Insurance Claims - Fraud Detection With Machine Learning Classification Problem

Contents:

1. Problem Defination:

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, I have been provided the 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.

The difficulty with machine learning for fraud detection is that fraudulent insurance claims are far less frequent than legitimate ones. Unbalanced class categorization is a problem that causes issues like these.

Frauds are immoral and cost the business money. I can reduce losses for the insurance business by creating a model that can categorise auto insurance fraud. More revenue is generated through minimal losses.

Here I have worked with some auto insurance data to demonstrate how one can create a predictive model that predicts if an insurance claim is fraudulent or not. I have applied a step-by-step approach using a machine learning classification problem.

In this study, we will be looking forward to the significant independent variables that actually portrays and helps to predict further whether their could be a insurance fraud in specific case or not. The factors are based on the personality, choices, historical endowments, and relationship person holds.

We will be solving the following problems:

- What is the probability of an insurance claim to be a fraud?
- What are the key indicators of an insurance fraud?
- Considering the results, what strategies can be implemented to minimize future insurance fraud cases?

With auto insurance data, we have a standard supervised classification problem with a binary variable label, 0 (no fraud) and 1 (fraud). The target variable for this study is the chance that an insurance claim will be fraudulent.

Way to success:

In the absence of prior data, the model should be able to detect fraudulent claims reliably. The Since it's crucial to distinguish between fraudulent and legitimate claims, the area under the ROC curve (ROC AUC) will also be taken into account in the model selection process. This is due to the fact that fraud investigations may be time-consuming, expensive, and sometimes detrimental to the user experience. The ROC AUC must be greater than 0.50 to meet the requirement. Additionally, I want my ROC AUC to be at least 0.70.

2. Data Analysis:

In this case, we have the data for the auto insurance company that includes different independent variables holding significant affect on the insurance claims.

- **Data Discription**

Initially, we have loaded the dataset from the GitHub. It does have 1000 rows and 40 columns. In total we have 40 variables. Fraud reported is our target variable and rest of them are independent variables.

```
In [2]: df = pd.read_csv("https://raw.githubusercontent.com/mohittomar2008/Insurance-Claims--Fraud-Detection/main/Automobile_insurar
```

```
In [5]: df
```

Out[5]:

	months_as_customer	age	policy_number	policy_bind_date	policy_state	policy_csl	policy_deductable	policy_annual_premium	umbrella_limit	insure
0	328	48	521585	17-10-2014	OH	250/500	1000	1406.91	0	4
1	228	42	342868	27-06-2006	IN	250/500	2000	1197.22	5000000	4
2	134	29	687698	06-09-2000	OH	100/300	2000	1413.14	5000000	4
3	256	41	227811	25-05-1990	IL	250/500	2000	1415.74	6000000	6
4	228	44	367455	06-06-2014	IL	500/1000	1000	1583.91	6000000	6
...	...	...	...	...	...	...	...	...	...	
995	3	38	941851	16-07-1991	OH	500/1000	1000	1310.80	0	4
996	285	41	186934	05-01-2014	IL	100/300	1000	1436.79	0	6
997	130	34	918516	17-02-2003	OH	250/500	500	1383.49	3000000	4
998	458	62	533940	18-11-2011	IL	500/1000	2000	1356.92	5000000	4
999	456	60	556080	11-11-1996	OH	250/500	1000	766.19	0	6

1000 rows × 40 columns

In the dataset, various numerical and categorical columns contain information about relevant factors for individual insurance policies. Three data types has been included in the dataset - Object, Float and Integer. It has taken the memory space of 312.6+ KB.

```
In [4]: df.columns

Out[4]: Index(['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'],
      dtype='object')
```

There are in total 1000 rows and 40 columns available. Among which "Fraud_reported" is our target variable. Rest of the variables are independent and significant variables.

Data do hold some missing values in form of "?" but not in NaN or something. It has been handled later moving further with the exploratory data analysis.

From the general observation of the dataset and some statistical overviews, we got the general understanding of the data and how it needed to be processed.

- All variables hold equal number of rows.
- Standard deviation of almost all variables are quite high, implying data is not normally distributed. Some amount of skewness does exsist in the set. We will figure it out later.
- There is a significant difference in mean and median both --> Data is skewed.

```
In [9]: df.describe().T
```

Out[9]:

	count	mean	std	min	25%	50%	75%	max
months_as_customer	1000.0	2.039540e+02	1.151132e+02	0.00	115.7500	199.5	276.250	479.00
age	1000.0	3.894800e+01	9.140287e+00	19.00	32.0000	38.0	44.000	64.00
policy_number	1000.0	5.462386e+05	2.570630e+05	100804.00	335980.2500	533135.0	759099.750	999435.00
policy_deductable	1000.0	1.136000e+03	6.118647e+02	500.00	500.0000	1000.0	2000.000	2000.00
policy_annual_premium	1000.0	1.256406e+03	2.441674e+02	433.33	1089.6075	1257.2	1415.695	2047.59
umbrella_limit	1000.0	1.101000e+06	2.297407e+06	-1000000.00	0.0000	0.0	0.000	10000000.00
insured_zip	1000.0	5.012145e+05	7.170161e+04	430104.00	448404.5000	466445.5	603251.000	620962.00
capital-gains	1000.0	2.512610e+04	2.787219e+04	0.00	0.0000	0.0	51025.000	100500.00
capital-loss	1000.0	-2.679370e+04	2.810410e+04	-111100.00	-51500.0000	-23250.0	0.000	0.00
incident_hour_of_the_day	1000.0	1.164400e+01	6.951373e+00	0.00	6.0000	12.0	17.000	23.00
number_of_vehicles_involved	1000.0	1.839000e+00	1.018880e+00	1.00	1.0000	1.0	3.000	4.00
bodily_injuries	1000.0	9.920000e-01	8.201272e-01	0.00	0.0000	1.0	2.000	2.00
witnesses	1000.0	1.487000e+00	1.111335e+00	0.00	1.0000	1.0	2.000	3.00
total_claim_amount	1000.0	5.276194e+04	2.640153e+04	100.00	41812.5000	58055.0	70592.500	114920.00
injury_claim	1000.0	7.433420e+03	4.880952e+03	0.00	4295.0000	6775.0	11305.000	21450.00
property_claim	1000.0	7.399570e+03	4.824726e+03	0.00	4445.0000	6750.0	10885.000	23670.00
vehicle_claim	1000.0	3.792895e+04	1.888625e+04	70.00	30292.5000	42100.0	50822.500	79560.00
auto_year	1000.0	2.005103e+03	6.015861e+00	1995.00	2000.0000	2005.0	2010.000	2015.00

