**Insurance Claims- Fraud Detection**

**Problem Definition:**

Insurance fraud is a very huge problem in the industry and it is very difficult to identify fraud claims or cases. It is an incorrect or misrepresented or false claim by an insured person for financial gain. Insurance fraud can be committed at different levels.

According to a study, estimates approximately of $80 billion in fraudulent claims are made annually in the United States. This includes all lines of insurance. Healthcare fraud alone is estimated to cost Americans $54 billion a year.

The insurance fraud that occurs more frequently, which are staged to claim the insurance money are:

* Auto/motor accidents
* Health insurance
* Theft or burglary
* Motor or car thefts
* Staged home fires

To reduce these fraud claims we need to find whether the insurance claim made is a genuine or a fraudulent one. Machine learning plays a major role in doing so.

This article is basically on ‘Insurance claim-Fraud detection’ that takes you to a step-by-step process to understand the whole Machine learning building process.

**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.

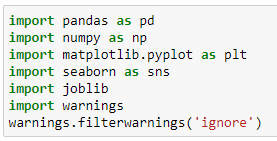
In this project, you are provided a dataset that has the details of the insurance policy along with the customer details. It also has the details of the accident based on which the claims have been made.

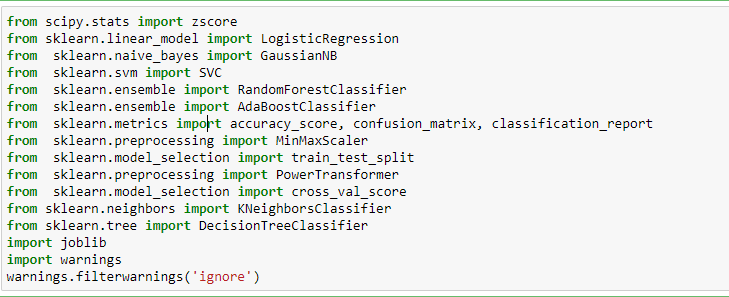
In this example, you will be working with some auto insurance data to demonstrate how you can create a predictive model that predicts if an insurance claim is fraudulent or not.

We are going to pick up a data set of auto insurance and perform analysis and predict if an insurance claim is fraudulent or not using Machine learning.

**Importing the data:**

We need to import all the relevant libraries:





We have now imported all the important libraries that we will be needing during analysis.

We need to import the .csv file into the Jupyter notebook.



This data set contains of Independent and Dependent (target) variables.

Independent variable: They are also known as Input variables. These are the input for a process that is being analyzed.

Dependent variable: They are also known as Output or Target variables. They are dependent on Independent variables for their outcome.

After importing the dataset, display a sample of data. The variables in the dataset are as follows:

* months\_as\_customer,age
* policy\_numbe
* 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
* capita\_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

**Data Analysis (EDA)**

**Now we need to understand the dataset by performing Exploratory Data Analysis.**

Let’s check the shape of the data set:



We can see that there are 1000 rows and 40 columns in the dataset.

We cannot have null values in the data as this will affect the data and eventually the predicted result. Therefore we must check for any null values in the dataset.

Checking for null values in the data set:

ic.isna().sum() *#Checking for null values*

**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**

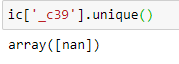
**fraud\_reported 0**

**\_c39 1000**

dtype: int64

we can see that there are 1000 null values in ‘\_ c39’ column and total no of columns is 1000.

Checking for unique values in ‘\_ c39’ column



There is only one unique value, i.e: ‘nan’ which is null values. Therefore we can eliminate this column.



we can even drop 'policy number' Column as each policy number is unique.



we can drop ‘incident\_location’ Column as each incident\_location is unique.



we can drop 'policy\_bind\_date' as it is insignificant



**Missing values treatment:**

After dropping the column now the dataset has 1000 rows and 39 columns.There are few '?' values in 'collision\_type','property\_damage' and 'police\_report\_available' columns. Therefore we need to replace them with null or ‘nan’ values for handling them.

After replacing we can recheck for null values:

ic.isnull().sum()

months\_as\_customer 0

age 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 178

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 360

bodily\_injuries 0

witnesses 0

police\_report\_available 343

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

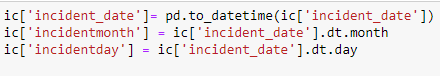
dtype: int64

There are 178, 360 and 343 null values in collision\_type, property\_damage and police\_report\_available respectively.

