Fraudulent Claim Analysis: Case Study

Poorna Ramakrishnan

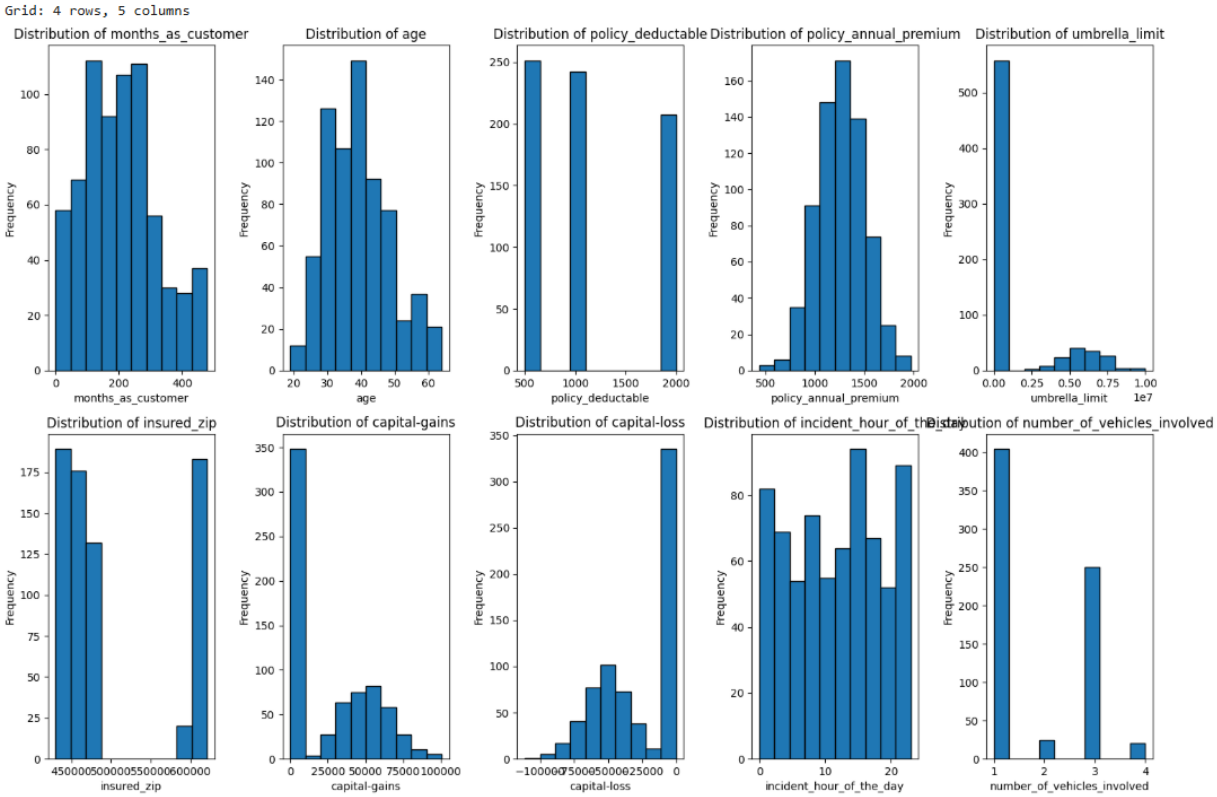
Prasita Shetty

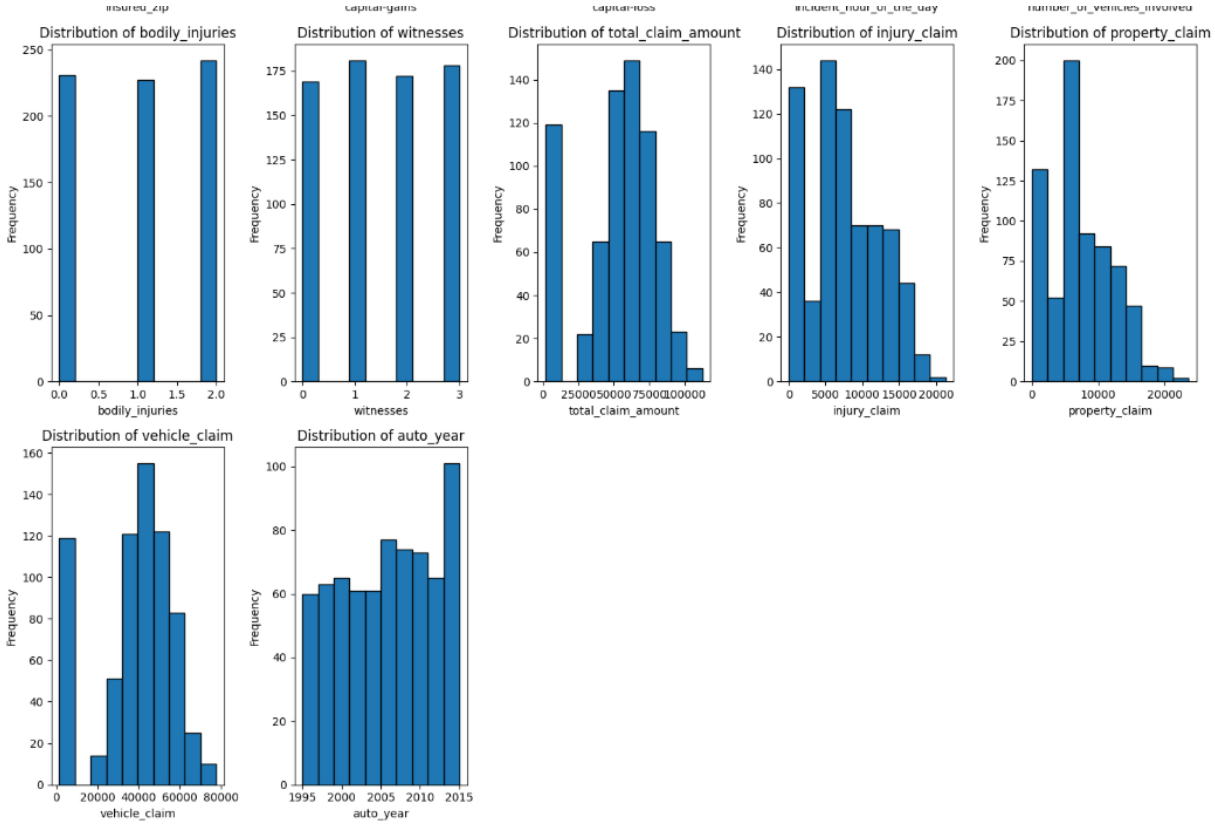
Problem Statement

Global Insure, a leading insurance company, processes thousands of claims annually. However, a significant percentage of these claims turn out to be fraudulent, resulting in considerable financial losses. The company's current process for identifying fraudulent claims involves manual inspections, which is time-consuming and