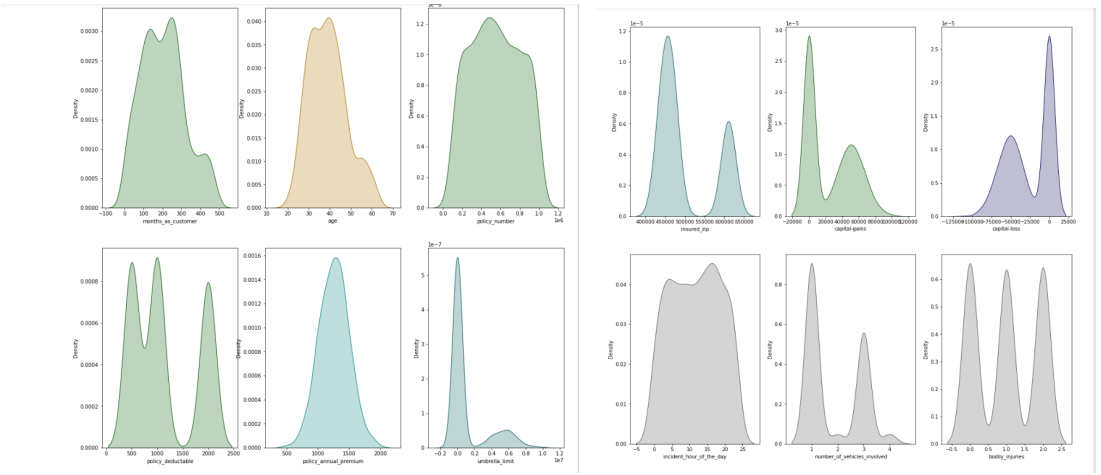
We have broken down the variables on the basis of categorical and continuous nature further. We got 21 categorical variables and 18 continuous variable.

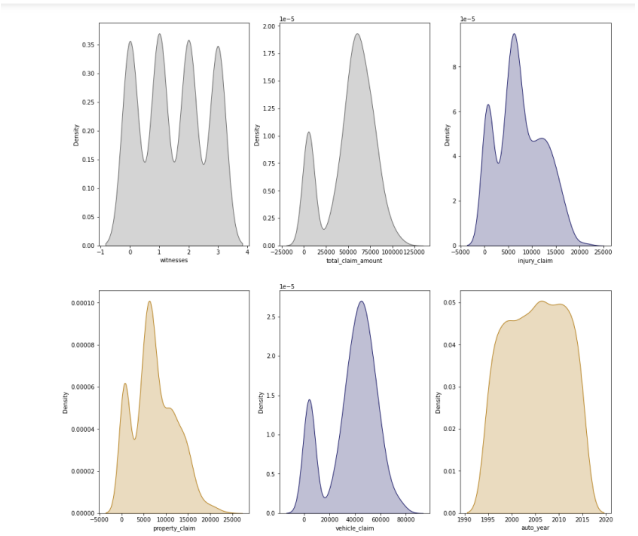
```
In [10]:
catg=[col for col in df.columns if df[col].dtypes=='object']
cont=[col for col in df.columns if df[col].dtypes!='object']

In [11]:
print(f'Number of Categorical features in the dataset: {len(catg)}')
print(f'Number of Continuous features in the dataset: {len(cont)}')

Number of Categorical features in the dataset: 21
Number of Continuous features in the dataset: 18
```

- **Numerical variables features**

Based on the numerical information and histograms, we can make the following observations:

- Some variables don't hold normal distribution and contains skewness on high level. Data is not bell-shaped.
- Some numerical features have a right-skewed distribution; indeed, some distributions are tail-heavy. In order to fit a model to the data, it may be necessary to transform the data prior to fitting it to a normal distribution.



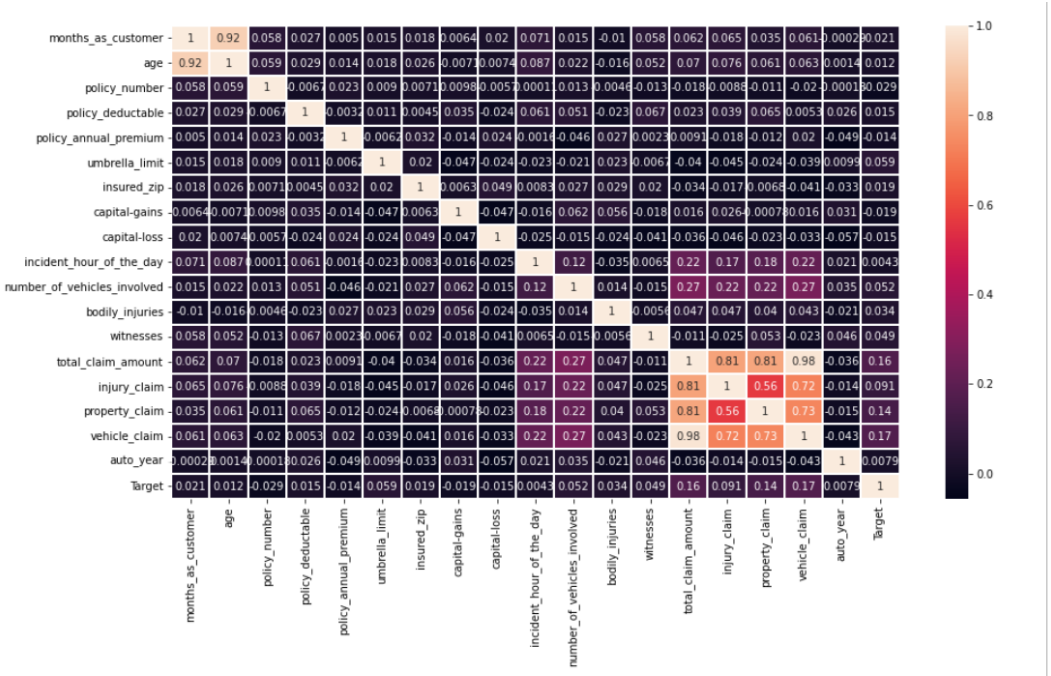


- **Correlation Among The Independent Variables**

Subsequently, correlations between ordinal, interval, and ratio continuous variables were examined. For variables, including the target, that had a Pearson's correlation coefficient of at least 0.3, a heatmap was created. Age and month as a client had a 0.92 correlation. Most likely because drivers get auto insurance when they own a vehicle, and the cost of auto insurance only rises with age.

There is a strong association between incident severity and various claim categories (r = 0.36–0.50). In addition to that, the data don't appear to show many relationships. Other than the possibility that all the claims are connected and that the overall number of claims has somehow adjusted for them, there doesn't seem to be a multicollinearity problem.

The other claims, on the other hand, offer some level of specificity that is not otherwise covered by total claims.