* For collision\_type we can replace the null values with the mode of that column. The mode of the column for collision\_type is no.
* For property\_damage we can replace the null values with 'NO' because it could be that they weren't any property damage in the first place. Also, the mode of the column is 'NO'
* Similarly, for police\_report\_available we can replace the null values with 'NO' because it could be that they haven't reported. Similarly here also the mode of the column is 'NO'

All the null values have been removed from the data and we are good to go.

'incident\_date' is in object type (date format). We have to convert the date into months and days. There is no need to extract year because the year is same in all the rows, i.e 2015. Extracting of months and days from incident\_date can be done by writing the below code.

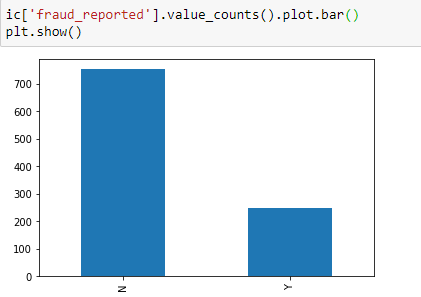


After extracting, we can drop the ‘incident\_date’ column as we have extracted the information in we have all the information in 'incidentmonth' and 'incidentday'.

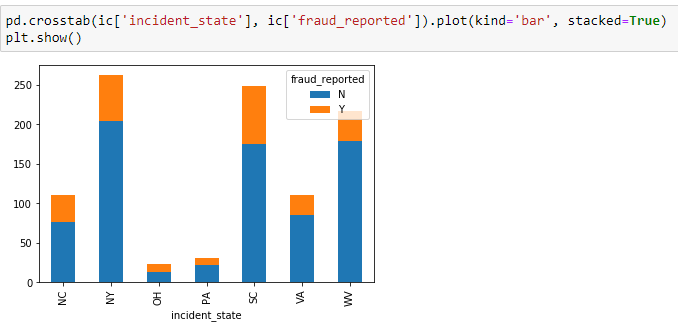
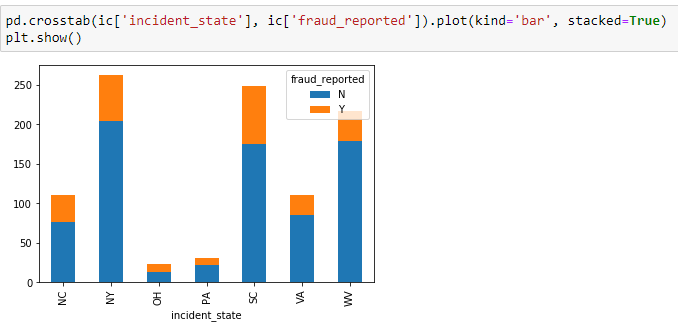
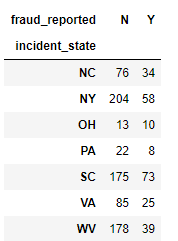
**Data Visualization and EDA Concluding Remarks:**

In the given data, ‘fraud\_reported’ feature is the Target feature or variable. The unique values of this feature are only 2 i.e Y and N (Yes and No), which means it has only two classes. So, as there are only two unique values this is a ‘Classification Problem.’

The dependent or target variable has 753 nonfraudulent cases and 247 fraudulent cases which can be seen below in the form of value counts and bar chart.

