# INSURANCE CLAIM FRAUD DETECTION

Insurance fraud is a significant issue that leads to substantial financial losses for insurance companies, estimated to be billions of dollars annually. It also results in higher premiums for policyholders and undermines the integrity of the insurance system. Detecting and preventing fraud is essential to mitigate these impacts and maintain trust in the industry.

Insurance claim fraud detection is a multifaceted process that employs a variety of techniques to identify and prevent fraudulent activities. While challenges exist, advancements in data analytics, machine learning, and other technologies continue to enhance the effectiveness and efficiency of fraud detection systems, ultimately benefiting both insurance companies and policyholders.

Insurance claim fraud detection is crucial for maintaining the financial health and stability of insurance companies. Fraudulent claims result in substantial financial losses, draining resources that could otherwise be used for legitimate claims and business growth. By implementing effective fraud detection measures, insurers can significantly reduce unnecessary payouts, thereby protecting their financial reserves and ensuring the company's long-term viability.

Customer trust and satisfaction are also significantly enhanced through robust fraud detection systems. When customers perceive that an insurance company is committed to fairness and integrity, their trust in the insurer increases.

Effective fraud detection systems help insurance companies to:

- Reduce Financial Losses: Minimize payouts on fraudulent claims, protecting the company's bottom line.
- Maintain Lower Premiums: Preventing fraud helps keep insurance premiums affordable for honest policyholders.
- Enhance Trust: Build and maintain trust with customers by ensuring fair handling of claims.
- Improve Operational Efficiency: Automating fraud detection processes can streamline operations and reduce the burden on human investigators.

In this project, we are provided 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, I worked with some auto insurance data to demonstrate how you can create a predictive model that predicts if an insurance claim is fraudulent or not. This dataset consists of 1000 rows, 40 features describing each policy characteristics and target variable. fraud_reported is target variable to be predicted. As target variable is categorial in nature, this case study falls into classification machine learning problem. We have two objectives here:

1. Which key factors result in fraud being reported?
2. Building ML Model for predicting fraud.

## Data Preparation: Load, Clean and Format

Let's begin with importing libraries for EDA and dataset itself.

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
# Importing Loan Predication CSV dataset file using pandas
column_names=['months_as_customer', 'age', 'policy_number', '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', 'capital-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']
df=pd.read_csv("Automobile_insurance_fraud.csv",header=None, names=column_names)
```

[3]:
```
df.head()
```

[3]:

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

5 rows × 40 columns

[4]:
```
print('No of Rows:',df.shape[0])
print('No. of Columns:',df.shape[1])
```

```
No of Rows: 1000
No. of Columns: 40
```

Checking different datatypes in dataset: -

```
[5]: df.info()
```

```
<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        909 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
```

We have 21 features with object datatypes and rest are Numeric feature with int64and float64

Above nomenclature will help in better understanding of data when we perform EDA in this case study.

**Data Integrity Check:** Dataset can have missing values, duplicated entries and whitespaces. Now we will perform this integrity check of dataset.

```
[9]: sns.heatmap(df.isna())
     df.isna().sum()
```

```
[9]: 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                178
     incident_severity               0
     authorities_contacted          91
     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
     _c39                         1000
     dtype: int64
```

there is missing data! So lets remove nulls and dropping unnecessary columns.

```
[5]: df.duplicated('policy_number').sum()
```

```
[5]: 0
```

```
[6]: df.isin([' ','NA','-']).sum().any()
```

```
[6]: False
```

Dataset doesn't contain Any duplicate entry, whitespace, 'NA', or '-'.

```
[11]: # Replacing  Nulls with Mode of the column because it contain categorical data.
      df['collision_type'].fillna(value=df['collision_type'].mode()[0], inplace= True)
      df['property_damage'].fillna(value=df['property_damage'].mode()[0], inplace= True)
      df['police_report_available'].fillna(value=df['police_report_available'].mode()[0], inplace= True)

[12]: df['authorities_contacted'].fillna(value=df['authorities_contacted'].mode()[0], inplace= True)

[13]: df.isna().sum().any()

[13]: False

[14]: # Droping unnecessary columns
      df.drop(['incident_location','insured_zip','policy_number'],axis=1,inplace=True)

[15]: # Spliting and extracting policy_csl at '/'
      df['CSL_Personal']=df.policy_csl.str.split('/',expand=True)[0]
      df['CSL_Accidental']=df.policy_csl.str.split('/',expand=True)[1]

[16]: # Now we can drop policy_csl column
      df.drop("policy_csl",axis=1,inplace=True)

[17]: # Converting Date columns from object type into datetime data type
      df['policy_bind_date']=pd.to_datetime(df['policy_bind_date'])
      df['incident_date']=pd.to_datetime(df['incident_date'])

[18]: # Extracting Day, Month and Year column from policy_bind_date
      df['policy_bind_day'] = df['policy_bind_date'].dt.day
      df['policy_bind_month'] = df['policy_bind_date'].dt.month
      df['policy_bind_year'] = df['policy_bind_date'].dt.year