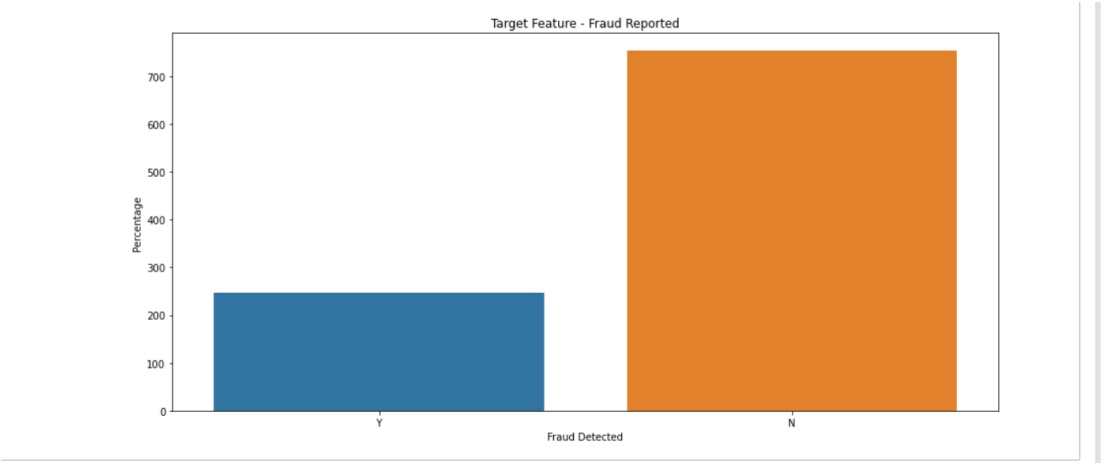
- **Feature Distribution - Univariate and Bivariate | Visualizing the variables**

- **Target Feature: Fraud Reported**

The target variable data is in two classified factors - Yes and No. In total, frauds which are not reported are 753 and frauds reported are 247. The fraud rate was 24.7%, while the non-fraudulent rate was 75.3%.

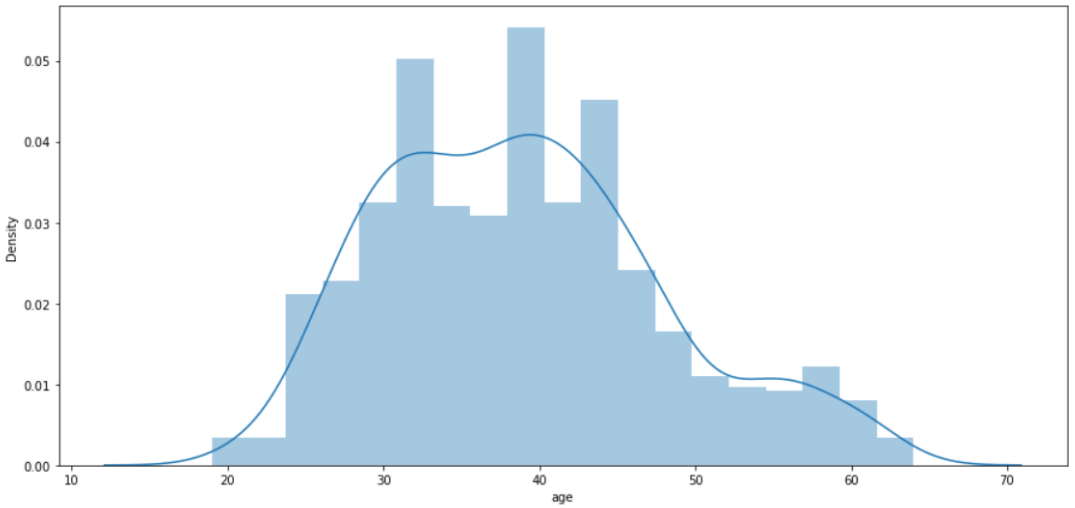
In order to know more in detail about the classification within the variable, let’s make the bar graph to know the estimate. It is not the case that fraud reports are high. Compariteviy, fruads

reported are very less in numbers. However, significant number of frauds has not been reported.



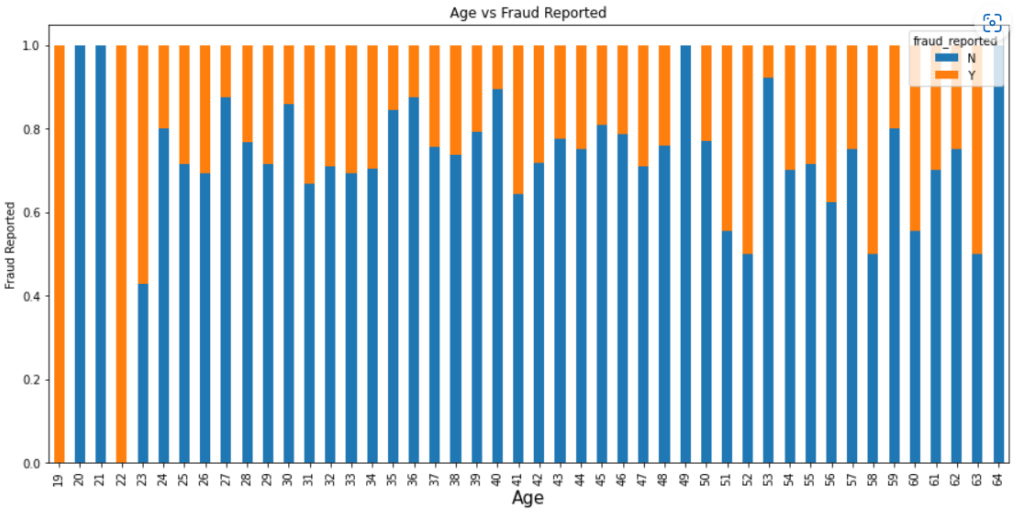
- **Age**

Generally people who have opted for the insurance, they are within the age group of 27 to 42. Somehow the age distribution is quite balanced.



