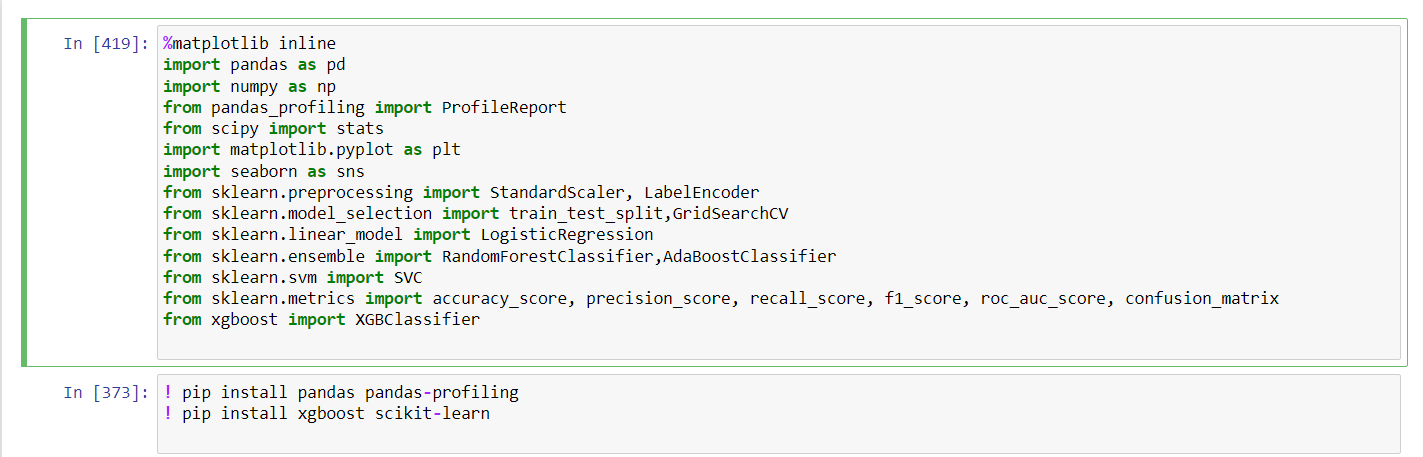
# **Fraud Claim Detection**

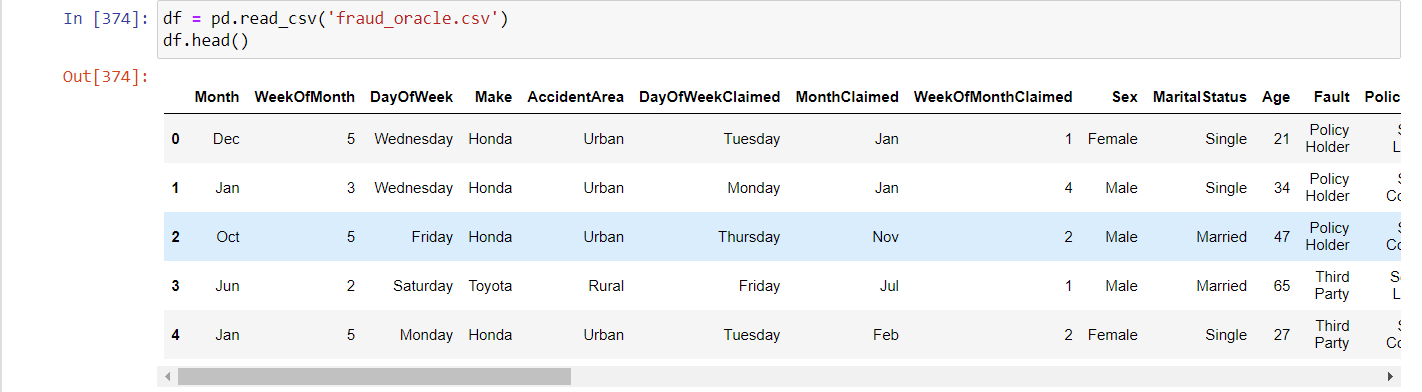
**We are given a dataset in which we have to find if the insurance claim applied for the accident is fraud or not. We’ll pre-process the data and do the Exploratory Data Analysis(EDA), find the insights of the data & plot them and then apply the classification models and find the best model applicable according to the f1 score.**

## **Importing libraries:**

****

* **Importing the necessary libraries needed and some installations which were required for the usage of the libraries.**

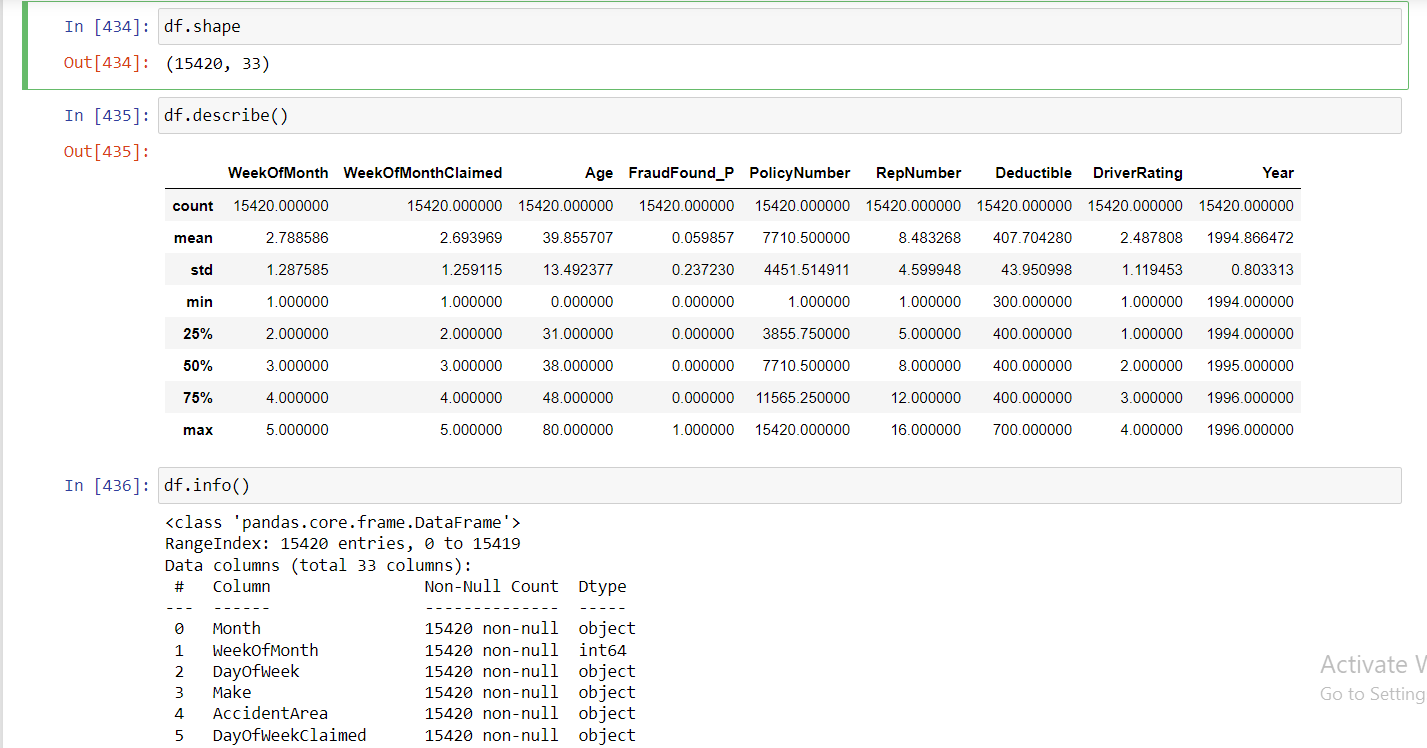
## **Loading the dataset:**



* **Stored the dataset in variable name df (doing the same thing about the 1000th time :) ), and displayed the top 5 rows with help of .head()**