      # Extracting Day, Month and Year column from incident_date
      df['incident_day'] = df['incident_date'].dt.day
      df['incident_month'] = df['incident_date'].dt.month
      df['incident_year'] = df['incident_date'].dt.year

[19]: # Since Extraction is done now we can Drop policy_bind_date and incident_date columns
      df.drop(['policy_bind_date','incident_date'],axis=1,inplace=True)

[20]: # Incident year for all data is 2015 so we gone drop it.
      df.drop(['incident_year'],axis=1,inplace=True)

[21]: # Lets extract age of the vehicle from auto_year by subtracting it from the year 2018
      df['Automobile_Age']=2015 - df['auto_year']
      # Droping auto year column
      df.drop("auto_year",axis=1,inplace=True)
```

# Splitting features

```
]: Category = ['policy_state',  'insured_sex', 'insured_education_level', 'insured_occupation',
              'insured_hobbies', 'insured_relationship', 'incident_type', 'collision_type', 'incident_severity',
              'authorities_contacted','incident_state', 'incident_city', 'property_damage','police_report_available',
              'auto_make','auto_model', 'fraud_reported']

   Numerical = ['months_as_customer','CSL_Personal','CSL_Accidental', 'age', 'policy_deductable', 'umbrella_limit', 'capital-gains', 'capital-loss',
               'incident_hour_of_the_day', 'number_of_vehicles_involved', 'bodily_injuries', 'witnesses',
               'total_claim_amount','injury_claim', 'property_claim', 'vehicle_claim', 'Automobile_Age','policy_annual_premium']
```

```
]: df.columns.to_series().groupby(df.dtypes).groups
```

```
]: {int32: ['policy_bind_day', 'policy_bind_month', 'policy_bind_year', 'incident_day', 'incident_month'], int64: ['months_as_customer', 'age', 'policy_ded
   uctable', 'umbrella_limit', 'capital-gains', 'capital-loss', 'incident_hour_of_the_day', 'number_of_vehicles_involved', 'bodily_injuries', 'witnesses',
   'total_claim_amount', 'injury_claim', 'property_claim', 'vehicle_claim', 'Automobile_Age'], float64: ['policy_annual_premium'], object: ['policy_state',
   'insured_sex', 'insured_education_level', 'insured_occupation', 'insured_hobbies', 'insured_relationship', 'incident_type', 'collision_type', 'incident_
   severity', 'authorities_contacted', 'incident_state', 'incident_city', 'property_damage', 'police_report_available', 'auto_make', 'auto_model', 'fraud_r
   eported', 'CSL_Personal', 'CSL_Accidental']}
```

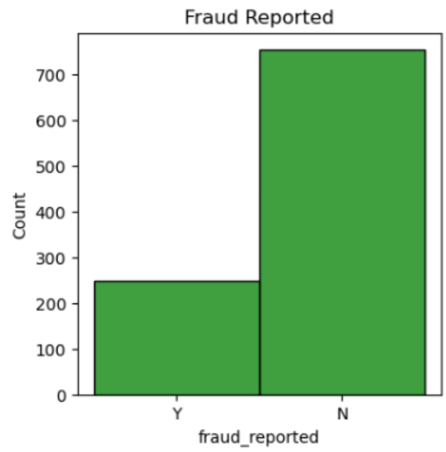
```
]: for i in Category:
       print(df[i].value_counts())
       print("="*100)
```

# Exploratory data analysis

**EXPLORATORY DATA ANALYSIS REFERS TO THE CRITICAL PROCESS OF PERFORMING INITIAL INVESTIGATIONS ON DATA SO AS TO DISCOVER PATTERNS, TO SPOT ANOMALIES, TO TEST HYPOTHESIS AND TO CHECK ASSUMPTIONS WITH THE HELP OF SUMMARY STATISTICS AND GRAPHICAL REPRESENTATIONS.**

## Univariate Analysis