inefficient. Fraudulent claims are often detected too late in the process, after the company has already paid out significant amounts. Global Insure wants to improve its fraud detection process using data-driven insights to classify claims as fraudulent or legitimate early in the approval process. This would minimise financial losses and optimise the overall claims handling process.

Exploratory Data analysis charts:

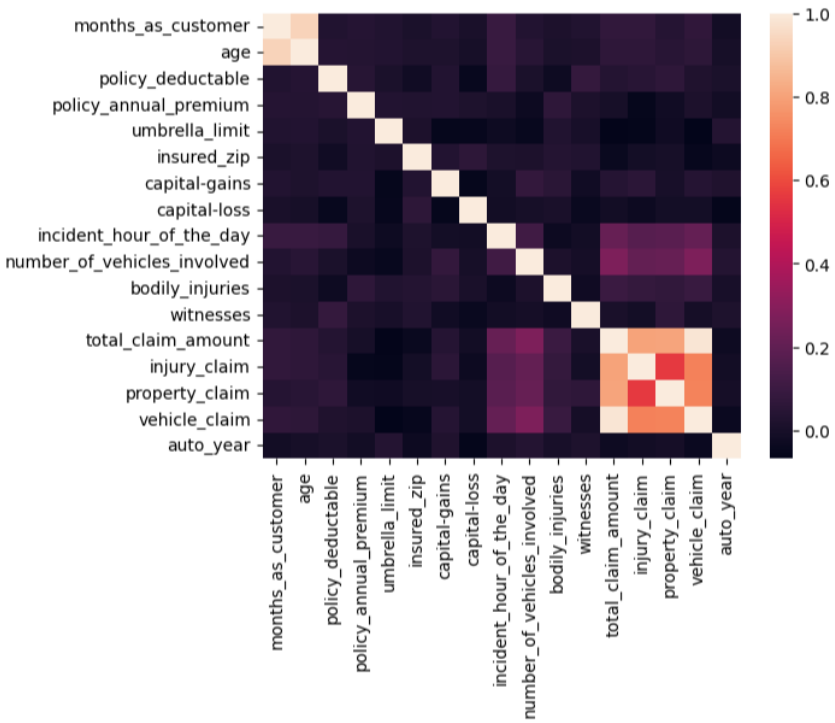
Univariate Analysis on Numerical columns