## **Exploratory Data Analysis(EDA):**

**One of the most important part of the project.**

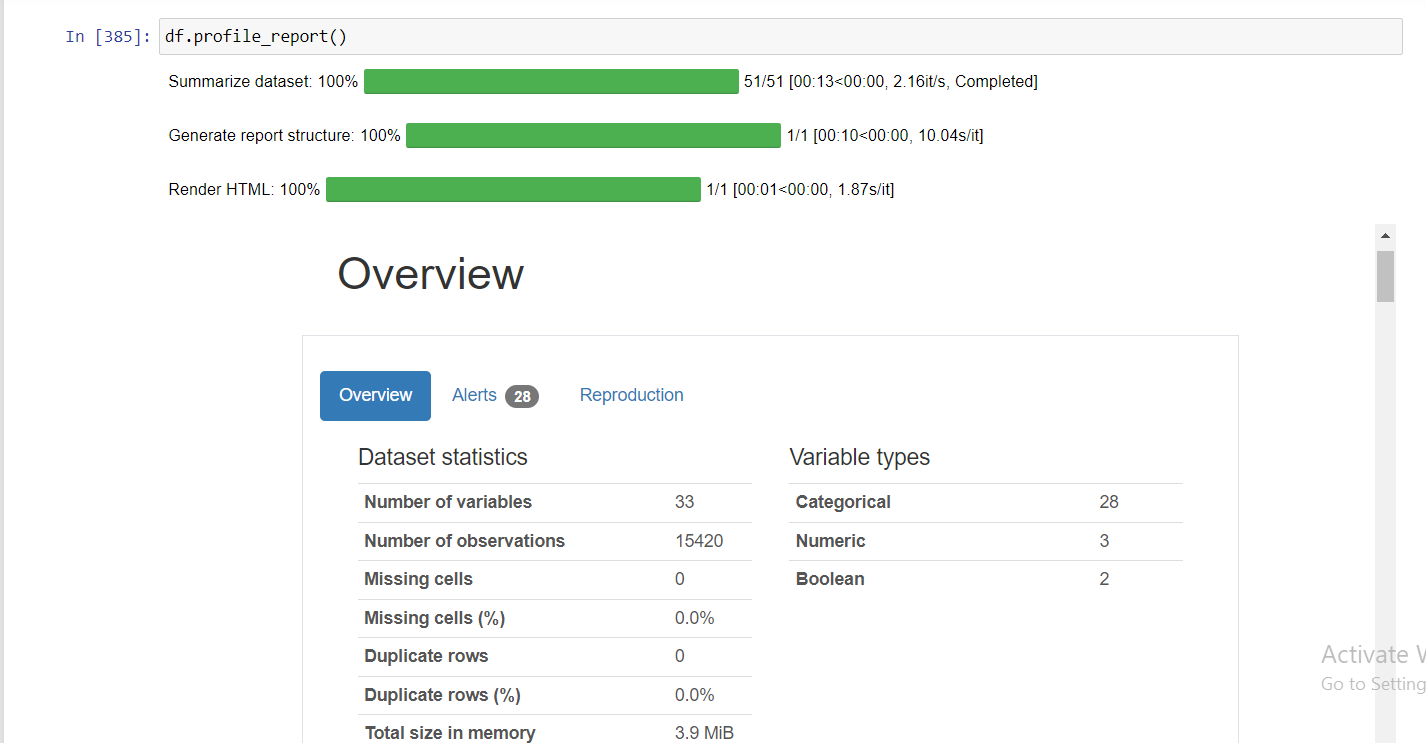
****

**First did the basic summarisation of the dataset.**

* **Got the shape of the dataset**
* **Got the stats of the numerical columns.**
* **And the dtype of the columns.**

****

**Then to know more about the project, got all the column names and the unique values of some certain columns.**

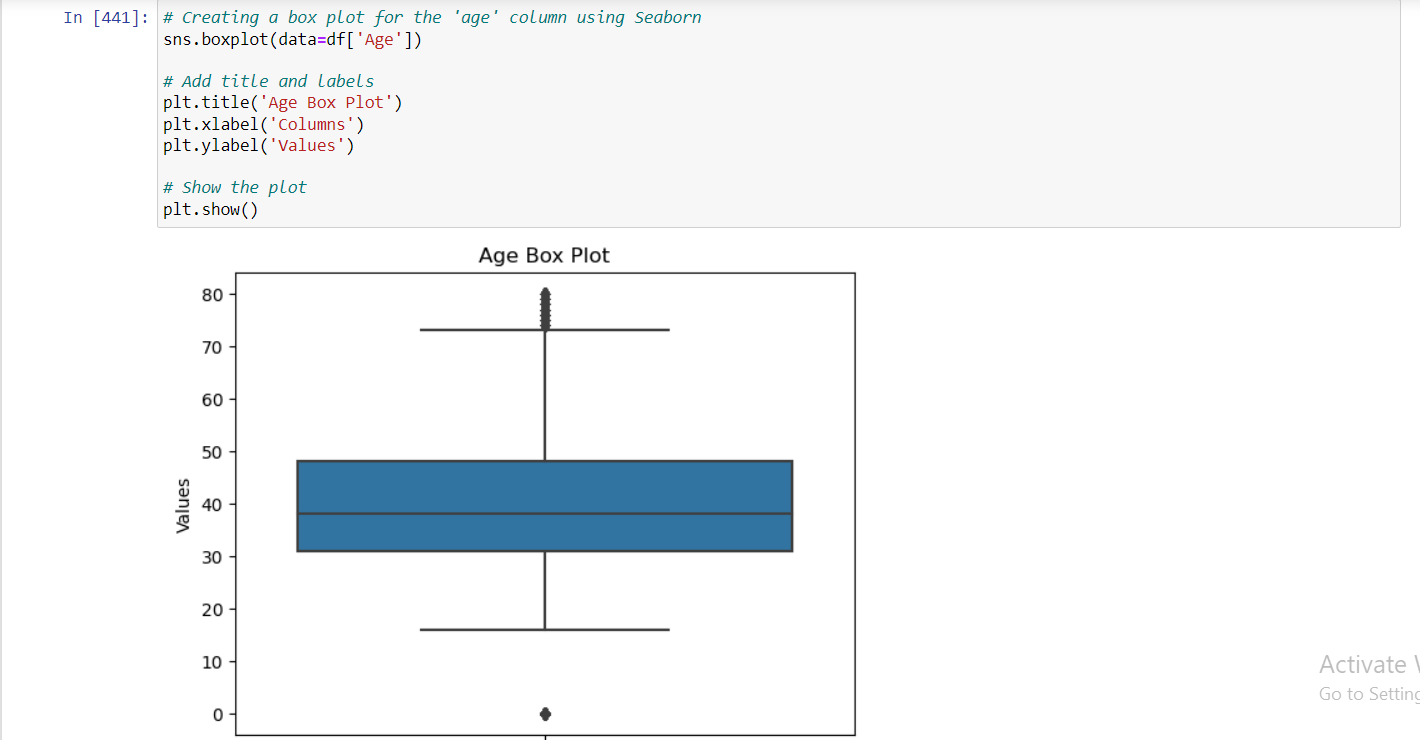
****

**Was going through ‘kaggle’ during our unsupervised learning project and found this profile report function which is really good for EDA. I studied it and got some nice insights and information.**