```
[21]: # Plotting histogram for  target variables.
      plt.figure(figsize=(4,4))
      sns.histplot(df['fraud_reported'],color='g')
      plt.title('Fraud Reported')
      plt.show()
```




'fraud_reported' is our target variable to be predicted. From hist plot we can say dataset is imbalanced in nature. *making our dataset to be consider as imbalanced* since much of the fraud was not reported.

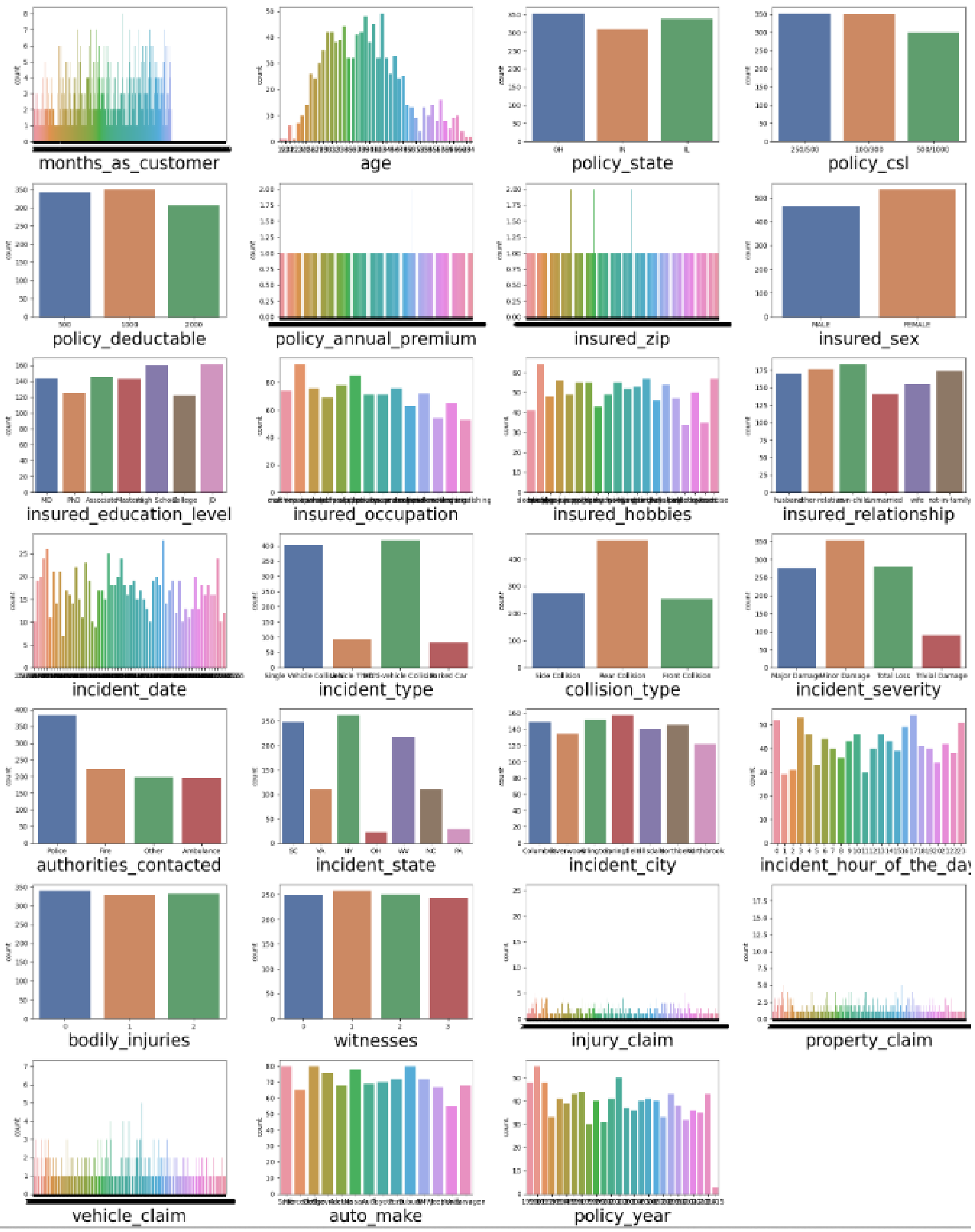
In this dataset we have features like injury_claim, property_claim, vehicle_claim which are inter related with each other. Let investigate this by visualisation of these features one by one to gain more insights.

```
sns.set_palette('deep')
plt.figure(figsize=(20,25))

categories =['months_as_customer', 'age',
       'policy_state', 'policy_csl', 'policy_deductable',
       'policy_annual_premium', 'insured_zip', 'insured_sex',
       'insured_education_level', 'insured_occupation', 'insured_hobbies',
       'insured_relationship','incident_date', 'incident_type', 'collision_type',
       'incident_severity','authorities_contacted', 'incident_state', 'incident_city',
        'incident_hour_of_the_day','bodily_injuries','witnesses',
       'injury_claim', 'property_claim', 'vehicle_claim', 'auto_make','policy_year']

for i, category in enumerate(categories, 1):
    plt.subplot(7, 4, i)
    sns.countplot(data=df, x=category)
    plt.xlabel(category, fontsize=25)