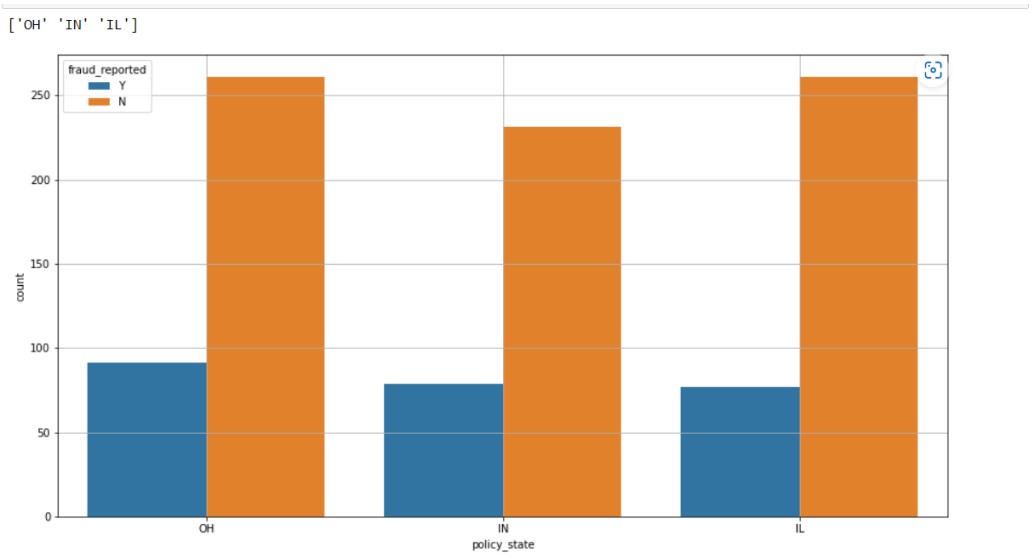
- **Age v/s Frauds Reported**

No frauds reports are done in the age group of 20,21,49, and 64. Highest fraud reports are done by the people of age 19,22, and 52.



- **Policy State**

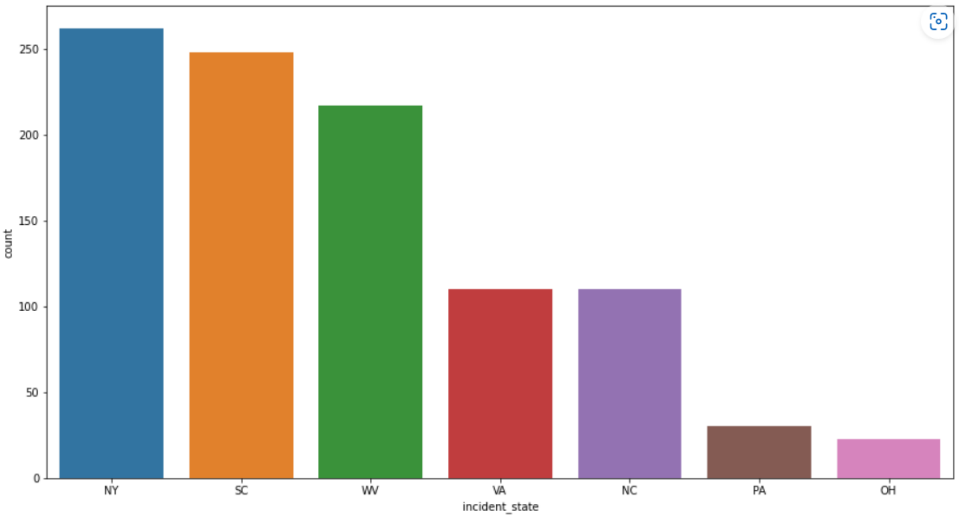
In terms of three types of policy states, all the polciies do hold the equal number of fraud case reported and not reported. There is nothing specific to some individual polciy.



- **Incident State**

There are seven incident states we have got with this distribution. The maximum number of incident state is with the respect of the state “NY”. Later on it is with SC and WV.

```
['SC' 'VA' 'NY' 'OH' 'WV' 'NC' 'PA']
Out[20]: <AxesSubplot:xlabel='incident_state', ylabel='count'>
```



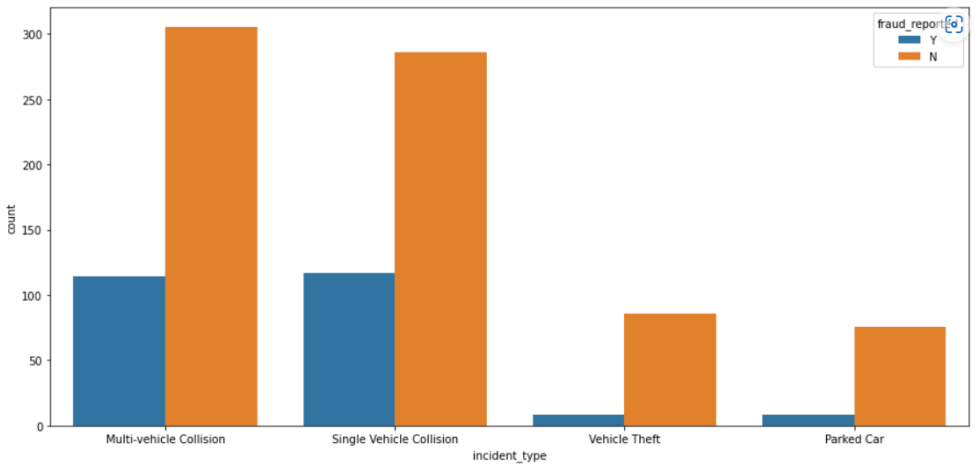
- **Incident Type**

Four types of incident types is given in the distribution. These are multi-vechile collision, single vehicle collision, vehicle theft, and parked car.

Maximum number of time the insurance fraud cases are reported is with the first two, that is, “Multi-vechile Collision” and “Single Vechile Collison”. However in other two, majority of the insurance fruad cases are not reported as per the data.

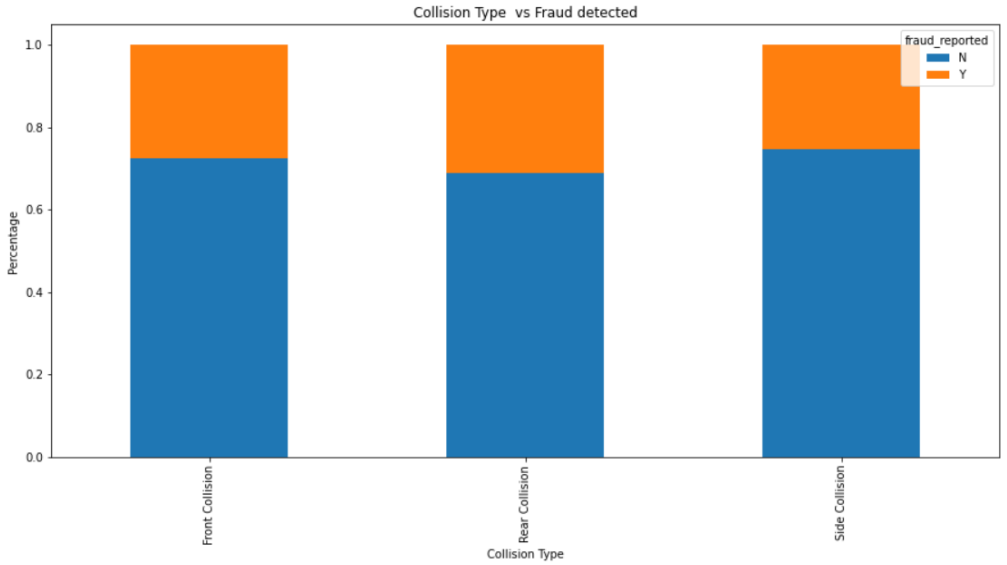
Maximum times frauds were reported in the case of vechile theft, parked car, and minimum in multi-vechile collision.

```
Out[21]: <AxesSubplot:xlabel='incident_type', ylabel='count'>
```