**Some basic information I got from here is that the dataset is highly imbalanced.**

**Now it's time to plot graphs and try to figure out the co-relations and the information.**

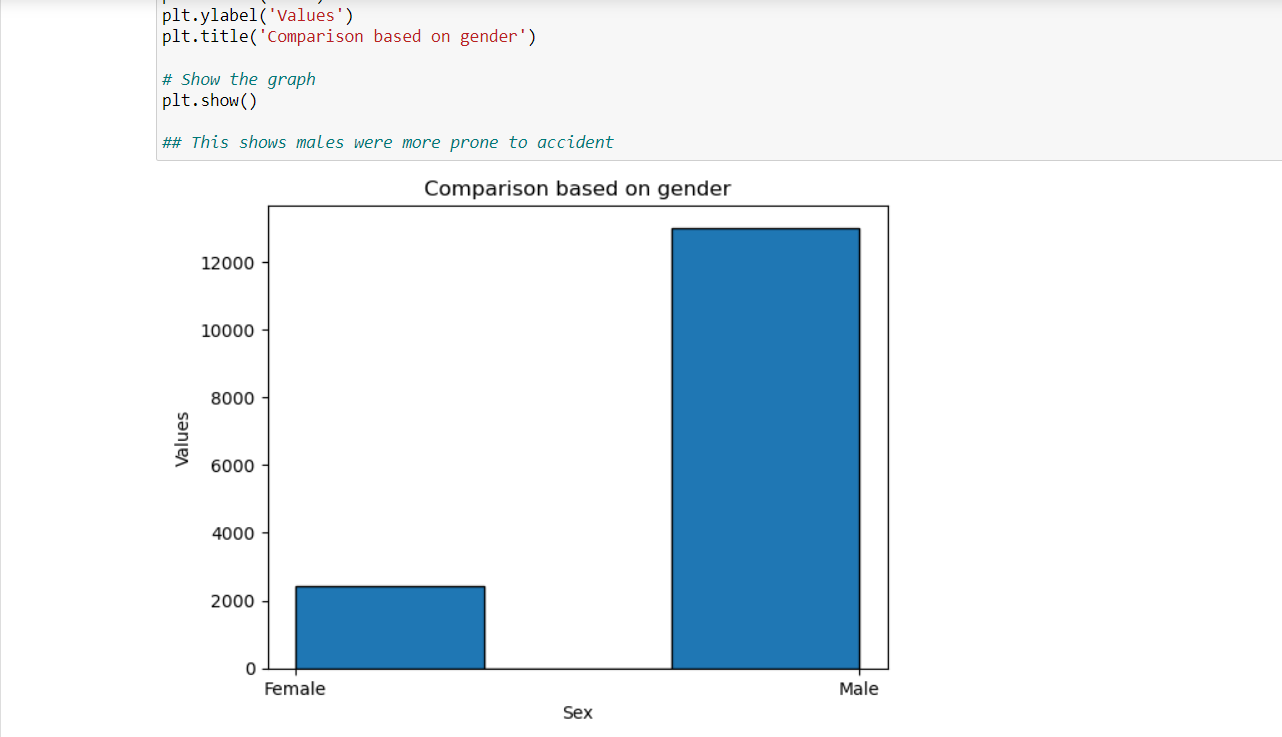
**1.Box Plot for the ‘Age’ column**

****

**Through this plot we clearly got the information that age=0 is an outlier, and we need to remove it.**

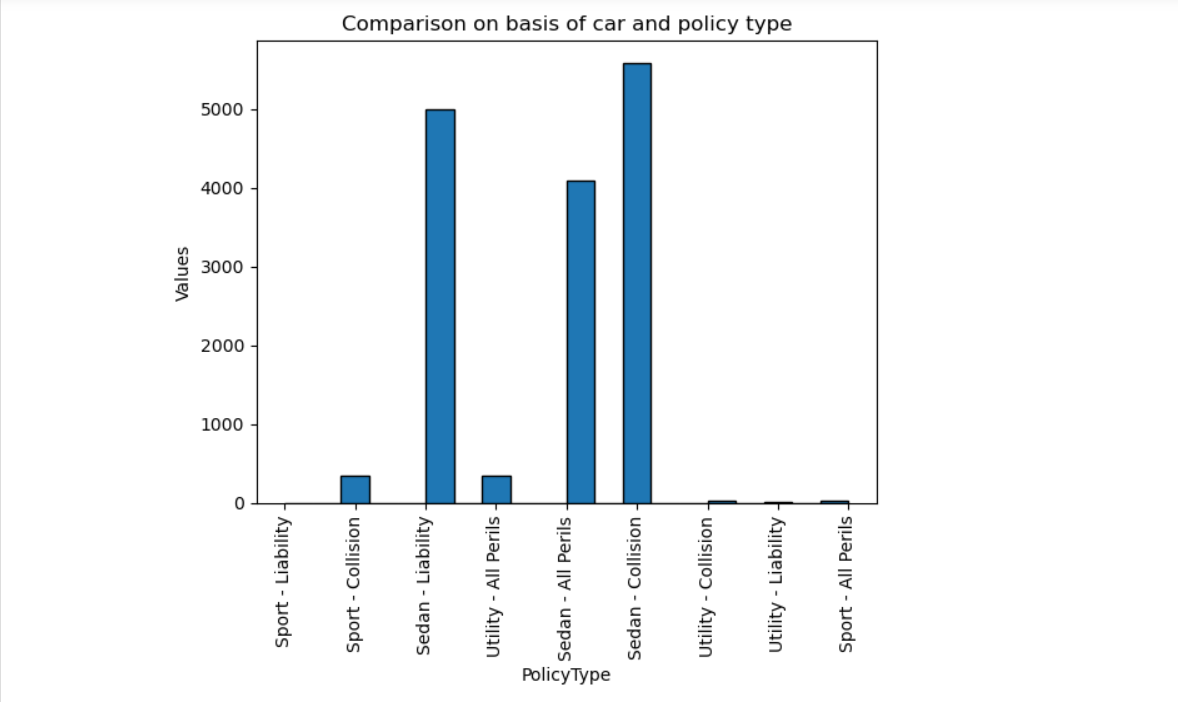
**So, as a part of pre-processing, I removed the rows with age=0, because it is practically impossible, so that information is useless.**

**2. Bar graph for comparison based on gender**

****

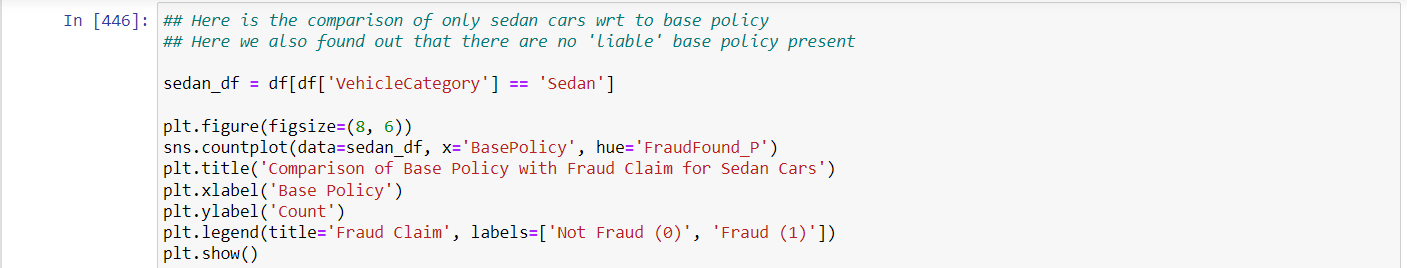
**This clearly showed that the males were dominant in the accidents.**