Correlation Heatmap of numerical features

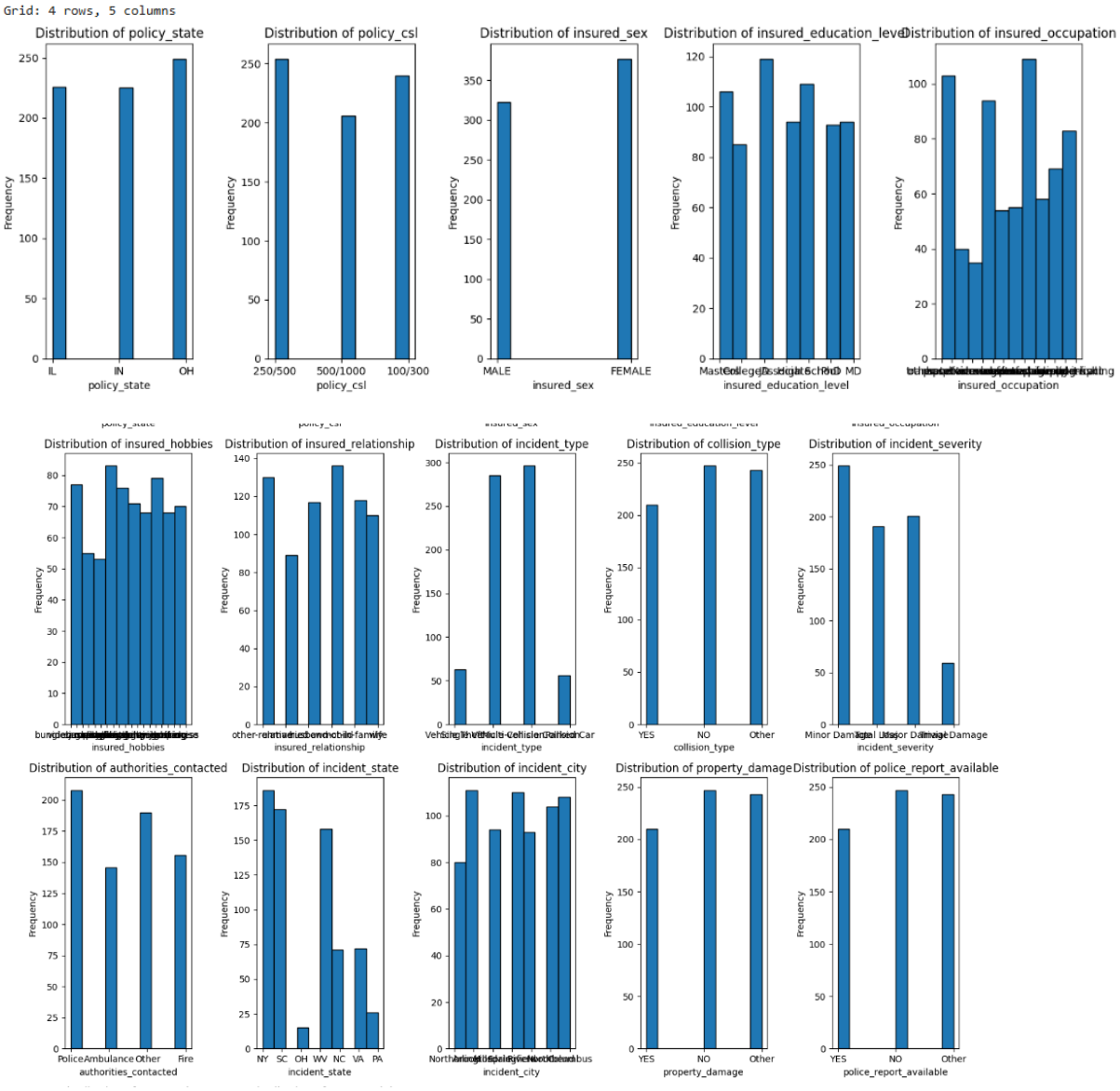
<Axes: >

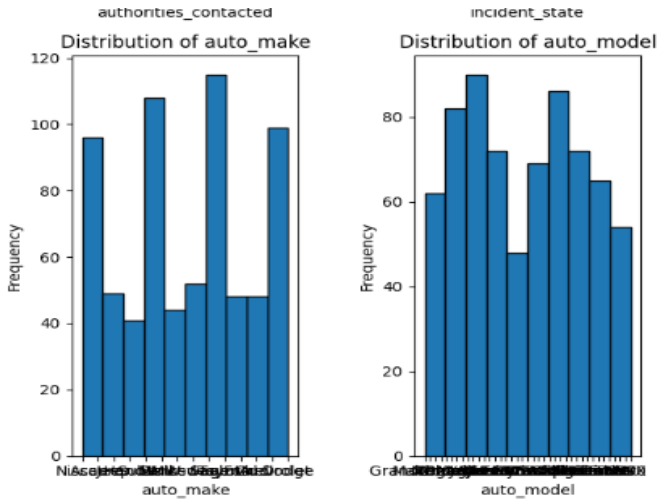


Variables with correlation > 0.6 (threshold = 0.6)

	Variable 1	Variable 2	Correlation
0	months_as_customer	age	0.933173
1	total_claim_amount	injury_claim	0.803185
2	total_claim_amount	property_claim	0.805547
3	total_claim_amount	vehicle_claim	0.981623
4	injury_claim	vehicle_claim	0.717862
5	property_claim	vehicle_claim	0.721810