plt.tight_layout()
plt.show()
```

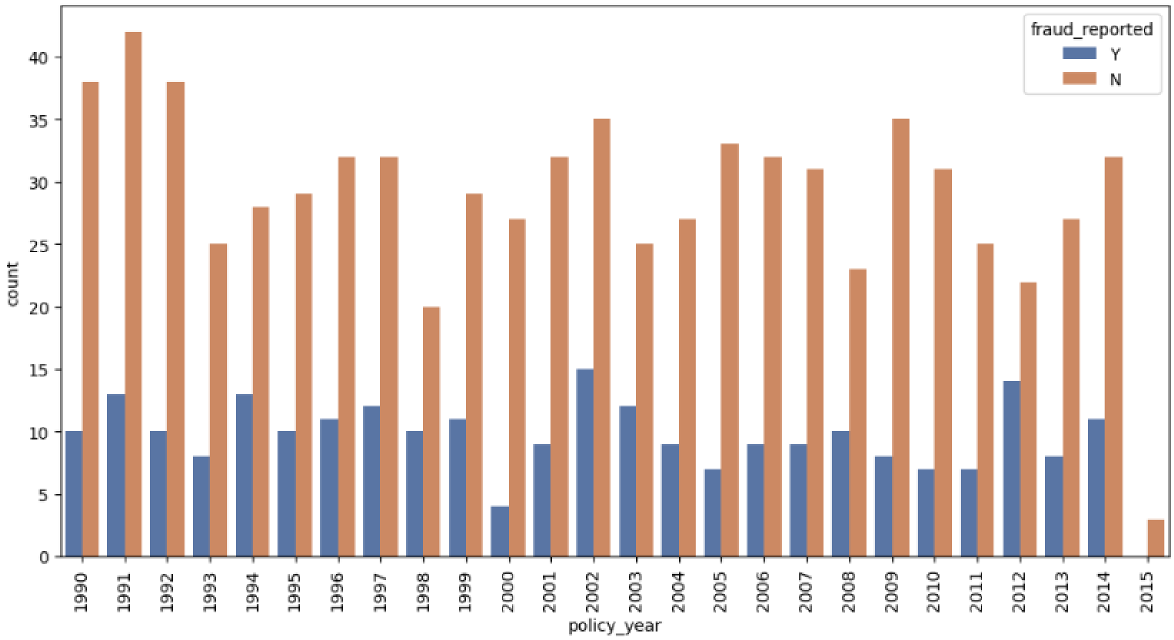


**Observation :**

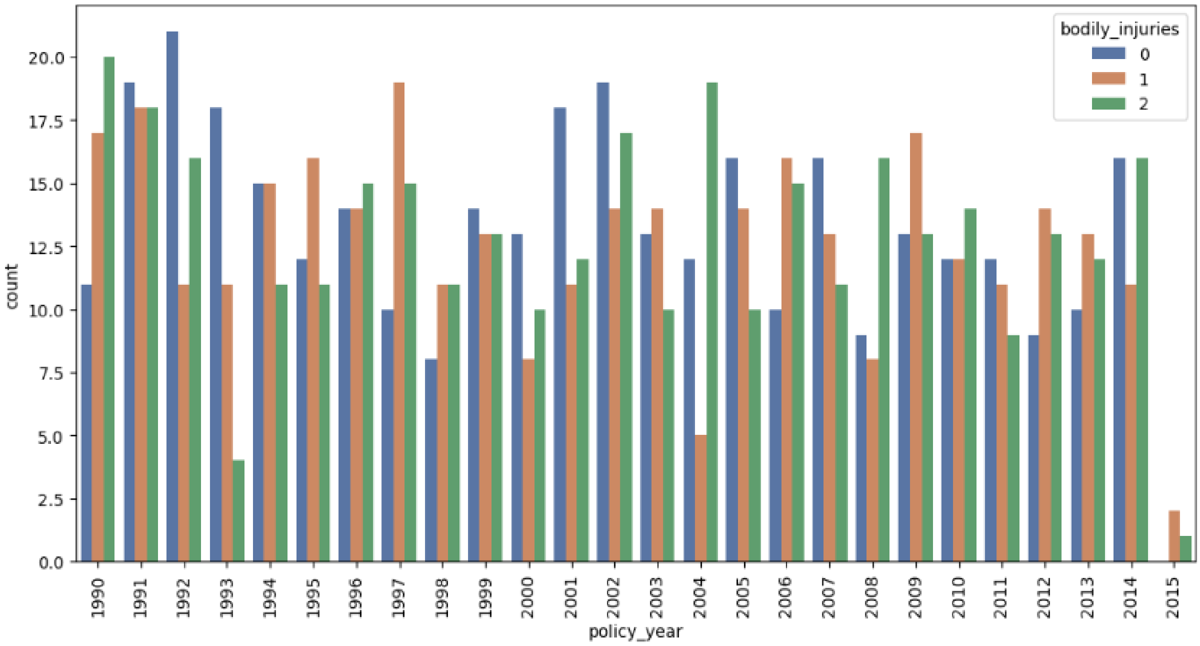
- Maximum fraud cases comes from people with age group of 31-50 year.
- Very few cases in 60+ year old peoples.
- Out of all cases around 24.7 % cases are Fraud.
- Almost same amout of cases come from each state.
- Maximum fraud cases come from state of Ohio.
- Number of claims come from female is higher than which reported by male insured.
- Almost same amount of fraud cases comes from same gender.

## Bivariate Analysis

```
# Analysis with Target variable.
plt.figure(figsize=(12,6))
sns.countplot(data=df, x='policy_year', hue='fraud_reported')
plt.xticks(fontsize=10,rotation=90)
plt.show()
```