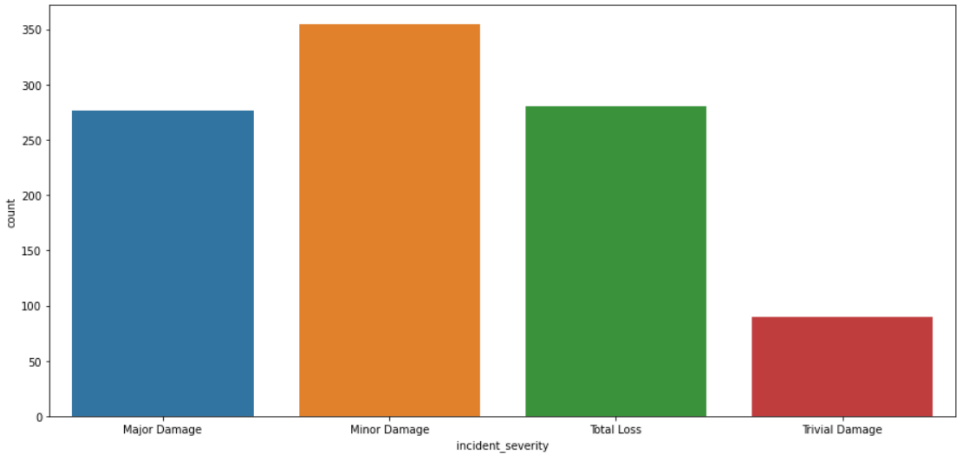
- **Collision Type**

We have three types of collision that is happened when the automobile is insured. These are Rear Collision, Side Collision, and Front Collision. The maximum number of fraud cases which were reported is with respect to the rear collision. However, majority of the fraud cases has not been reported in case of all three.



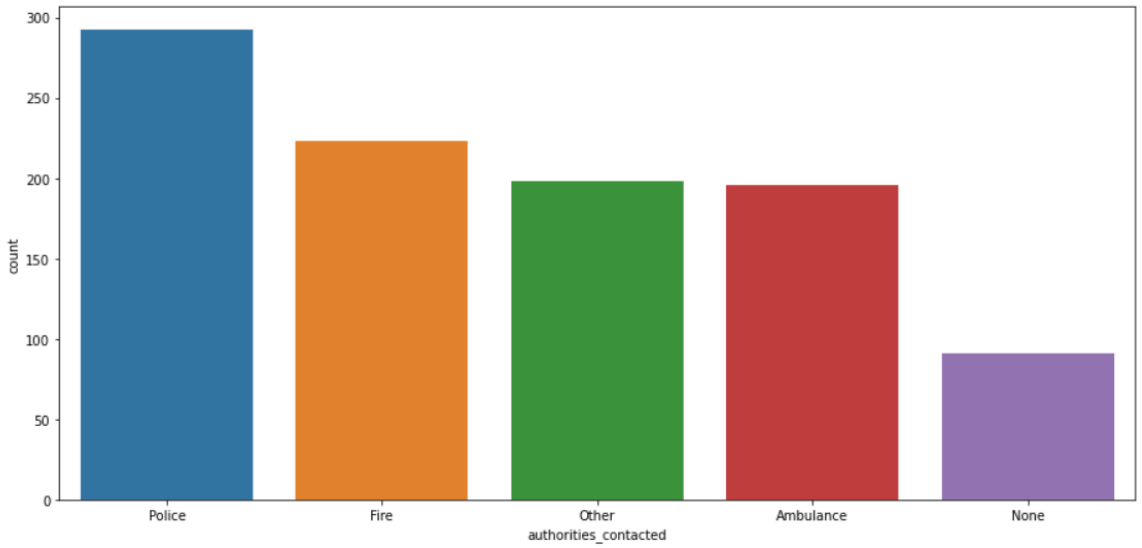
- **Incident Severity**

Majority of the times minor damage is recorded for the reports. Trivial damage case is very low.



- **Authorities contacted**

Police is contacted majority of the times if there is a incident of insurance fraud. Then there is a fire. Some individuals have not reported the fraud cases as well.



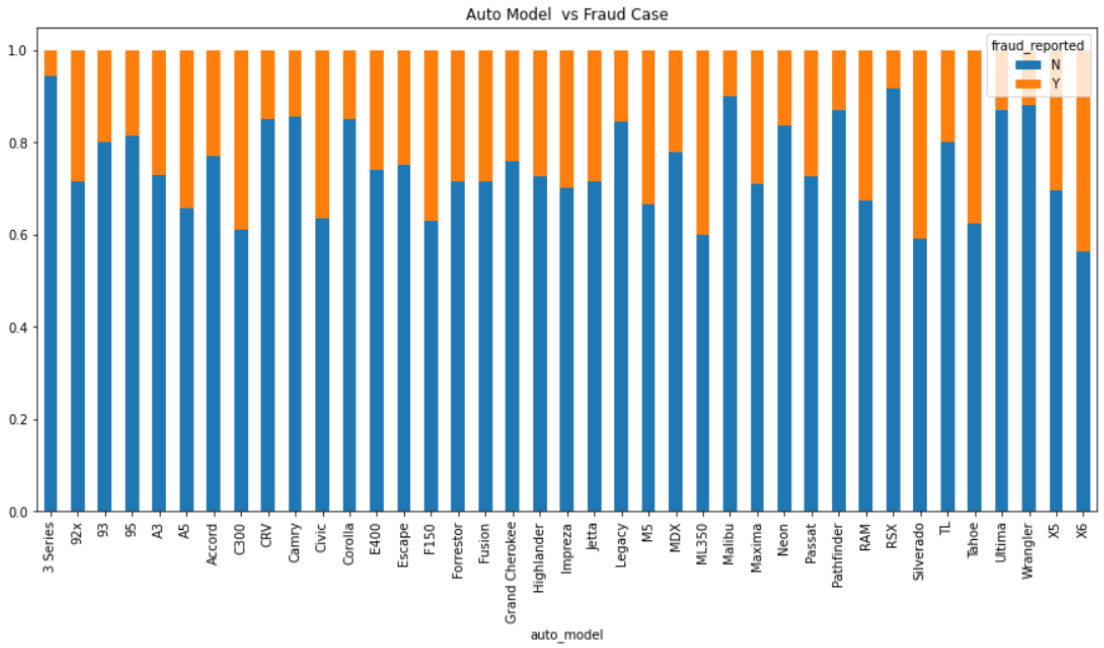
- **Property Damage and Fraud Cases**

In case of property damage, majority of the fraud cases has not been reported. However, maximum fraud cases didn't affected the property and didn't let to the property damage.

```
property_damage  fraud_reported
NO               N                272
                 Y                 66
YES              N                224
                 Y                 78
Name: fraud_reported, dtype: int64
```