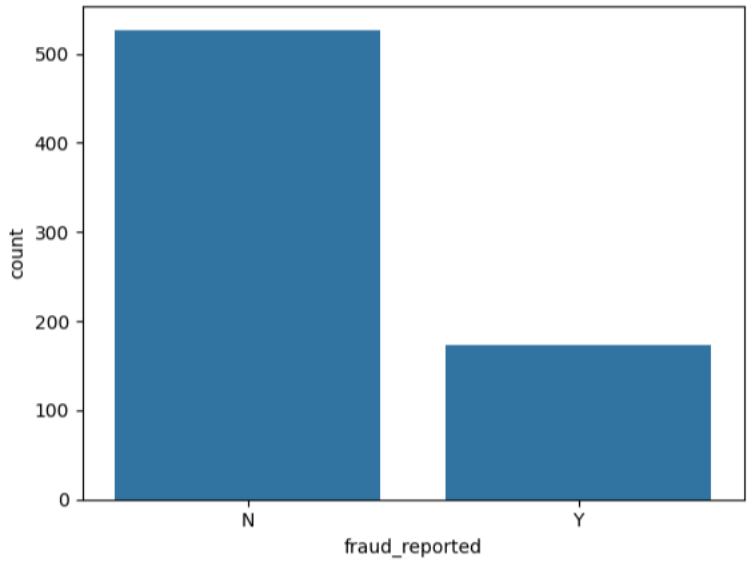
Univariate analysis of Categorical Variables





Class balance in train set

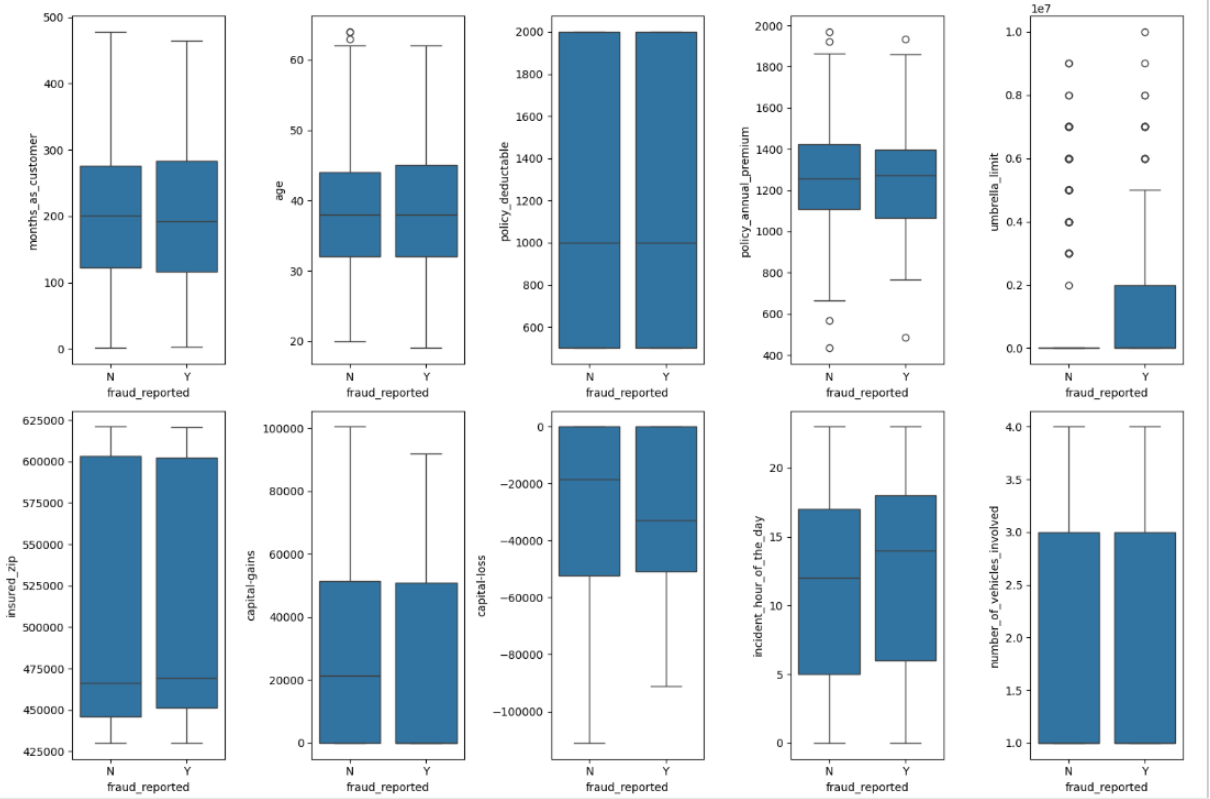
```
fraud_reported
N    75.285714
Y    24.714286
Name: count, dtype: float64
```