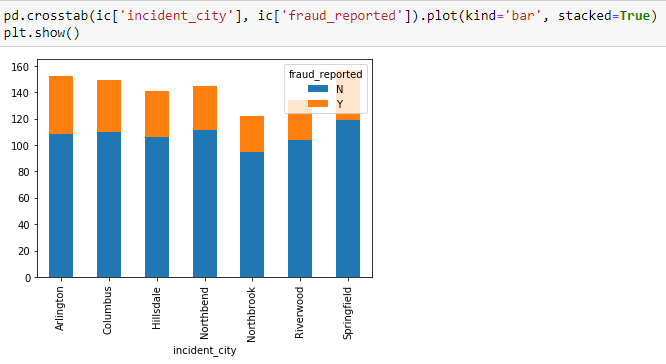


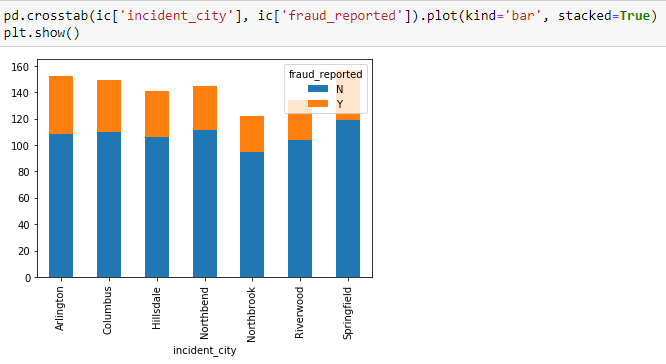
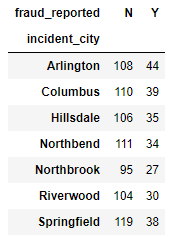
To see which incident\_state has the maximum of fraudulent cases reported we need to barplot incident\_state and fraud\_reported as shown below.

From the above bar plot, we can see that the fraudulent cases are highest in South Carolina (73 cases) followed by New York (58 cases). The cases are least in Ohio state.

To see which incident\_ city has the maximum of fraudulent cases reported we need to barplot incident\_city and fraud\_reported as shown below.

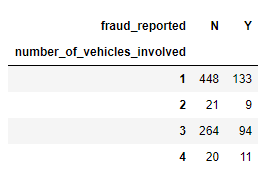


From the above bar plot, we can see that the fraudulent cases are highest in Arlington (44 cases) followed by Columbus (39 cases). The cases are least in Northbrook (27 cases)

To check the impact of number\_of\_vehicles\_involved on fraudulent cases we can barplot between number\_of\_vehicles\_involved and fraud\_reported as shown below.

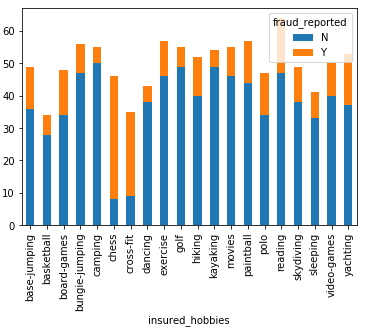


From the above figures, we can see that the fraudulent cases are maximum where no of vehicles involved is 1 (133 cases) followed by where no of vehicles involved is 2 (94 cases), and least when no of vehicles involved is 2 (9 cases) and 4 (11 cases).

To check the impact of insured\_hobbies on fraudulent cases we can barplot between insured\_hobbies and fraud\_reported as shown below.

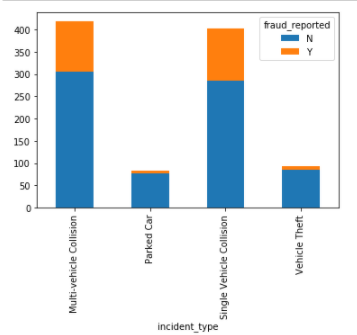
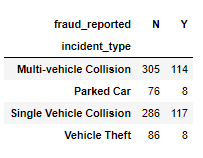


There is a significant relationship between insured\_hobbies and fraud\_reported. We can see from the above figures that people whose hobbies are chess (38cases) and cross fit (26 cases) are most likely to commit fraud.

To check the impact of incident\_type on fraudulent cases we can barplot between incident\_type and fraud\_reported as shown below.

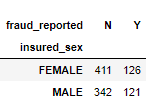


From the above figures, we can see that maximum of cases is fraudulent if it was a multi-vehicle and single-vehicle collision.

To check the impact of insured\_sex on fraudulent cases we can barplot between insured\_sex and fraud\_reported as shown below

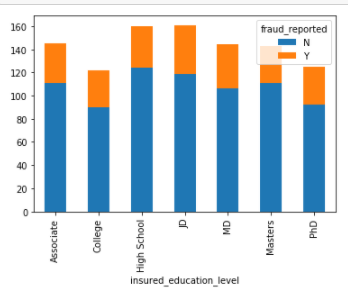
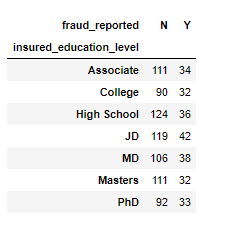


From the above figures, we can see that the fraudulent cases are slightly more if it is a female Compared to a male.