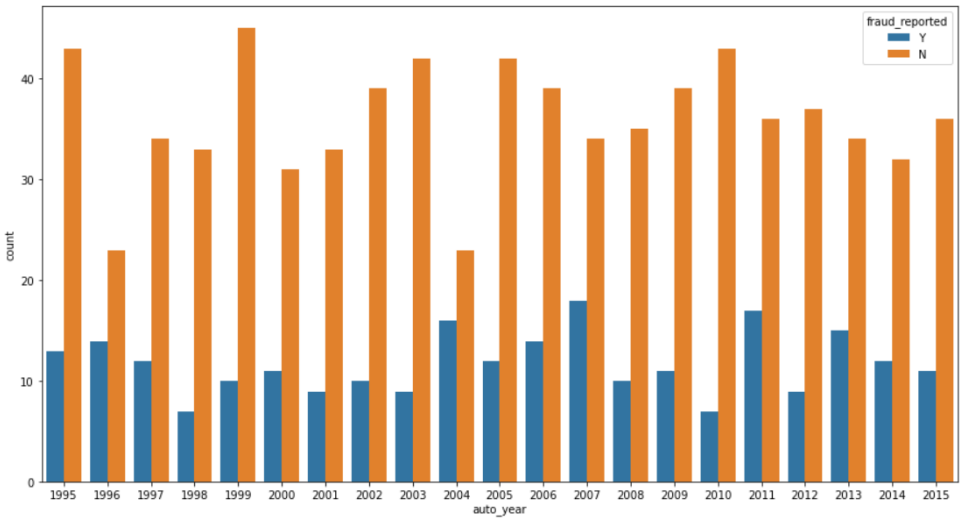
- **Auto model and Fruad Reported**

In aggregate, somewhere all the auto models has significantly been included in the frauds cases reported. Maximum number of frauds reported are with 3 Series auto model.



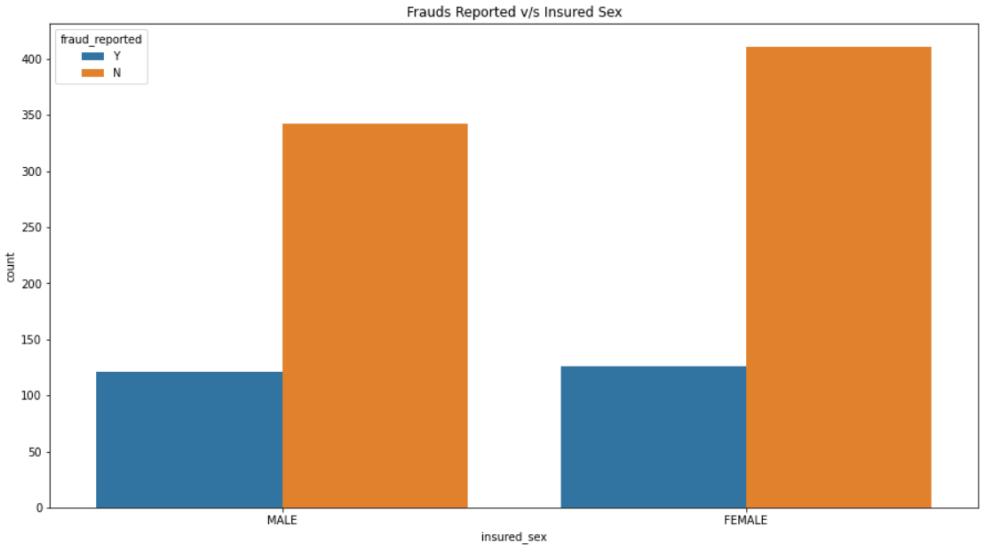
- **Auto year v/s fraud reported**

Maximum number of the fraud cases has been reported in the year of 2006 and 2010. Very less number of fraud cases is recorded in the year 1998. In the year 1999, majority of the fraud cases are not reported.



- **Insured Sex**

Females have taken the auto insurance more comparitvely to the men. 537 women and 463 men has taken the auto insurance. As per the data, females have not recorded the insurance fraud cases.



3. EDA Concluding Remarks

- Initailly there were missing values in the dataset but that has been taken care of with respect to the variable that holds significance with the target variable. All features in the dataset are of the correct data type and contain no missing or incorrect values.
- Generally people who have opted for the insurance, they are within the age group of 27 to 42.
- The strongest positive correlations with the target features are: Month as customer and age.
- All the other independent variables holds average correlation with the target variable.
- The dataset is imbalanced with the majority of observations with respect to the target variable.
- Females have taken the auto insurance more comparitvely to the men.
- Maximum number of the fraud cases has been reported in the year of 2006 and 2010.
- The maximum number of fraud cases which were reported is with respect to the rear collision.
- Maximum number of time the insurance fraud cases are reported is with the first two, that is, “Multi-vechile Collision” and “Single Vechile Collison”.

4. Pre-processing Pipeline

This section examines the steps involved in pre-processing data before implementing Machine Learning algorithms. Initially we have dropped the independent variables that are not relevant to the target variable.

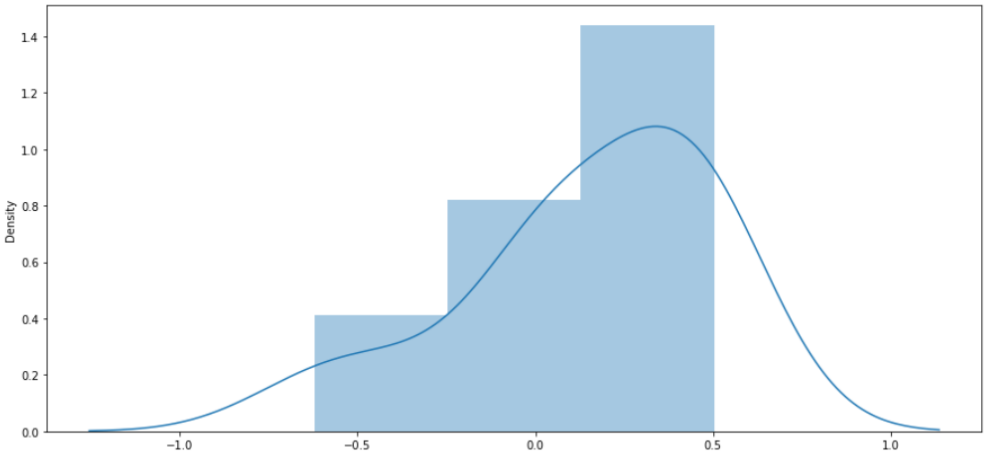
The target variable, fraud reported, had a fraud code of 1 and a non-fraud code of 0. There were six interaction words produced. interaction between the total claim amount and incident severity, policy yearly premium and total claim amount, umbrella limit and total claim amount, property claim amount and incident severity, vehicle claim amount and incident severity, injury claim amount and incident severity, etc.