```
# Analysis with Target variable.
plt.figure(figsize=(12,6))
sns.countplot(data=df, x='policy_year', hue='bodily_injuries')
plt.xticks(fontsize=10,rotation=90)
plt.show()
```




## Observation :

- Most of case comes from Multi-vehicle and single vehicle collision.
- Some claims are due to automobile robbery.
- **One claim out of three claim is fraud in multi or single vehicle collision incident.**

## Feature Engineering: Data Pre-processing

*Feature engineering is the process of transforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data.*

Feature Engineering is very important step in building Machine Learning model. Some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used. In Feature engineering can be done for various reason. Some of them are mention below:

1. Feature Importance: An estimate of the usefulness of a feature
2. Feature Extraction: The automatic construction of new features from raw data (Dimensionality reduction Technique like PCA)
3. Feature Selection: From many features to a few that are useful

4. Feature Construction: The manual construction of new features from raw data (For example, construction of new column for month out date - mm/dd/yy)

There are Varity of techniques use to achieve above mention means as per need of dataset. Some of Techniques important are as below:

- Handling missing values
- Handling imbalanced data using SMOTE
- Outliers' detection and removal using Z-score, IQR
- Scaling of data using Standard Scalar or Minmax Scalar
- Binning whenever needed
- Encoding categorical data using one hot encoding, label / ordinal encoding
- Skewness correction using Boxcox or yeo-Johnson method
- Handling Multicollinearity among feature using variance inflation factor
- Feature selection Techniques:
    - ✓ Correlation Matrix with Heatmap
    - ✓ Univariate Selection – SelectKBest
    - ✓ ExtraTreesClassifier method

In this case study we will use some of the mention feature engineering Techniques one by one.

## 1. Dropping unnecessary features

Feature like 'auto_year', 'policy_number' are irrelevant from ML model building perspective. We will drop these features.

```
df.drop(['auto_year','policy_number'],axis=1,inplace=True)
```

## 2. Encoding Categorical & Ordinal Features

Label Encoding is employed over categorical features.

```
Category = ['policy_state', 'insured_sex', 'insured_education_level', 'insured_occupation',
            'insured_hobbies', 'insured_relationship', 'incident_type', 'collision_type', 'incident_severity',
            'authorities_contacted', 'incident_state', 'incident_city', 'property_damage', 'police_report_available',
            'auto_make', 'auto_model', 'fraud_reported', 'CSL_Personal', 'CSL_Accidental']
```

```
# Using Label Encoder on categorical variable
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for i in Category:
    df[i] = le.fit_transform(df[i])
df.head()
```

Since now encoding is done we will move towards outliers' detection and removal.

## 3. Outliers' detection and removal

*Identifying outliers and bad data in your dataset is probably one of the most difficult parts of data clean-up, and it takes time to get right. Even if you have a deep understanding of statistics and how outliers might affect your data, it's always a topic to explore cautiously.*

- ***Page 167, Data Wrangling with Python, 2016***

Machine learning algorithms are sensitive to the range and distribution of attribute values. Data outliers can spoil and mislead the training process resulting in longer training times, less accurate models and ultimately poorer results. Outliers can be seen in boxplot of numerical feature. .

```
plt.figure(figsize=(12,30))
index=1
for column in Numerical:
    if index <=21:
        ax = plt.subplot(9,4,index)
        sns.boxplot(df[column], color='c')
        plt.xlabel(column,fontsize=12)
    index+=1
plt.show()
```

```
from scipy.stats import zscore
z = np.abs(zscore(df))
threshold = 3
df1 = df[(z<3).all(axis = 1)]

print ("Shape of the dataframe before removing outliers: ", df.shape)
print ("Shape of the dataframe after removing outliers: ", df1.shape)
print ("Percentage of data loss post outlier removal: ", (df.shape[0]-df1.shape[0])/df.shape[0]*100)

df=df1.copy() # reassigning the changed dataframe name to our original dataframe name
```

## 4. Checking Skewness

```
df[Numerical].skew()
```

```
months_as_customer             0.362608
age                            0.475385
policy_deductable              0.476090
umbrella_limit                 1.801424
capital-gains                  0.466619
capital-loss                  -0.376884
incident_hour_of_the_day      -0.039280
number_of_vehicles_involved    0.509725
bodily_injuries                0.003757
witnesses                      0.026211
total_claim_amount            -0.593593
injury_claim                   0.271759
property_claim                 0.361356
vehicle_claim                 -0.620936
policy_bind_day                0.054173
policy_bind_month             -0.029021
policy_bind_year               0.065022
incident_day                   0.037814
incident_month                 0.259907
Automobile_Age                 0.054522
policy_annual_premium          0.035964
dtype: float64
```

With the above plot, it's evident that the skewness in several columns exceeds the permissible limit of -0.5 to 0.5, indicating a need for removal.

Approach:- Skewness removal through Power transformer

```
# Making the skew less than or equal to +0.5 and -0.5 for better prediction using yeo-johnson method
skew=['total_claim_amount','vehicle_claim']

# Importing Powertransformer
from sklearn.preprocessing import PowerTransformer
scaler = PowerTransformer(method='yeo-johnson')