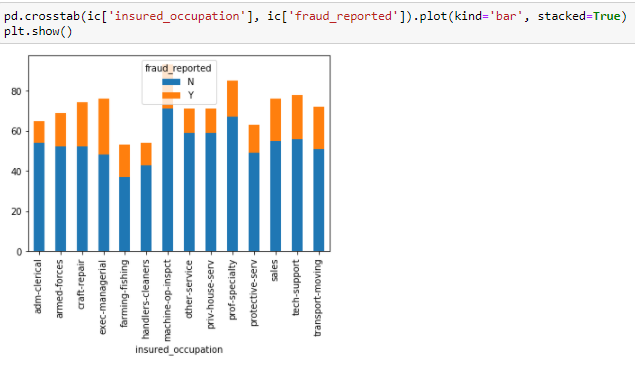
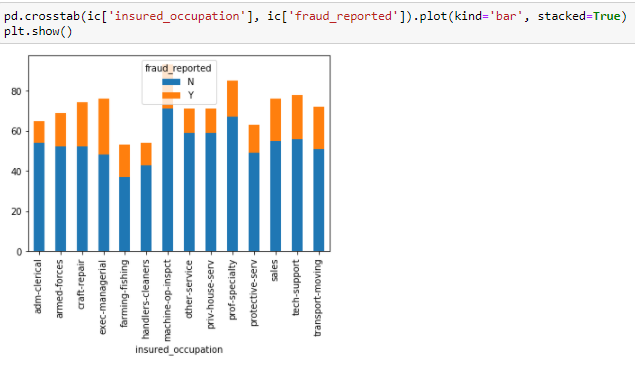
To check the impact of insured\_education\_level on fraudulent cases we can barplot between insured\_education\_level and fraud\_reported as shown below



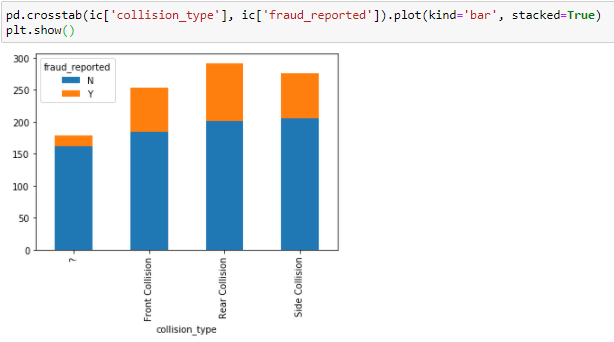
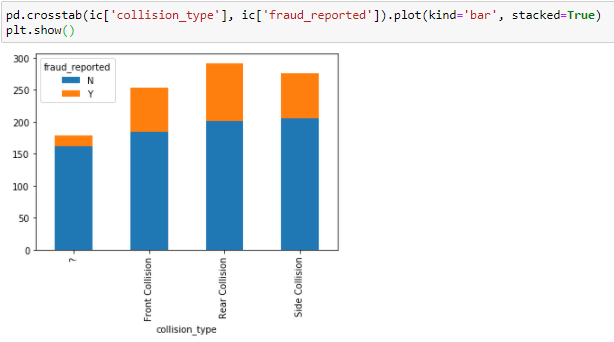
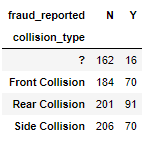
From the above figures, we can see that the no of fraudulent cases is maximum where the person has done JD.

To check the impact of insured\_occupation on fraudulent cases we can barplot between insured\_occupation and fraud\_reported as shown below

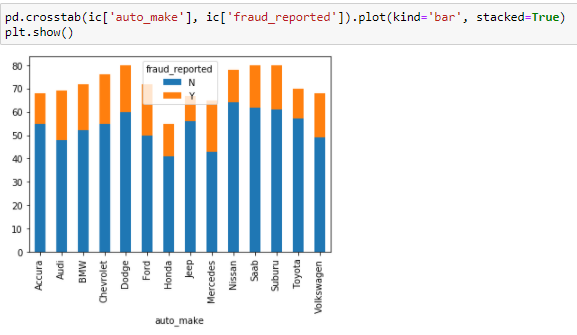
There is a significant relationship between insured\_occupation and fraud\_reported. We can see from the above figures that people whose occupation is exec-managerial are most likely to commit fraud.