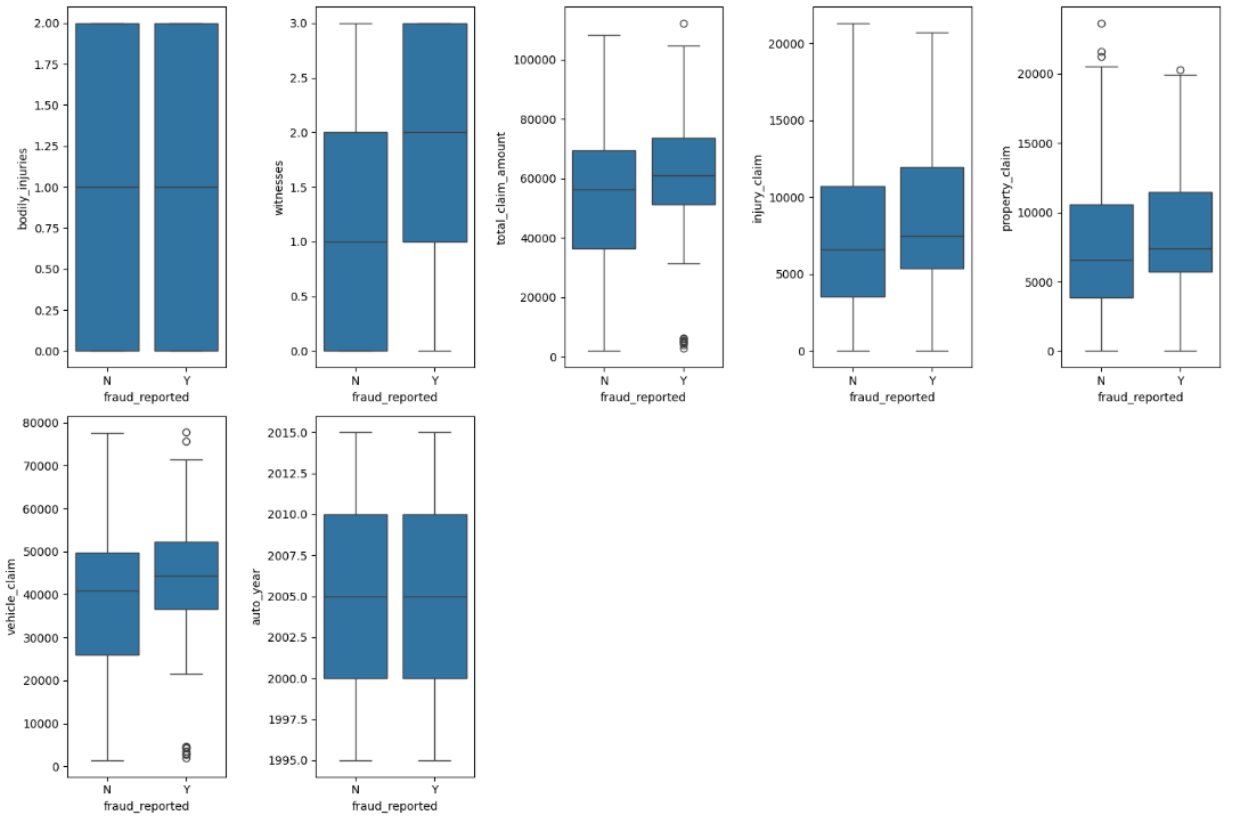


Bivariate Analysis

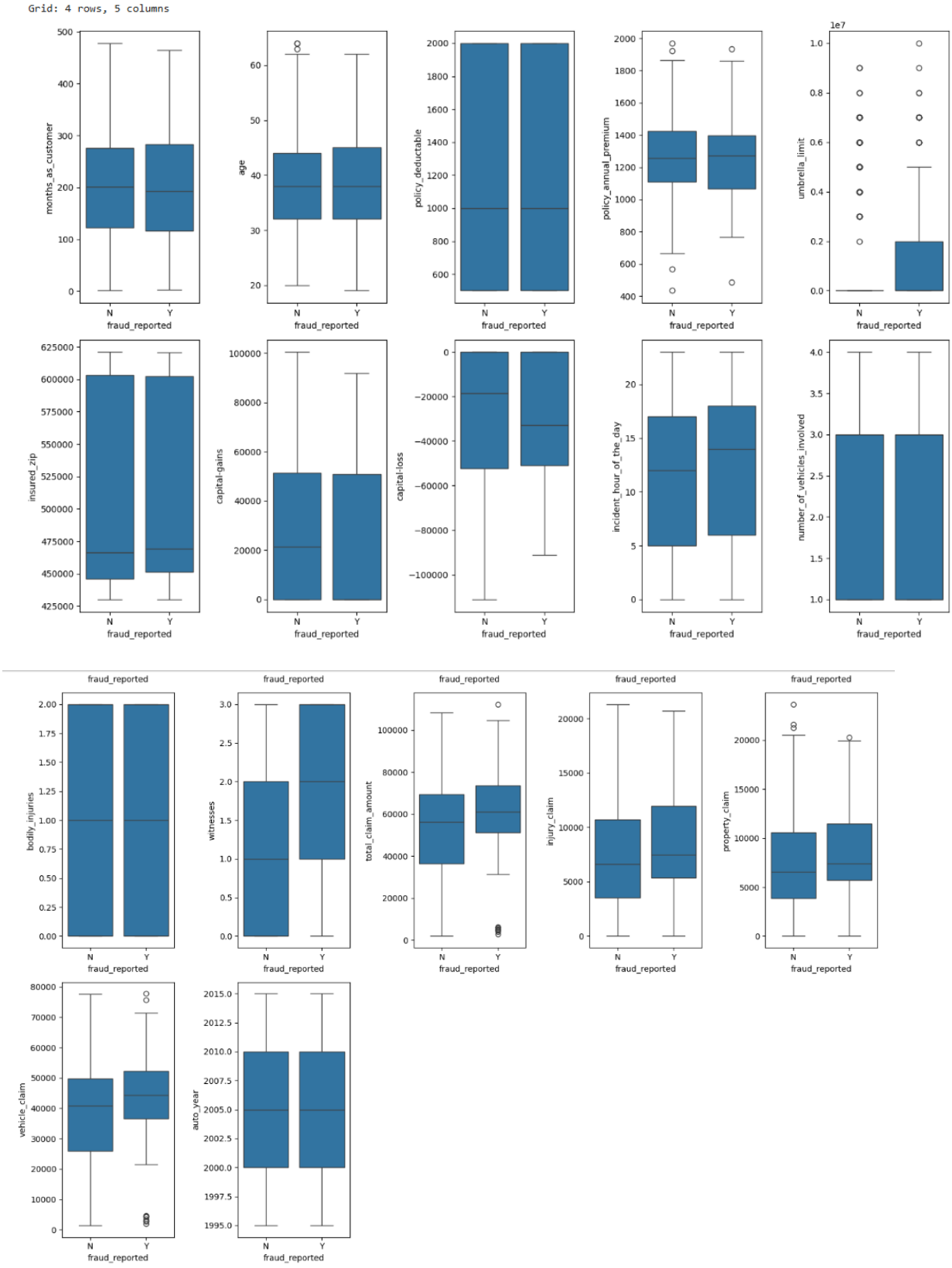
Numerical variables vs Target variable

Grid: 4 rows, 5 columns