**3. Bar graph for comparison based on car and policy type**

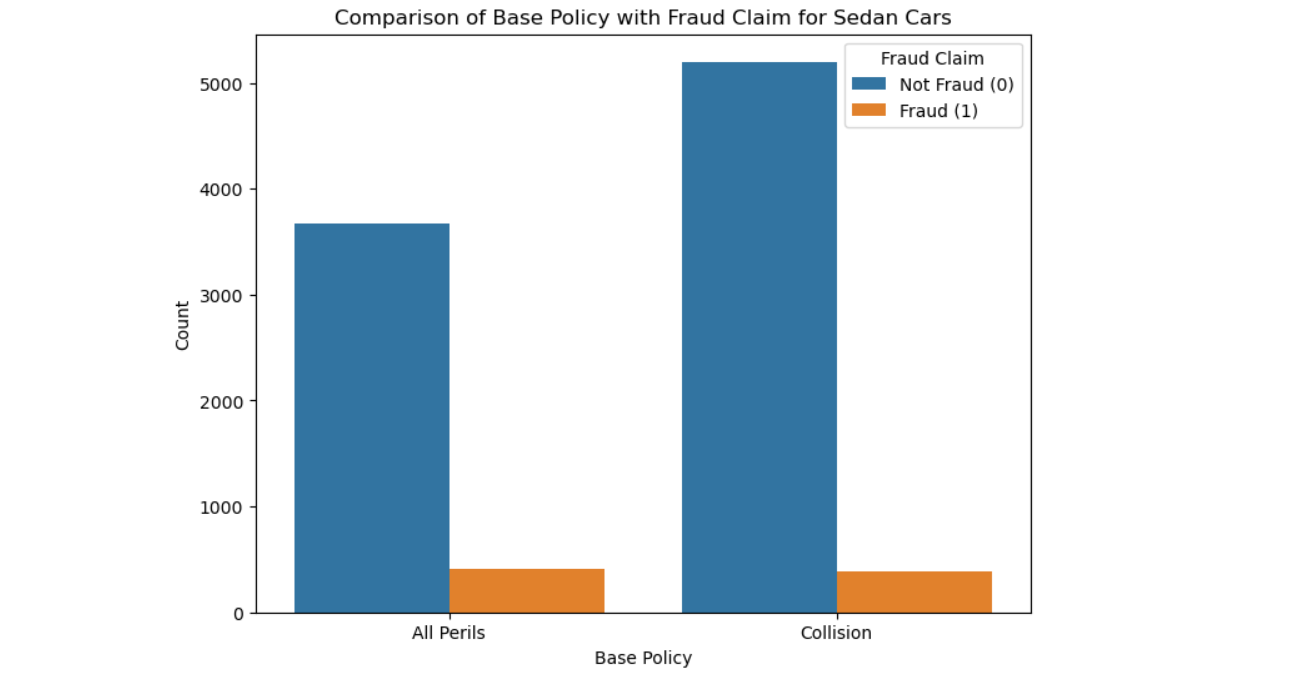
****

**This clearly showed that the sedans were the cars most included in the accidents.**

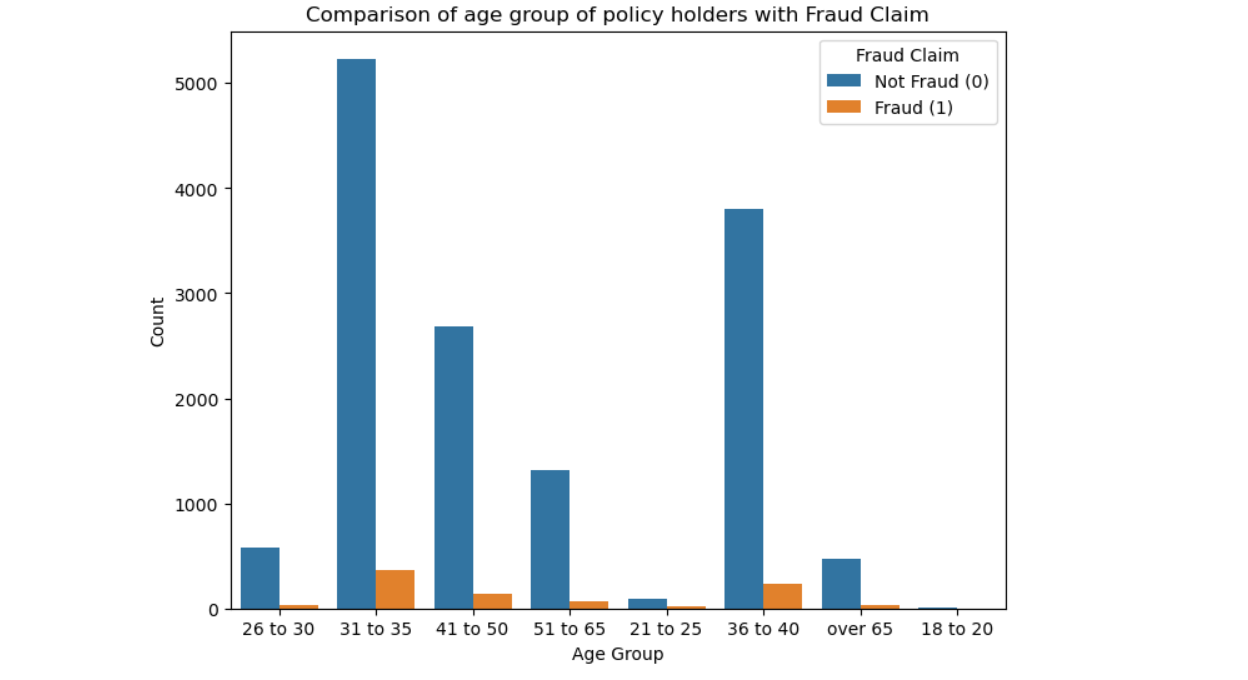
**Here we created a separate dataset only containing sedan car type, so we can do a bit analysis on this data.**