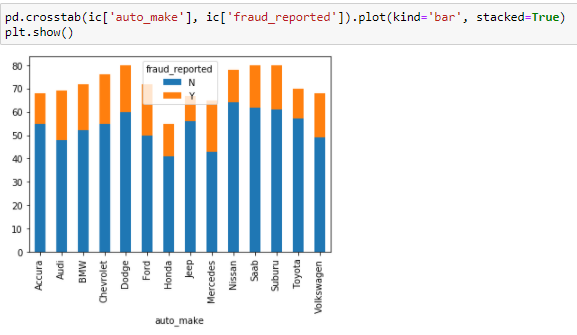
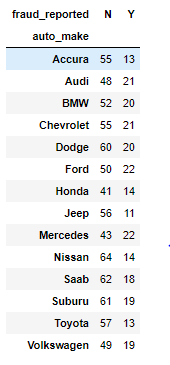
Impact of collision\_type on fraud\_reported can be seen below.

From the above figures, we can see that the fraudulent cases are more likely where there was a Rear collision, followed by front and side collisions.

Bar plot between auto\_make and fraud\_reported is as shown below.



From the above figures, we can see that fraudulent cases are more likely to occur where the auto\_make is Mercedes and Ford followed by Chevrolet and Dodge.

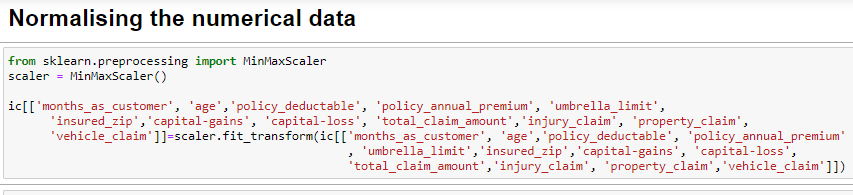
**Pre-processing Pipeline:**

The data set has variables in both object type and numerical type (int and float)

Therefore we have to pre-process the data to move forward.

All the float type or int type variables should be converted into the same scale since the range of values of raw data varies widely, in some machine learning algorithms, objective functions do not work correctly without normalization.

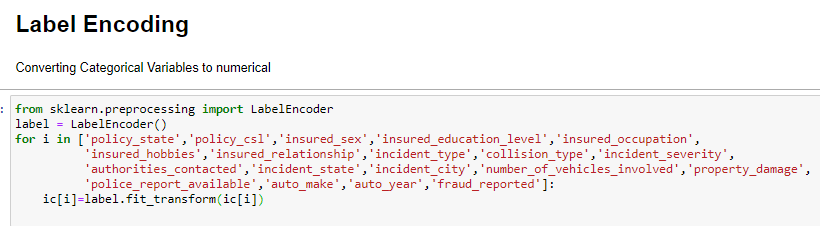
Therefore normalization is to be performed only on the numerical type (int and float type) variables.



After the Normalising is done all the numerical data fall under the same range or scale

We can see that there are object-type variables too. These variables contain string data that cannot be passed into the machine learning model as it won’t be able to recognize string data type. It only recognizes numerical data.

Therefore we need to convert the string data into numerical data. This can be done by manually encoding or by using an encoder such Label Encoder, one-hot encoder etc. For example: the target variable fraud\_reported consists of only two unique values, Y & N. after encoding this will get converted to 0 and 1. Similarly, if there are three unique values then it will be converted to 0,1, and 2.