# Transfroming skew data
df[skew] = scaler.fit_transform(df[skew].values)
```

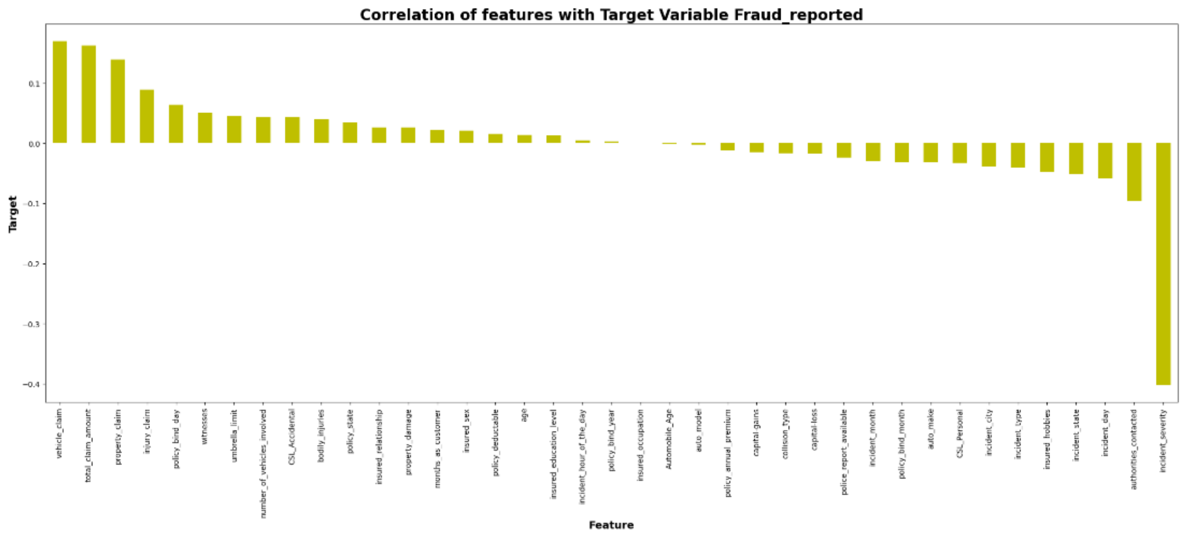
## Checking Skewness after transformation

```
df[skew].skew()
```

```
total_claim_amount   -0.508540
vehicle_claim        -0.521805
dtype: float64
```

## 5. Correlation Heatmap

Correlation Heatmap show in a glance which variables are correlated, to what degree, in which direction, and alerts us to potential multicollinearity problems. The bar plot of correlation coefficient of target variable with independent features shown below




## 6. Multicollinearity between features

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif = pd.DataFrame()
vif["VIF values"] = [variance_inflation_factor(X_scale,i) for i in range(len(X.columns))]
vif["Features"] = X.columns
vif
```

Variance Inflation factor imported from statsmodels.stats.outliers_influence to check multicollinearity between features.

| | VIF values | Features |
|---|---|---|
| 0 | 7.559692 | months_as_customer |
| 1 | 7.489042 | age |
| 2 | 1.052529 | policy_state |
| 3 | 1.067848 | policy_deductable |
| 4 | 1.048319 | policy_annual_premium |
| 5 | 1.039650 | umbrella_limit |
| 6 | 1.090014 | insured_sex |
| 7 | 1.057657 | insured_education_level |
| 8 | 1.034840 | insured_occupation |
| 9 | 1.073165 | insured_hobbies |
| 10 | 1.065090 | insured_relationship |
| 11 | 1.068746 | capital-gains |
| 12 | 1.069593 | capital-loss |
| 13 | 3.815669 | incident_type |
| 14 | 1.086157 | collision_type |
| 15 | 1.401586 | incident_severity |
| 16 | 1.354759 | authorities_contacted |
| 17 | 1.089963 | incident_state |
| 18 | 1.054723 | incident_city |
| 19 | 1.110886 | incident_hour_of_the_day |
| 20 | 3.650098 | number_of_vehicles_involved |
| 21 | 1.065034 | property_damage |
| 22 | 1.058702 | bodily_injuries |
| 23 | 1.079069 | witnesses |
| 24 | 1.091164 | police_report_available |
| 25 | 42923.761345 | total_claim_amount |
| 26 | 1696.696180 | injury_claim |
| 27 | 1693.990896 | property_claim |
| 28 | 21305.322421 | vehicle_claim |
| 29 | 1.089914 | auto_make |
| 30 | 1.090452 | auto_model |
| 31 | 1.234070 | CSL_Personal |
| 32 | 1.204257 | CSL_Accidental |
| 33 | 1.044421 | policy_bind_day |
| 34 | 1.053241 | policy_bind_month |
| 35 | 1.041757 | policy_bind_year |
| 36 | 1.069472 | incident_day |
| 37 | 1.114975 | incident_month |
| 38 | 1.063516 | Automobile_Age |

## 7. Handling imbalanced data using SMOTE

This two-class dataset is imbalanced (76% vs 24%). As a result, there is a possibility that the model built might be biased towards to the majority and over-represented class. We can resolve this by Synthetic Minority Oversampling Technique (SMOTE) to over-sample the minority class.

```
from imblearn.over_sampling import SMOTE
```

```
# Splitting data in target and dependent feature
X = df.drop(['fraud_reported'], axis =1)
Y = df['fraud_reported']
```

```
# Oversampleing using SMOTE Techniques
oversample = SMOTE()
X, Y = oversample.fit_resample(X, Y)
```

```
Y.value_counts()
```

```
fraud_reported
1    740
0    740
Name: count, dtype: int64
```

