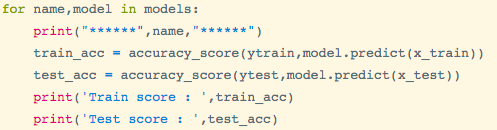
Here I will create instances of multiple models and will fit the models using a loop so that I can get the evaluation report of all different classifiers and further I will pick the model which will be performing best out of these and will further cross-validate that model.

Instances of the model -

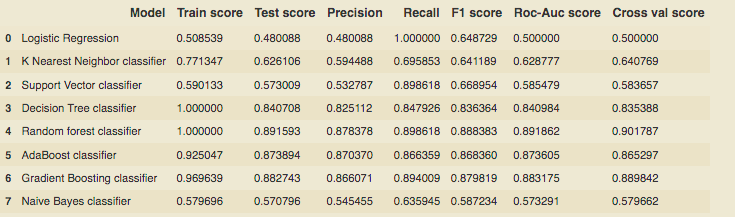


Further, I will train and evaluate these models using for loop -





Now since we have evaluated and cross-validated let’s create these scores in a data frame for a clear interpretation



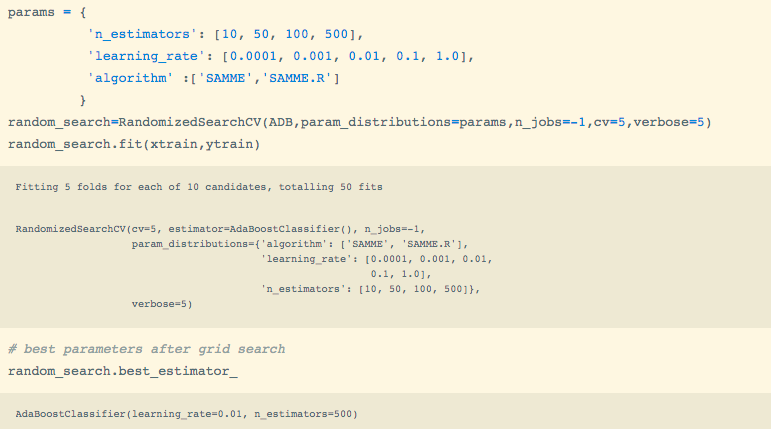
By checking the evaluation scores I will be selecting the Adaboost classifier as the best model because it has good train and test scores and also a good cross Val score of 0.865 and also AdaBoost classifier is the most generalized among all the models.

Now let’s move to hyperparameter tuning to find the best hyperparameters of our model.

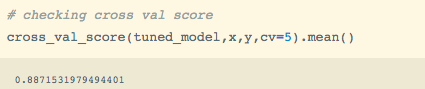
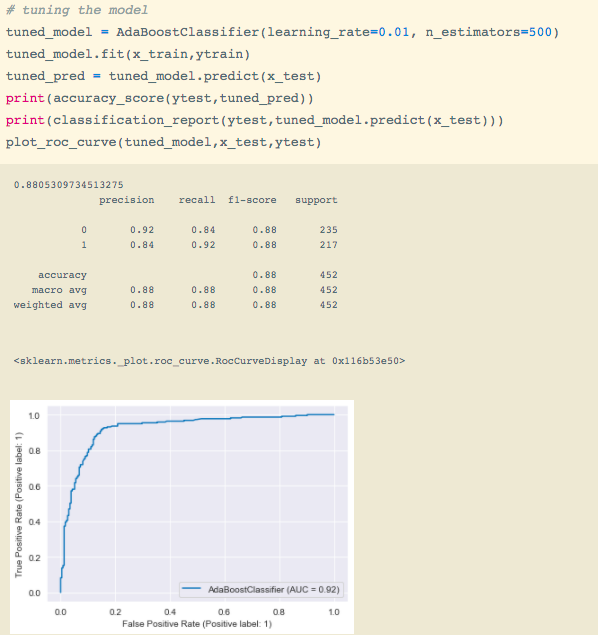
# Hyperparameter Tuning

Hyperparameter tuning is a problem of choosing optimal parameters for a machine learning algorithm. The idea of this is to find out the best parameters for the model so that it can minimize the error and produce good results.

I will be using Random search for my model just because it is a little bit faster than grid search.



Now I will plug in the values we got by checking the best estimators and will evaluate our tuned model and will finalize that model.



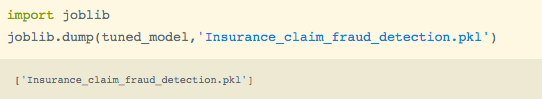
From the above report, we can conclude:-

* The overall performance has increased of the model
* Accuracy and cross Val scores have increased and
* Auc score is 92% which implies our model is now 92% correctly identifying different classes.

Since we have designed our model and tuned it now I will save it for further deployment and prediction.

# Saving the model

This is the very last stage of our project. Here I will be saving the model by serializing it using the joblib package. Joblib is a utility package for saving and loading Python objects.



We can see above that I have saved the final model using joblib.

# 

# Conclusion

This marks the end of our project. We moved step by step through every process starting from EDA to model building and achieved a good model with good accuracy of 88%.