**Correlation between 'fraud\_reported' and 'Independent features'**



fraud\_reported 1.000000

vehicle\_claim 0.170049

total\_claim\_amount 0.163651

property\_claim 0.137835

injury\_claim 0.090975

umbrella\_limit 0.058622

number\_of\_vehicles\_involved 0.051839

witnesses 0.049497

bodily\_injuries 0.033877

insured\_sex 0.030873

policy\_state 0.029432

insured\_relationship 0.021043

months\_as\_customer 0.020544

insured\_zip 0.019368

property\_damage 0.017202

policy\_deductable 0.014817

incidentmonth 0.014495

age 0.012143

insured\_education\_level 0.008808

auto\_year 0.007928

incident\_hour\_of\_the\_day 0.004316

insured\_occupation 0.001564

policy\_annual\_premium -0.014480

capital-loss -0.014863

collision\_type -0.017315

capital-gains -0.019173

auto\_make -0.027519

police\_report\_available -0.027768

policy\_csl -0.037190

incident\_city -0.040403

incidentday -0.044151

authorities\_contacted -0.045802

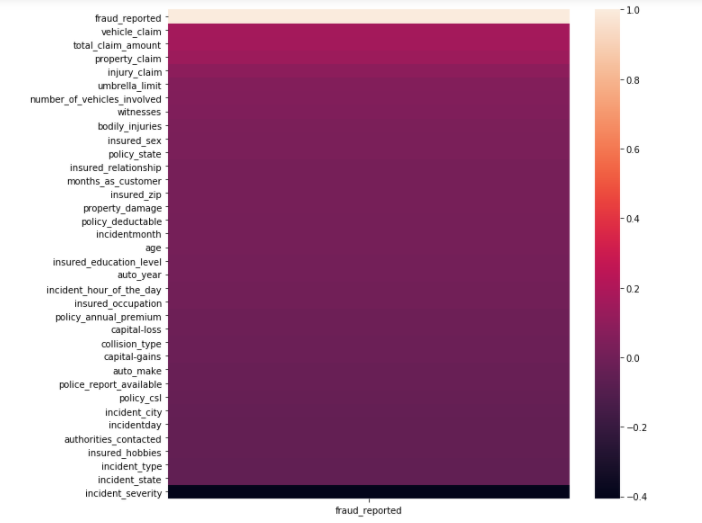
insured\_hobbies -0.046838

incident\_type -0.050376

incident\_state -0.051407

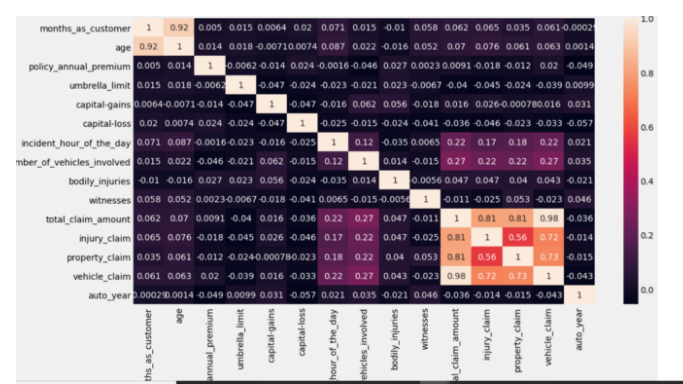
incident\_severity -0.405988

Name: fraud\_reported, dtype: float64



From above we can see the important features in descending order from top to bottom.

Now we can check the correlation between all the variables. (Note: correlation of all independent variables can be only done after encoding as correlation does not consider string values)



We can see from the above correlation heat map that correlation is high between month\_as\_customer and age as they both represent no. of months. We can also see there is a high correlation for total\_claim\_amount, injury\_claim, property\_claim, and vehicle\_claim as total\_claim is the sum of injury\_claim, property\_claim and vehicle\_claim. Therefore dropping them will not affect the dataset

Now the preprocessing is completed. We now have to move to data modeling and prediction.

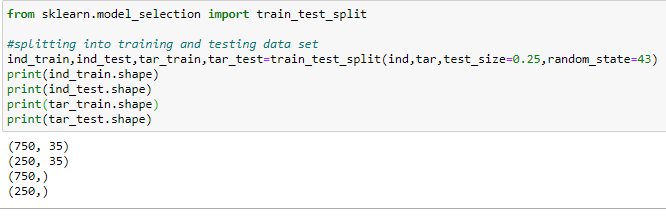
**Building Machine Learning Models:**

We have to now split the data into independent and target variables.



Here the target variable is fraud\_reported and the rest of them are independent variables.

We have to now split the independent and target variables into training and testing datasets as shown below.