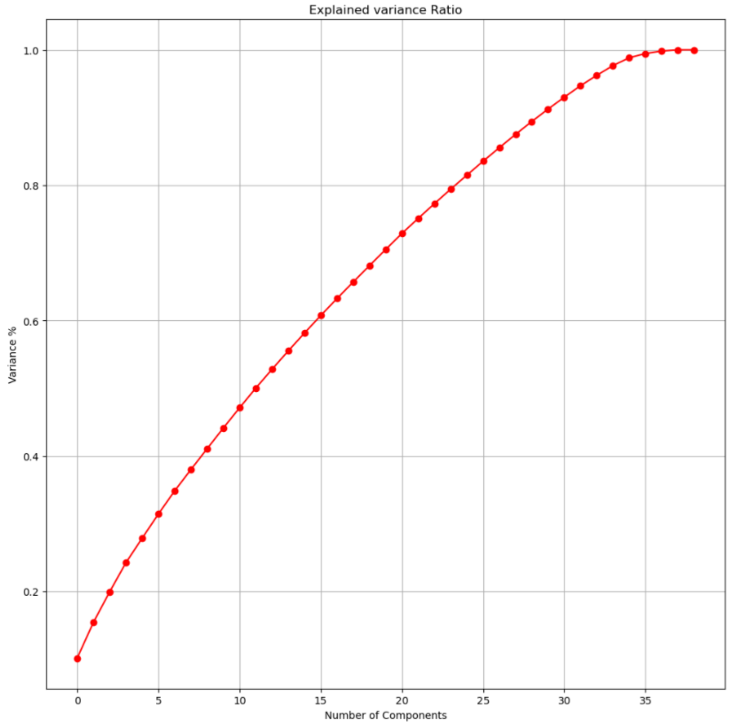
## 8. Scaling of data using Standard Scalar

```python
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
```

```python
X_scale=scaler.fit_transform(X)
```

9. Dimensionality Reduction Using PCA

PCA used find patterns and extract the latent features from our dataset.




We can see that 28 principal components attribute for 90% of variation in the data. PCA applied for 28 components.

```python
pca_new = PCA(n_components=28)
x_new = pca_new.fit_transform(X_scale)
```

```python
principle_x=pd.DataFrame(x_new,columns=np.arange(28))
```

## Machine Learning Model Building:

In this section we will build Supervised learning ML model-based classification algorithm. As objective is to predict fraud_reported in 'Yes' or 'No' leads to fall problem in domain of classification algorithm. train_test_split used to split data with size of 0.3

```
X_train, X_test, Y_train, Y_test = train_test_split(principle_x, Y, random_state=99, test_size=.3)
print('Training feature matrix size:',X_train.shape)
print('Training target vector size:',Y_train.shape)
print('Test feature matrix size:',X_test.shape)
print('Test target vector size:',Y_test.shape)
```

```
Training feature matrix size: (1036, 28)
Training target vector size: (1036,)
Test feature matrix size: (444, 28)
Test target vector size: (444,)
```

First we will build base model using logistic regression algorithim. Best random state is investigated using for loop for random state in range of (1,250).

```
maxAccu=0
maxRS=0
for i in range(1,250):
    X_train,X_test,Y_train,Y_test = train_test_split(principle_x,Y,test_size = 0.3, random_state=i)
    log_reg=LogisticRegression()
    log_reg.fit(X_train,Y_train)
    y_pred=log_reg.predict(X_test)
    acc=accuracy_score(Y_test,y_pred)
    if acc>maxAccu:
        maxAccu=acc
        maxRS=i
print('Best accuracy is', maxAccu ,'on Random_state', maxRS)
```

```
Best accuracy is 0.8063063063063063 on Random_state 143
```

Logistics regression model is train with random state 9. The evalution matrix along with classification report is as below :

```
Logistics Regression Evaluation


Accuracy Score of Logistics Regression : 0.7612612612612613


Confusion matrix of Logistics Regression :
 [[170  57]
 [ 49 168]]


classification Report of Logistics Regression
              precision    recall  f1-score   support

           0       0.78      0.75      0.76       227
           1       0.75      0.77      0.76       217

    accuracy                           0.76       444
   macro avg       0.76      0.76      0.76       444
weighted avg       0.76      0.76      0.76       444
```

As Now base model is ready with f1-score of 0.76, we will train model with different classification algorithm along with k-5 fold cross validation. The final evaluation matrix different classification algorithm is as shown table below:

| ML Algorithm | Accuracy Score | CVMean Score | f-1 Score | Recall | Precision |
|---|---|---|---|---|---|
| Logistics Regression | 0.76 | 0.756 | 0.76 | 0.75 | 0.78 |
| SVC | 0.8310 | 0.807 | 0.84 | 0.85 | 0.82 |
| GaussianNB | 0.7905 | 0.7722 | 0.80 | 0.80 | 0.79 |
| DecisionTreeClassifier | 0.7319 | 0.6905 | 0.72 | 0.68 | 0.77 |
| RandomForestClassifier | 0.8175 | 0.8087 | 0.82 | 0.84 | 0.81 |
| ExtraTreesClassifier | 0.8355 | 0.8216 | 0.84 | 0.84 | 0.84 |

(Min Value in column -Green, Max Value in column - Pink Colour )