****

**4. This a base policy comparison with fraud claims only for sedan cars**

****

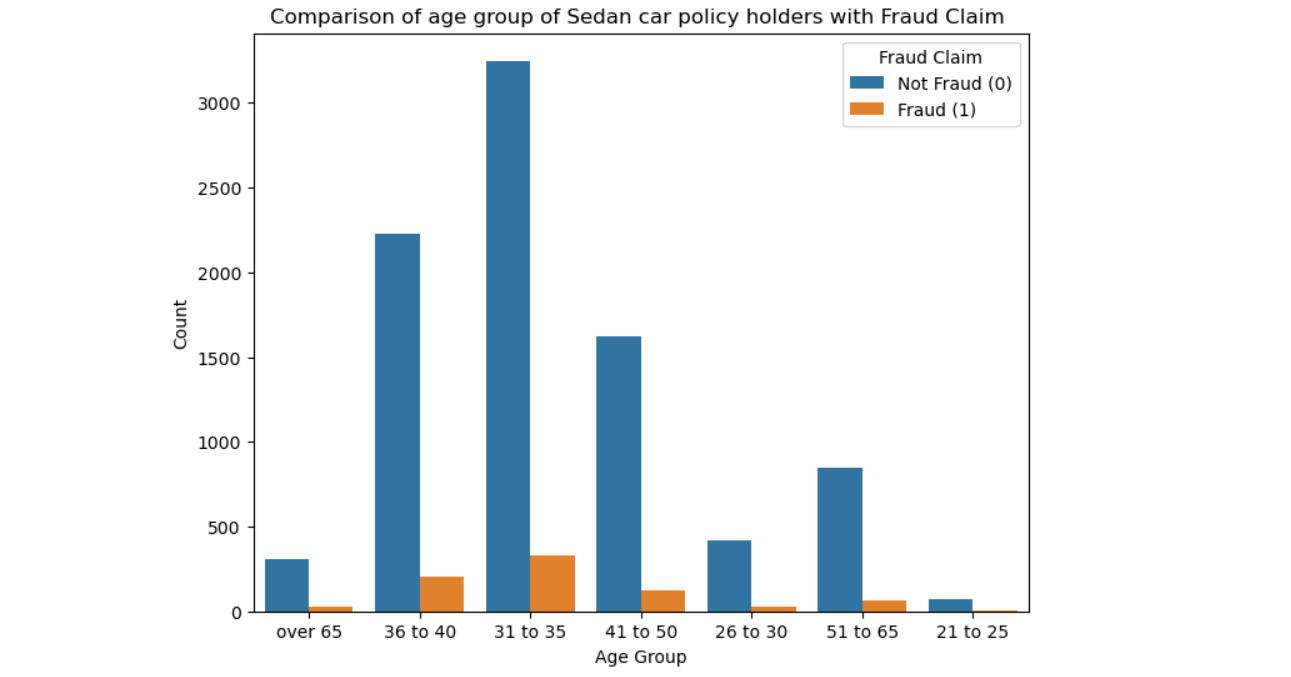
**It showed that the collision policies were more for sedans and the liability policy were not there for any sedan.**

**5. Age group comparison**

**Most claims and fraud claims are done by the age group ‘31-40’.**

**Then the age group ‘41-50’.**

**6.Age group comparison for sedan cars only**

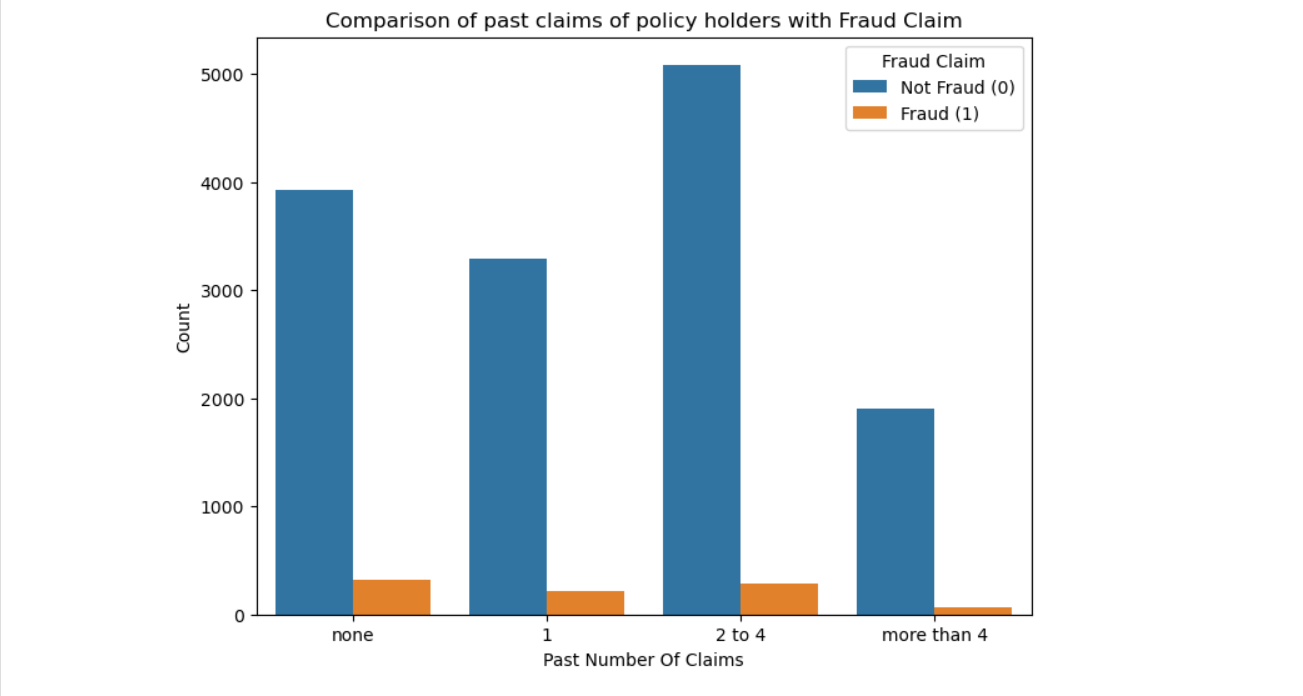
****

**Similar relations can be seen here as the previous one.**

**“ So now if I combine the results from previous plots we can say till now that most insurance claims and most fraud insurance claims are done by the Sedan car holders of age group '31-35' and '36-40' ”**

**Now if we move further for some more exploration-**

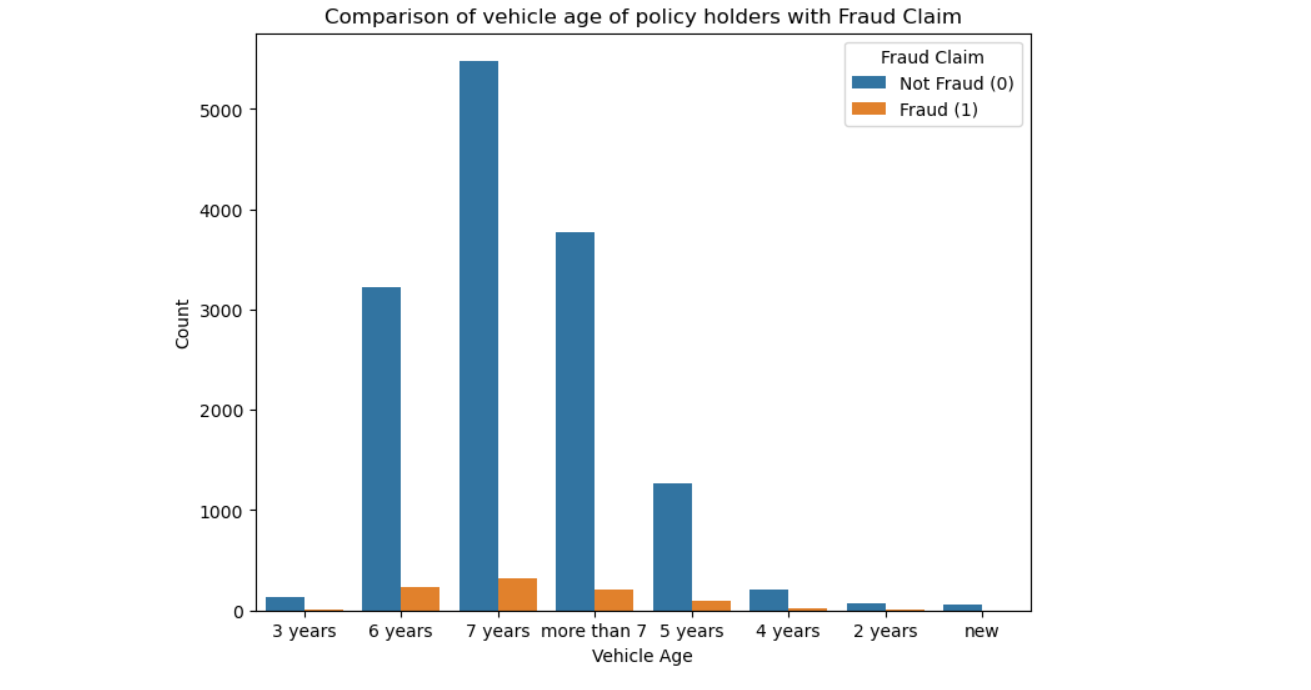
**7. Comparison to see how the past claims of policy holders are related to frauds**