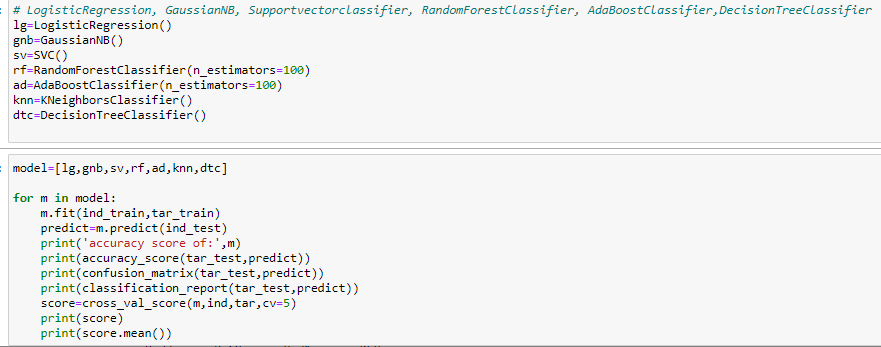


We will use a machine-learning algorithm to learn from the training set and use the model to predict the testing set and compare it with the predicted data with the target testing set to know how close the values. If the error between the predicted and target testing data is less that means the accuracy of the model is high and we can use this model to predict the result of similar datasets.

In this, we have used 7 Machine learning Algorithms

* LogisticRegression
* GaussianNB
* Supportvectorclassifier
* RandomForestClassifier
* AdaBoostClassifier
* KNeighborsClassifier
* DecisionTreeClassifier

We can train and predict the data using the above 7 ML algorithms and save the model which has the highest frequency.



**Cross Validation:**

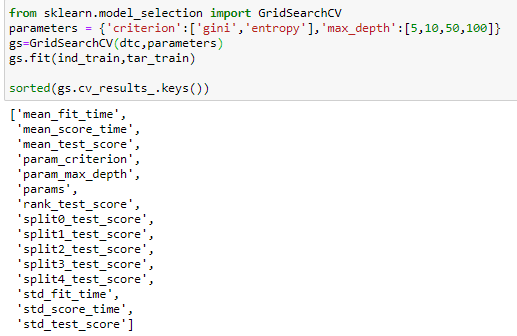
K-Folds cross validation is one method that attempts to maximize the use of the available data for training and then testing a model. It is particularly useful for assessing model performance. "Cross\_val\_score" splits the data into say 5 folds, then for each fold, it fits the data on 4 folds and scores the 5th fold. Then it gives you the 5 scores from which you can calculate a mean and variance for the score. It is useful to tune parameters and to get an estimate of the score.

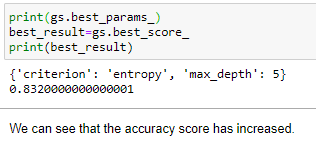
Accuracy score, Cross\_Validation score and Standard Deviation are given in the below table:

|  |  |  |  |
| --- | --- | --- | --- |
| **Machine Learning Model** | **Accuracy** | **Cross\_Validation score** | **Standard Deviation** |
| LogisticRegression | 0.756 | 0.771 | 0.034771 |
| GaussianNB | 0.763 | 0.719 | 0.053749 |
| Supportvectorclassifier | 0.74 | 0.753 | 0.032573 |
| RandomForestClassifier | 0.76 | 0.777 | 0.044733 |
| AdaBoostClassifier | 0.792 | 0.801 | 0.054672 |
| KNeighborsClassifier | 0.696 | 0.72 | 0.052536 |
| DecisionTreeClassifier | 0.828 | 0.767 | 0.047339 |

According to Cross val score and accuracy we can see that the DecisionTreeClassifier has the least difference between Accuracy and Cross val score, therefore we select DecisionTreeClassifier model.

We can use Gridsearch CV to get the best parameters of the selected model as shown below:





We can see that by using the best parameters the accuracy score of the model has increased.

# AUC ROC CURVE:

# AUC: Area Under the curve;   ROC: Receiver Operator Characteristic

# The greater the ROC score the better is the model. If ROC=1, then it perfectly fits.

# If the maximum of the area falls under True positive then the model is doing good.

# 

# 

We can see that the ROC Score is 0.74 and the area under the curve falls under True Positive Rate, Therefore, we can conclude that the model is performing well and we need to save the model in obj file for future use.



**Concluding Remarks:**

From the above results of the data modeling and prediction we can see that the Decision Tree Model is performing well as the accuracy score, cross val score and Roc score are good also the maximum of the area under the curve fall under true positive rate. Therefore we can save the model as .obj file so that it can be used to predict the result of the different data sets.

In this kind of problems Pre-processing and data-cleaning is the most important thing. We need to handle both the categorical and numerical data properly and also need to check by building different ML model on the same dataset. We need to check accuracy and cross val score of each model and chose the one which has the best of the same.