We have divided the data in independent and dependent variable - X and Y. To clean the data, initially we have divided the data in categorical and continuous variables.

```
In [73]: catg=[col for col in X.columns if X[col].dtypes=='object']
         cont=[col for col in X.columns if X[col].dtypes!='object']

In [75]: print(len(catg) + len(cont))
         32
```

Unbalanced data distribution is followed with the continous variables. Plus outliers do exsist as well. To remove the skewness and outliers from the continuous data distribution, quanltile method is followed. With testing the multicollinerity with Vraiance Inflation Factor, we have removed the factor “total_claim_amount”.

Later, transformation and standarisation is done to process the data. Power transformation method is implemented on X data distribution.



Further moving ahead, label eoncding is done to transform the categorical variables into integer to carry forward the training on the data.

```
In [94]: ord=['umbrella_limit','insured_education_level','insured_occupation']
         for i in ord:
             X[i]=le.fit_transform(X[i])

In [95]: ### The rest of the categorical variables are nominal in nature.
         X=pd.get_dummies(X,drop_first=True)

In [96]: print(X.shape  , Y.shape)
         (1000, 124) (1000,)
```

By the end of pre processing procedure, SMOTE technique is implemented to balance the data. We got the equal distribution in the target variable for smooth training of the model.

```
In [97]: sm=SMOTE()
         x,y=sm.fit_resample(X,Y)

In [98]: print(x.shape , y.shape)
         (1506, 124) (1506,)

In [99]: y.value_counts(normalize=True)

Out[99]: 1    0.5
         0    0.5
         Name: Target, dtype: float64
```

The data set was divided into a train and test set with stratification based on reported fraud, with nominal variables one-hot encoded.

5. Building Machine Learning Models

In order to implement or apply Machine Learning algorithms, we must decouple our training dataset from our master dataset before implementing any algorithm.

- **Baseline algorithm**

Before moving on to more complex solutions, let's first employ a variety of baseline methods (using standard hyper-parameters). These algorithms: LogisticRegression, RandomForest, SVM, KNN, DecisionTreeClassifier, and Gaussian NB are taken into consideration in this section.

```
In [100]: x_train,x_test,y_train,y_test= train_test_split(x,y,random_state=8,test_size=.3)

In [101]: LR_model= LogisticRegression()
          RD_model= RidgeClassifier()
          DT_model= DecisionTreeClassifier()
          SV_model= SVC()
          KNR_model= KNeighborsClassifier()
          RFR_model= RandomForestClassifier()
          SGH_model= SGDClassifier()
          Bag_model=BaggingClassifier()
          ADA_model=AdaBoostClassifier()
          GB_model= GradientBoostingClassifier()

          model=[LR_model,RD_model,DT_model,SV_model,KNR_model,RFR_model,SGH_model,Bag_model,ADA_model,GB_model ]
```

Lets evolve accuracy score with respect to the model instance made. We got the highest accuracy score with the Logistic Regression.

```
In [103]: pd.DataFrame({'Model':model,'Accuracy':accuracy})
```

Out[103]:

	Model	Accuracy
0	LogisticRegression()	92.26
1	RidgeClassifier()	92.04
2	DecisionTreeClassifier()	86.50
3	SVC()	89.16
4	KNeighborsClassifier()	48.23
5	(DecisionTreeClassifier(max_features='sqrt', r...	90.93
6	SGDClassifier()	90.93
7	(DecisionTreeClassifier(random_state=134628861...	91.15
8	(DecisionTreeClassifier(max_depth=1, random_st...	91.15
9	([DecisionTreeRegressor(criterion='friedman_ms...	91.37

- **Cross Validation**

After implementing the cross validation, we got the following results.

In [106]:
```
ac=[]
cros=[]
dif=[]
for i in model:
    ac.append(accuracy_score(y_test,i.predict(x_test))*100)
    cros.append(cross_val_score(i,x,y,cv=5, scoring='accuracy').mean()*100)
    dif.append((accuracy_score(y_test,i.predict(x_test))*100)- (cross_val_score(i,x,y,cv=5, scoring='accuracy').mean()*100))

pd.DataFrame({'Model':model,'Accuracy':ac,'Cross Validation':cros,'Difference':dif})
```

Out[106]:

	Model	Accuracy	Cross Validation	Difference
0	LogisticRegression()	92.256637	85.670282	6.586355
1	RidgeClassifier()	92.035398	84.344459	7.690939
2	DecisionTreeClassifier()	86.504425	83.472751	2.964129
3	SVC()	89.159292	83.351961	5.807331
4	KNeighborsClassifier()	48.230088	55.512750	-7.282661
5	(DecisionTreeClassifier(max_features='sqrt', r...	90.929204	86.265869	5.590487
6	SGDClassifier()	90.929204	85.668742	6.256919
7	(DecisionTreeClassifier(random_state=134628861...	91.150442	86.527909	4.026727
8	(DecisionTreeClassifier(max_depth=1, random_st...	91.150442	84.742030	6.408413
9	([DecisionTreeRegressor(criterion='friedman_ms...	91.371681	86.129238	5.242223

- **Hyperparameter tuning**

All of the models' hyperparameters were tuned and chosen using GridSearchCV. Gridsearch is a effective option when compared to randomised search because of the quantity of parameters and models that were run specifically in this case. The model with its chosen hyperparameters was fitted to the training set after a 10-fold GridSearchCV.

It was calculated the mean accuracy scores for the top GridSearchCV estimators, accuracy scores on the training set, and accuracy scores on the test set. Then, the ROC AUC scores, sensitivity, specificity, and accuracy were calculated.