****

**Not a particular dominating group, many fraudsters showed no previous claims.**

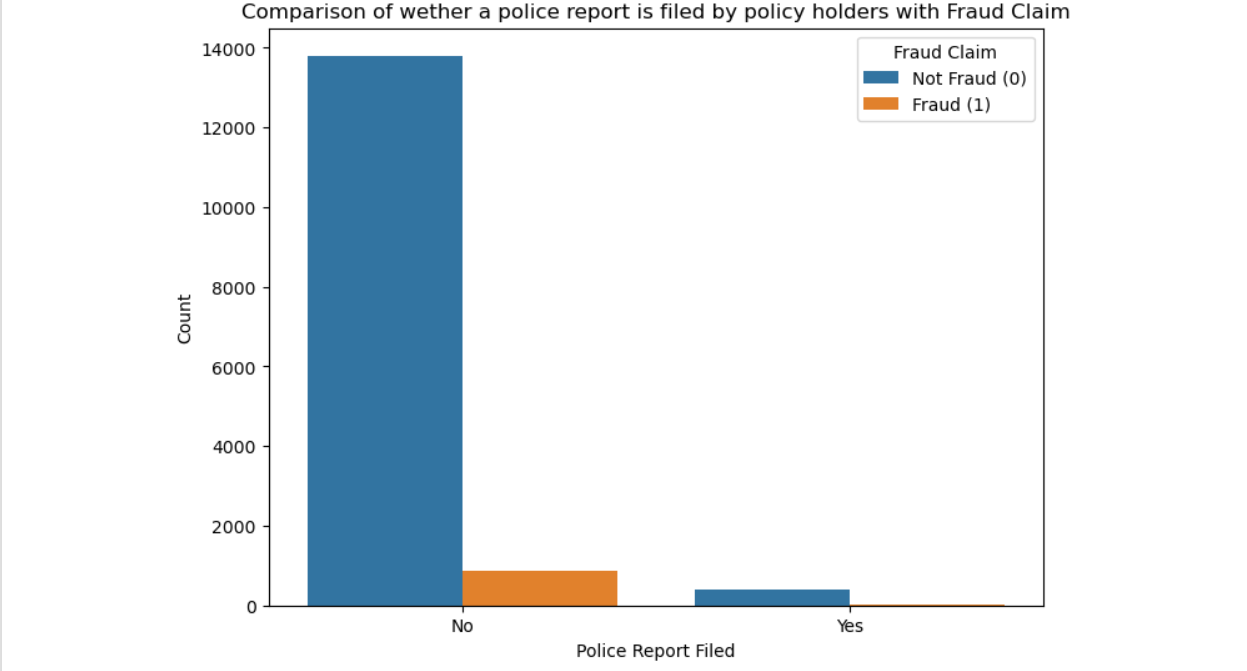
**But from here we can see that most claims come from the people who have already done the past claims and no. of fraud claims is high too.**

**8. Comparison of vehicle age with claims**

****

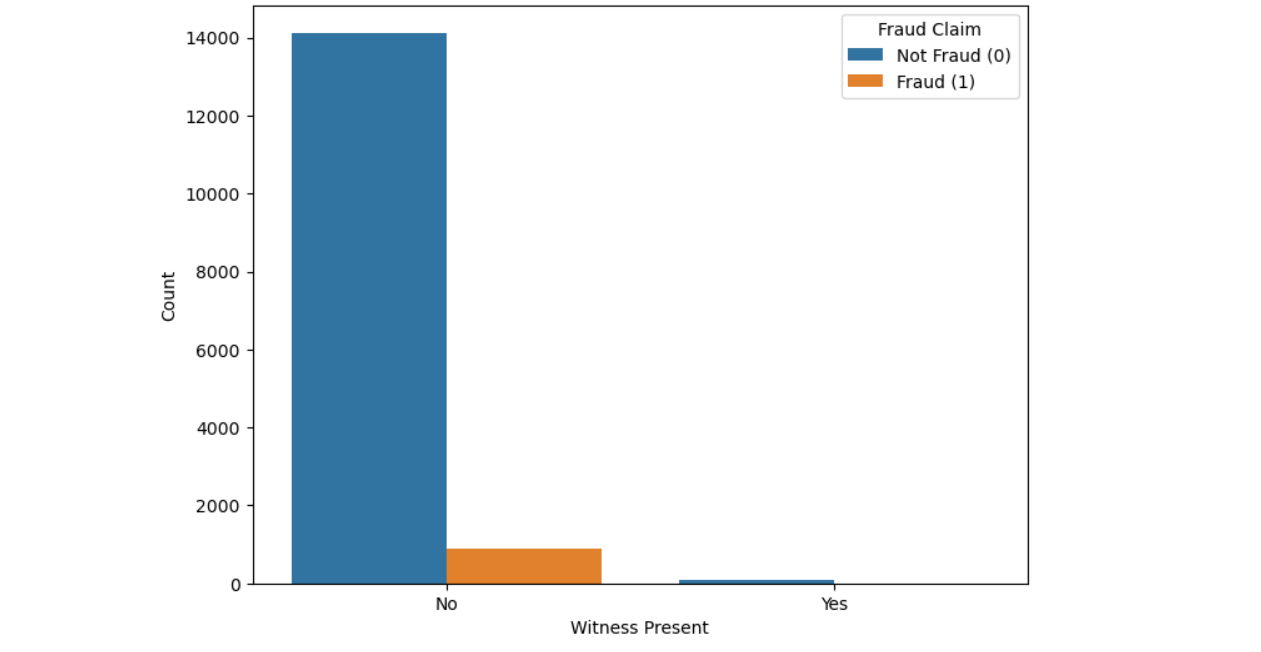
**Clearly older vehicles are more prone for accidents and claims, and fraud claims follow the same trend.**