***We can see that Extra Trees Classifier gives us maximum f1-score & mean cross validation score. We will perform hyper parameter tuning on extra trees classifier to build final ML Model***

```
from sklearn.model_selection import GridSearchCV
```

```
param = {
    'penalty': ['l1', 'l2'],
    'C': [0.001, 0.01, 0.1, 1, 10],
    'solver': ['liblinear', 'lbfgs', 'saga'],
    'max_iter': [100, 200, 300],
    'class_weight': [None, 'balanced']
}
```

```
grid_search = GridSearchCV(LogisticRegression(),param, cv=5, scoring='accuracy',verbose=5)
grid_search.fit(X_train, Y_train)
```

Next step is to build final machine learning model over best params in Hyper parameter tuning.

```
]: grid_search.best_params_
```

```
]: {'C': 0.01,
    'class_weight': None,
    'max_iter': 100,
    'penalty': 'l2',
    'solver': 'liblinear'}
```

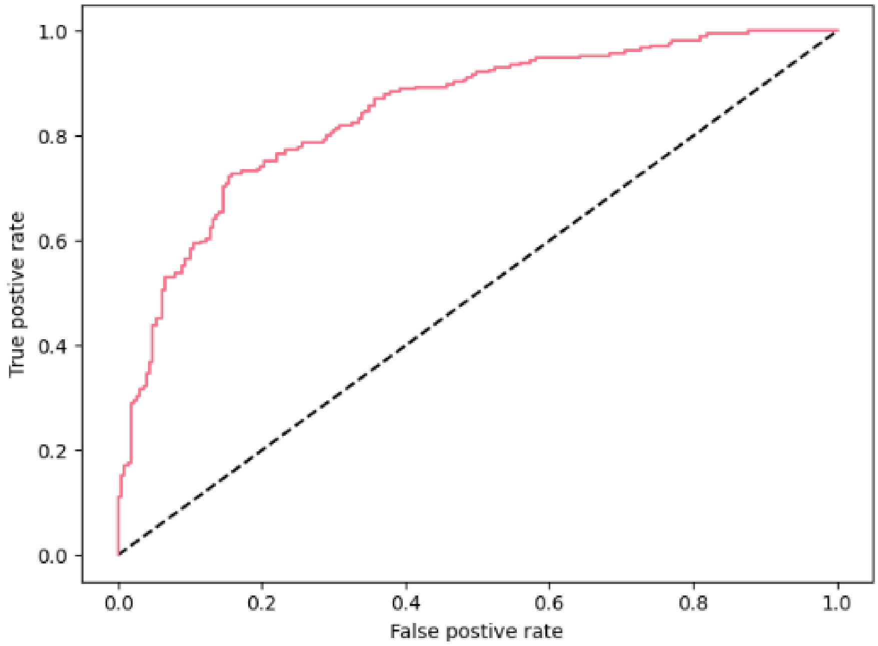
## Final Model

```
]: model = LogisticRegression(C = 0.001, class_weight = None, max_iter = 100, penalty = 'l2', solver = 'liblinear'
```

```
]: model.fit(X_train,Y_train)
y_pred=model.predict(X_test)
print('\033[1m'+'Accuracy Score :'+'\033[0m\n', accuracy_score(Y_test, y_pred))

Accuracy Score :
 0.7522522522522522
```

We can see that Final model with hyper parameter tuning leads to slight decrease in accuracy score from 0.7612 in original model to 0.7522. This complete possible We will use model with default values as our final model. AOC-ROC score of final model is shown below:




At last, we will save final model with joblib library, so it can be deploy on cloud platform.

```
import joblib
joblib.dump(model,'Insurance_claims_Final.pkl')
```

```
['Insurance_claims_Final.pkl']
```

# Predicting the Final Model

```
# Prediction
prediction = model.predict(X_test)
```

```
pred_=pd.DataFrame({'predicted values':y_pred,'actual values':Y_test})
pred_.head()
```

| | predicted values | actual values |
|---|---|---|
| 1035 | 1 | 1 |
| 337 | 0 | 0 |
| 514 | 0 | 0 |
| 855 | 1 | 1 |
| 1449 | 1 | 1 |

## Concluding Remarks on EDA and ML Model

- Individuals between the age of 60- 69 have second-cheapest car insurance rates, behind only those in their 50s., I can summarize that other than the age of 22, the older you are, the higher the auto insurance claim amount is..
- When comparing the longevity of customer membership to the total claims amount for auto insurance, I hypothesize that it would affect the amount.

Customer loyalty is important, so benefits for being a customer for a long duration of time would have an effect to how much money you would get back.

- The majority of individuals are customers up to around 300 months. However, it is incorrect to assume that customer longevity has a significant impact on total claims amount.
- The eldest vehicles made in 1995 had the largest amount of auto insurance claims than the more recent year (2015). This could be for numerous reasons. Damages to older cars could be significant or even dangerous, so the insurance would need to reimburse their customers.
- Different feature engineering techniques like balancing data, outliers' removal, label encoding, feature selection & PCA are perform on data.
- Extra trees Classifier model gives maximum Accuracy.

You can get code of this case study from my GitHub Profile