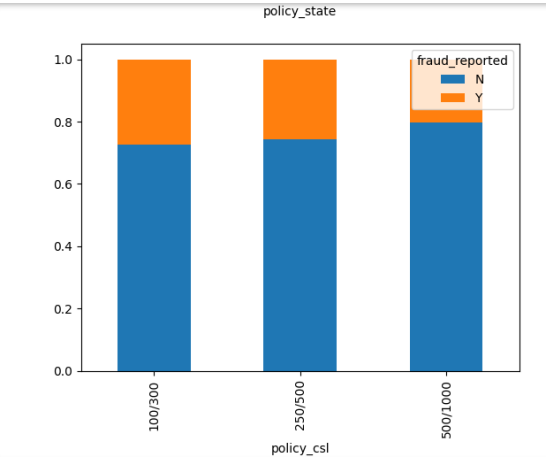


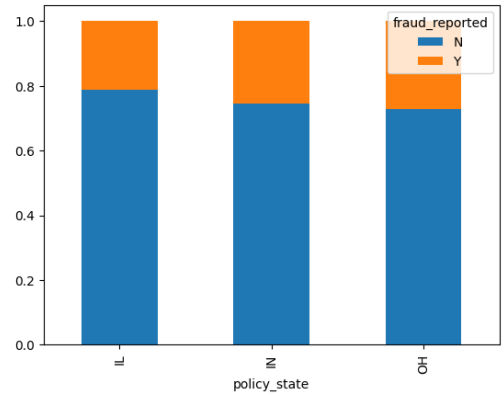


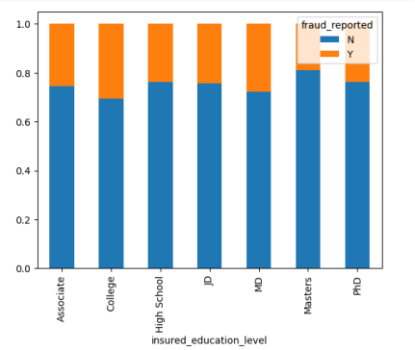
Categorical variables vs Target variable