**9. If police report is filed or not**

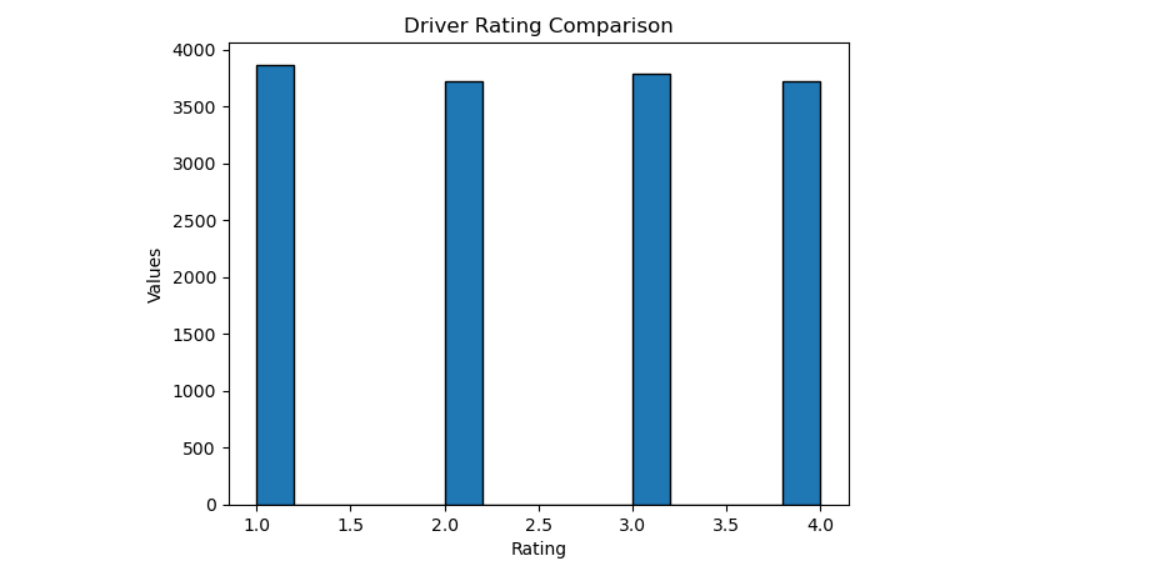
****

**People are not filling in the police reports after the accidents and obviously fraud claims will follow the same trend.**

**10. Whether the witness was present or not**

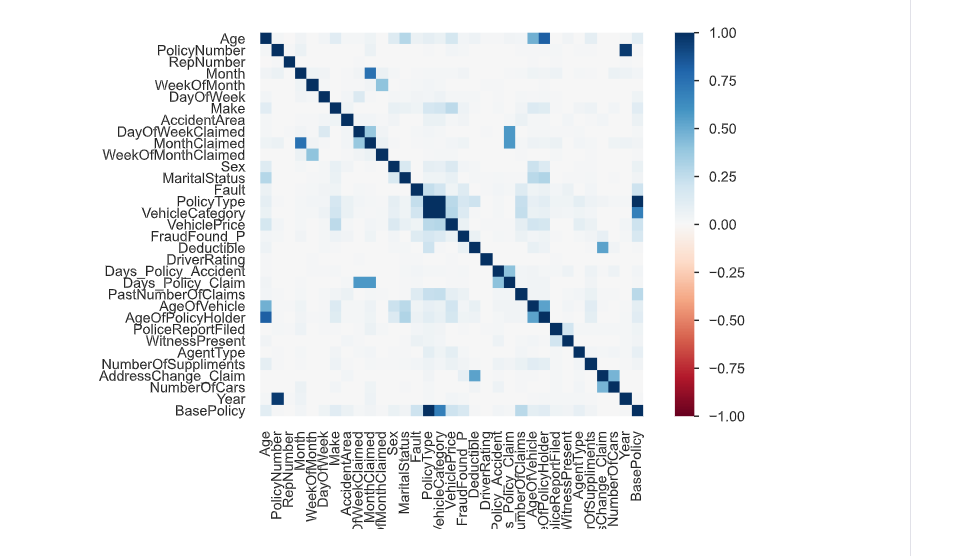
****

**So no witnesses were present for most of the accidents.**

**11. If driver rating affects claims**

**Clearly shows that driver rating has no effect on accidents and there claims.**

**12. Heatmap**

****

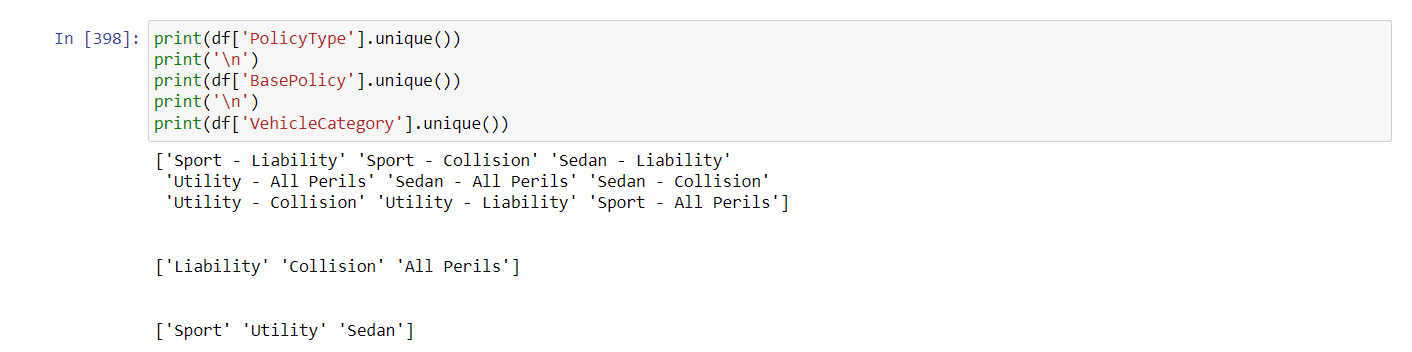
**Really good correlation matrix and clearly see what all aspects are closely related.**

* **Age group is related to age of policy holder (which is obvious)**
* **Month accident happened related to month of claim, these are related because either the claim happened in same or 1-2 month after the accident**
* **PolicyType realted to BasePolicy because a part of information is same in both**
* **Year and policy type are randomly related**

## **Pre-processing**

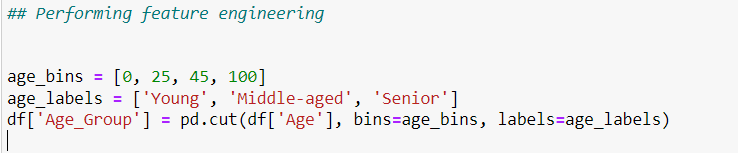
**There weren't many extensive things to pre-process as there were no null values, no duplicate rows which I checked in the starting of the code.**

****

****

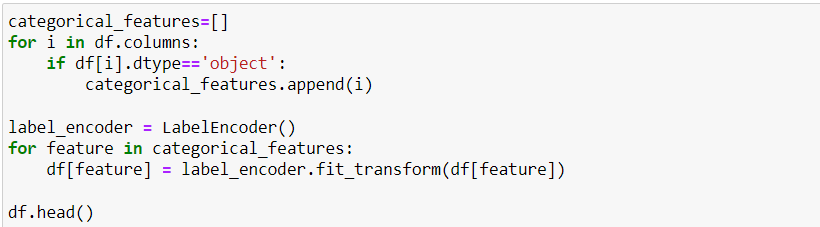
**Now above we can see that these 3 columns contain some common information, so we’ll remove the policy type column, as we’ll need to label encode this data afterwards so it will be better if label encode the below 2 columns separately.**

**Feature Engineering:**

****

**Didn’t found many features to make, so here is one general feature to see if person coming for claim belongs to which age category.**

**Now it's time to label encode the categorical features, which is required in order to use them in our models-**

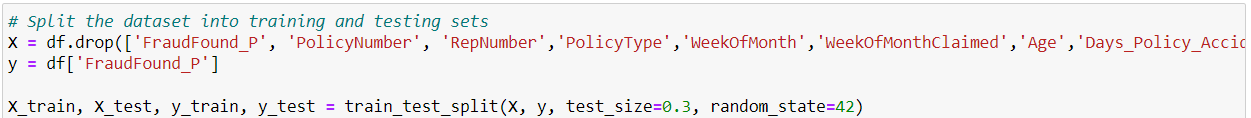
****

**Here I made a list of all categorical features, and then label encoded them easily.**

**I didn’t use one hot encoding here because there are already many columns and if I used it then then for every categorical column there would have been more than 3 columns and our dataset will become a chaos.**