In [110]:
```
from sklearn.model_selection import GridSearchCV
params= {"learning_rate"    : [0.01,.05,.1,.2,.3,.5 ] ,
         'n_estimators':[5,50,100,200,300,400],
        "max_depth"        : [ 3, 4, 5, 6, 8]
         }

GCV= GridSearchCV(GB_model,params,cv=5,scoring='accuracy', n_jobs=-1)
GCV.fit(x_train,y_train)
```

Out[110]:

GridSearchCV
estimator: GradientBoostingClassifier
GradientBoostingClassifier

In [111]:
```
GCV.best_estimator_
```

Out[111]:

GradientBoostingClassifier
GradientBoostingClassifier(learning_rate=0.3, max_depth=8, n_estimators=200)

In [112]:
```
pred=GCV.best_estimator_.predict(x_test)
accuracy_score(y_test,pred)
```

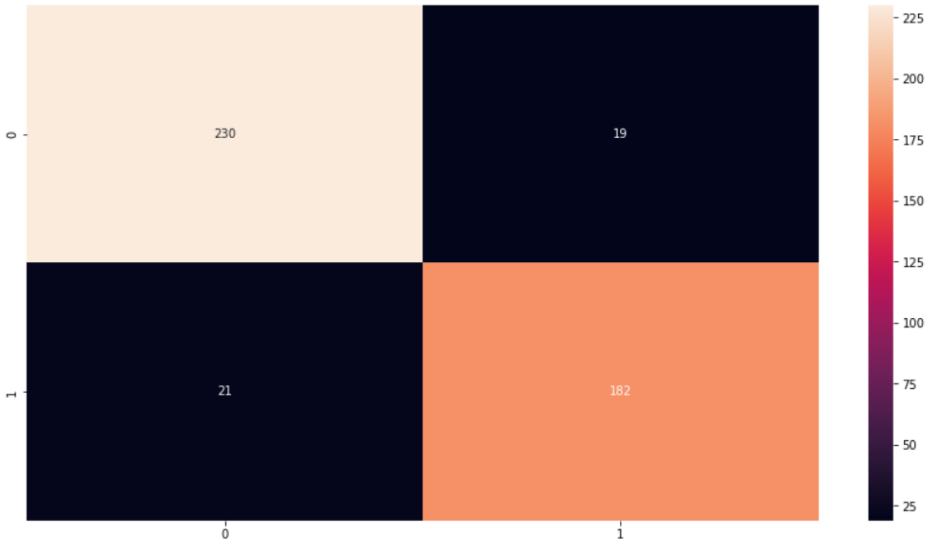
Out[112]:
```
0.911504424778761
```

We got the accuracy score with hyperparameter tuning is 0.91. It is comparatively a better accuracy score.

- **Confusion Matrix**

The model scored 0.91 for test accuracy. High accuracy scores indicate less prejudice (it is only a hint as accuracy is not a good measure of bias in imbalance class problems). A 0.123 difference in accuracy between the train and test is not very significant. This model thus has a low variance and is generalizable to data that have not yet been seen.

The confusion matrix below displays the number of cases for each class in the test set. The x-axis displays the anticipated classes, while the y-axis displays the actual classes.
Each quadrant displays the percentage of the test set's overall sample size.

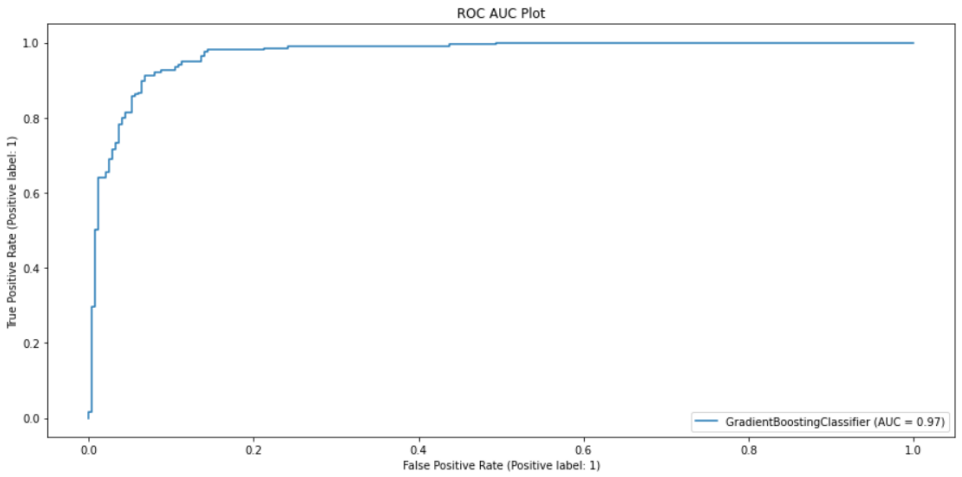


- **ROC Graph**

How successfully our model balances the genuine positive rate and the false positive rate is seen by the ROC curve below. The ideal situation is to have a 100% true positive rate of fraud detection and a 100% true negative rate of non-fraud detection (or, alternatively, a 0% false positive rate, which is 100% — 100% true negative rate). This indicates that our predictions are accurate for both courses. This is quite hard to do in the real world when there are issues with class imbalance. Additionally, a trade-off exists seen between true positive rate and the true negative rate, as well as the false positive rate's opposite.

This graph displays our ability to discriminate between two groups at each true positive and false positive rate level. This measure uses the area under the curve as a summary percentage. In conclusion, the model has significantly exceeded the baseline ROC AUC values.

The final AUC score we got is 98%. Hence model is a great fit for the distribution.



Although our algorithm was superior at predicting non-fraud situations, it also did a great job with fraud cases. We experience more false alarms than fraudsters who manage to avoid detection. In our situation, it is preferable to uncover more frauds than to allow fraud instances to go undetected. This methodology has therefore been successful in identifying charges of fraud. We are also able to balance this out under this approach, unlike the basic model, which devotes excessive resources on investigations and degrades user experience. More fraud can be found, and we can balance this with accurate predictions of non-fraud situations.

6. Concluding Remarks

An auto insurance fraud detection model has been developed as part of this research. The model can lower losses for insurance firms in this way. The difficulty with machine learning for fraud detection is that fraudulent insurance claims are far less frequent than legitimate ones.

In this study, five different classifiers—logistic regression, K-nearest neighbours, random forest, SVM classifiers, and AdaBoost—were employed. With these five classifiers, four alternative approaches to addressing imbalance classes were tested: a model using class weighting and hyperparameter adjustment, oversampling with SMOTE, and bootstrapping oversampling. Two max vote ensembles were developed on top of that. One each on the train set and the train set that was self-funded. Additionally, two blending ensembles were presented. One each on the train set and the train set that was bootstrapped.

The weighted Logistic Regression model, which produced a ROC AUC of 0.84, was the best and most successfully fitted model. The model outperformed the ROC AUC objective of 0.97 by a wide margin. The model has the greatest ROC AUC when compared to the other models. In conclusion, the model proved highly accurate in differentiating between legitimate and fraudulent claims.