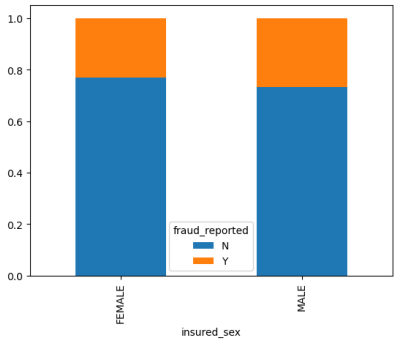


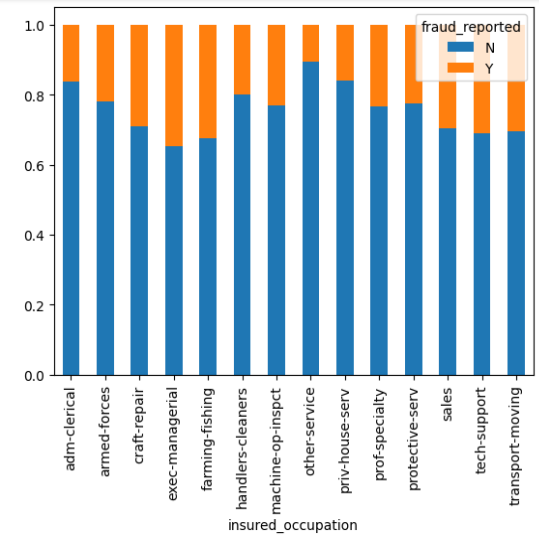
Categorical variables vs Target Variable