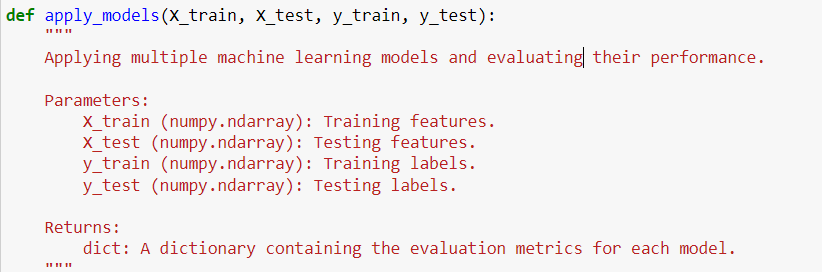
## **Model Application**

**Splitting the dataset-**

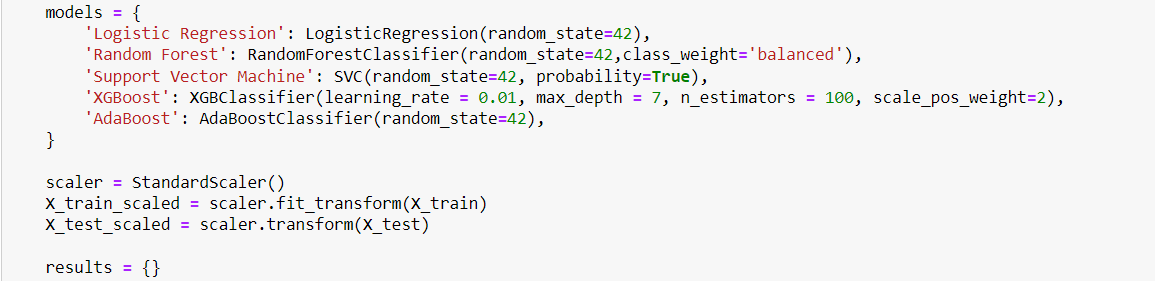
****

**I excluded many features from the input variable, because they weren’t related to the target variable.**

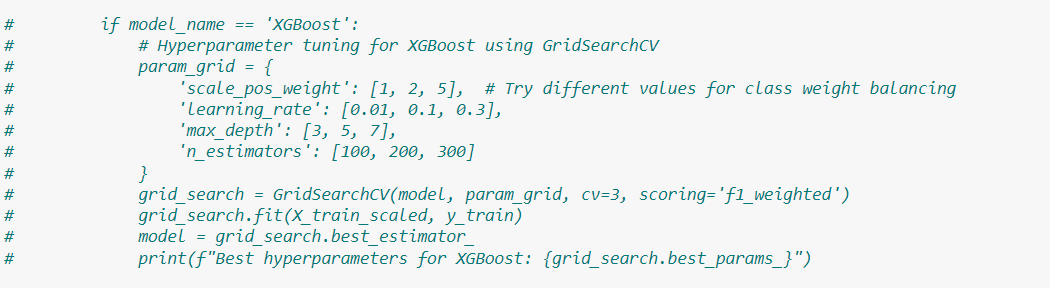
**I chose the test size to be 30% of the state.**

****

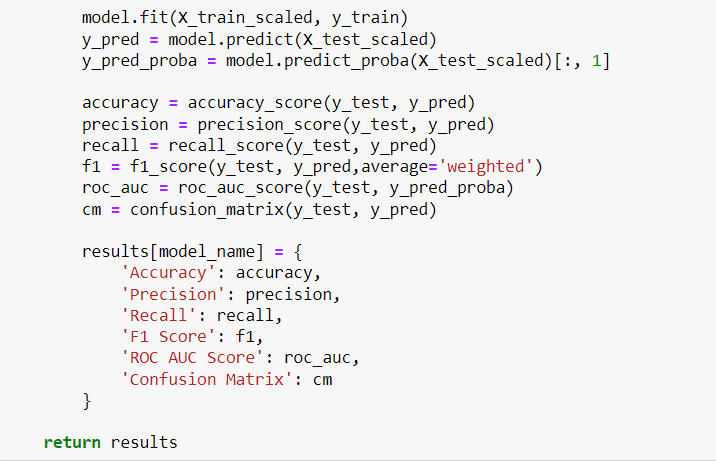
**So, I created a function which takes the input of our required variables and returns the evaluation metrics. This allows us to easily apply and check all the models, and choose which one is best for us.**

****

* **There I mentioned the dictionary containing all the models I want to apply.**
* **Then I scaled the data here only (it is a pre=processing step, could have been done before if wanted).**
* **Then created an empty dictionary for results.**

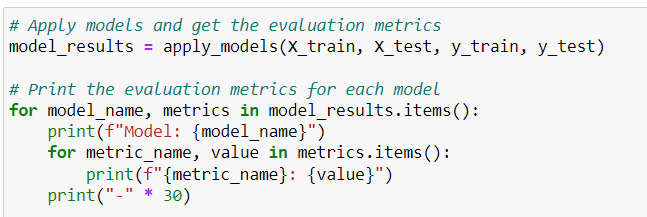
****

**Performed the hyperparameter training for XGBoost, and found the best parameters possible. I used those parameters and then put this to read only text form to decrease the running time.**

****

**Now, finally we fit the scaled values to models one by one using a for loop and predict the target variable and calculate all the required evaluation metrics.**

**[Used average= ‘weighted’ in f1 score because the dataset is highly imbalanced and weighted f1 score used class wise sample calculations to get improved f1 scores]**

****

**At the end applying the model and printing the results.**

**So, now if we compare we have the best f1 score for XGBoost, though there is not very much difference from the other models.**

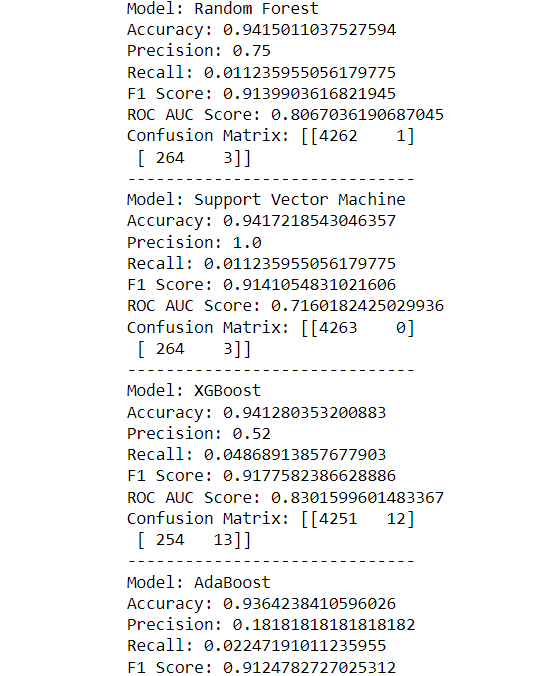
**This is all I did, definitely much better can be done, I’ll definitely try and better my performance.**

**For models, I first applied ‘Logistic Regression’, a simple and interpretable model that models the probability of the target class using a logistic function, but didn't get the desired results as precision and recalls were 0.**

**Then came the ‘Random Forest’ and ‘SVM’ , RF which combines multiple decision trees to make the predictions and ‘SVM’ a model which finds a linear separator or hyperplane according to the requirements of separating the features by maximising the margins. These models gave quite good f1 scores.**

**Then I used boosting models that build multiple weak learners (e.g., decision trees) in sequence to improve predictive accuracy.**

**XGBoost, an optimised and efficient implementation of gradient boosting gave the best results.**

****