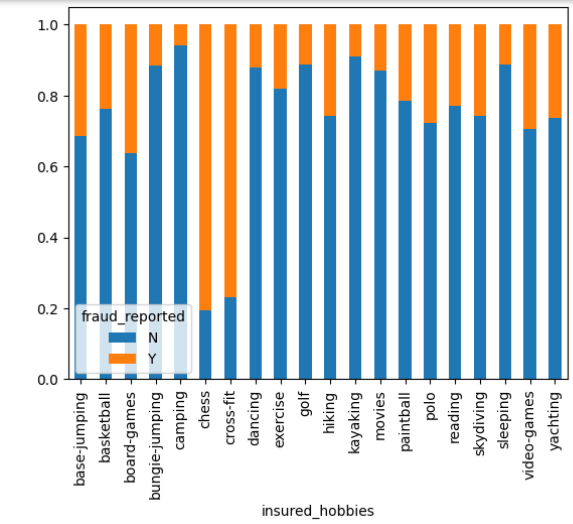


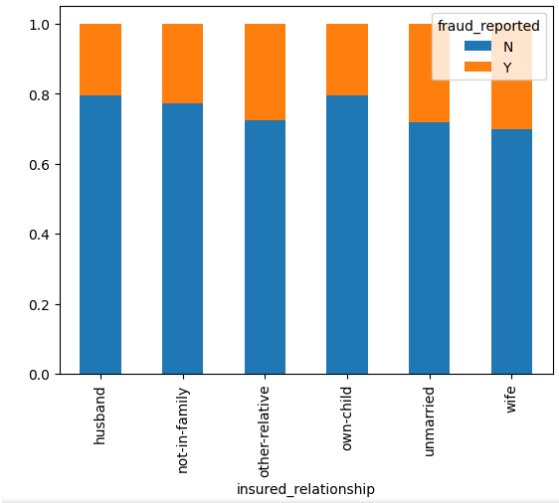


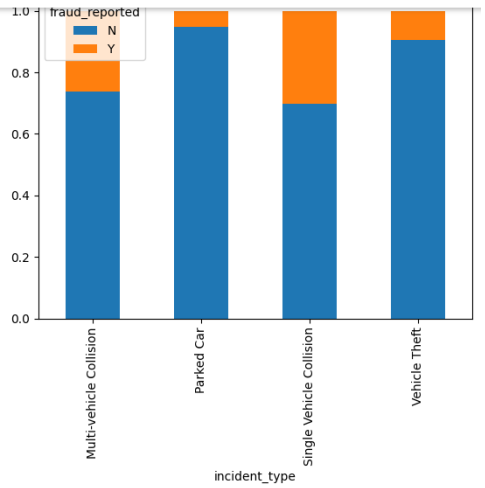


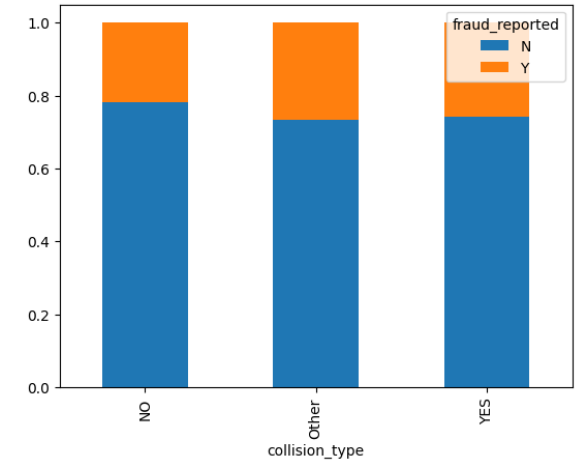


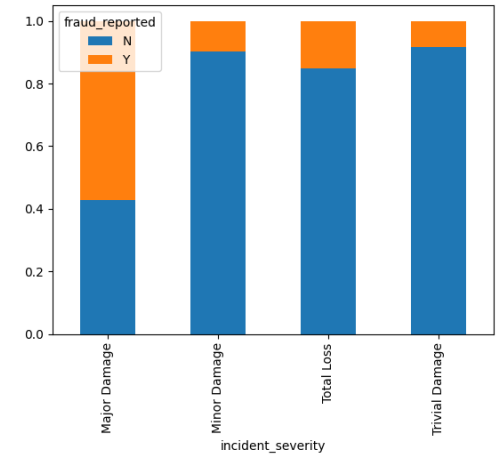


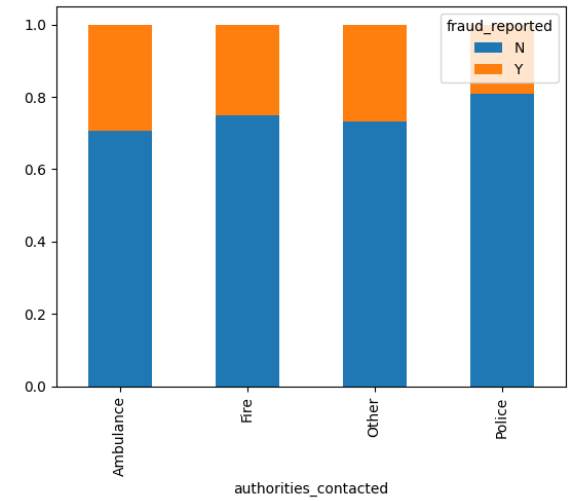


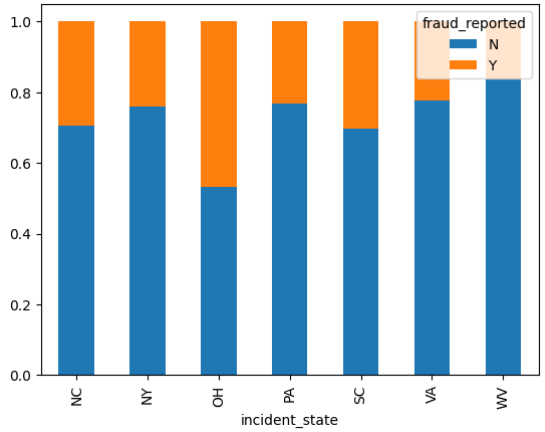

fraud_reported
N
Y
1.0
0.8
0.6
0.4
0.2
0.0
husband
not-in-family
other-relative
own-child
unmarried
wife
insured_relationship

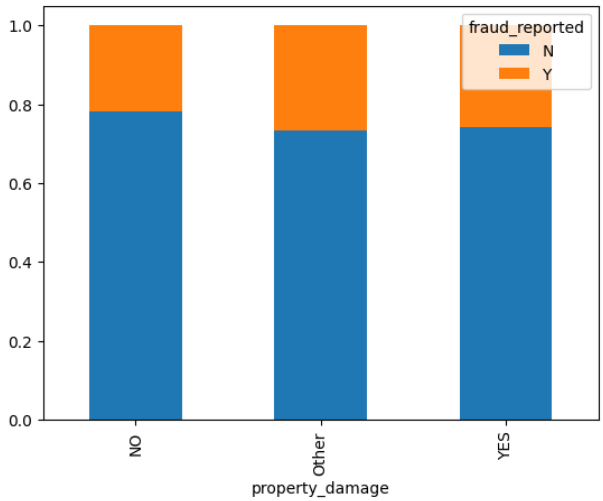

fraud_reported
N
Y
1.0
0.8
0.6
0.4
0.2
0.0
Multi-vehicle Collision
Parked Car
Single Vehicle Collision
Vehicle Theft
incident_type

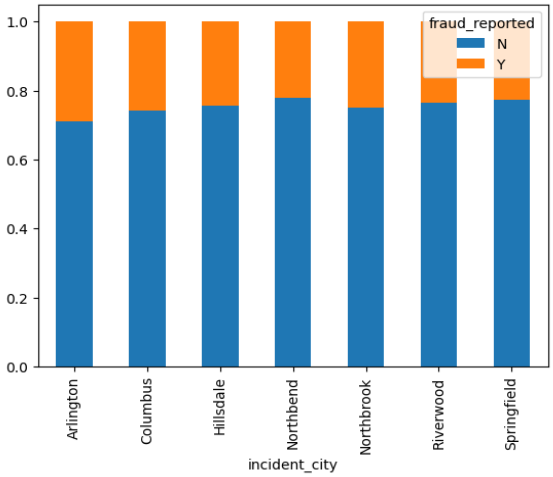

fraud_reported
N
Y
1.0
0.8
0.6
0.4
0.2
0.0
NO
Other
YES
collision_type

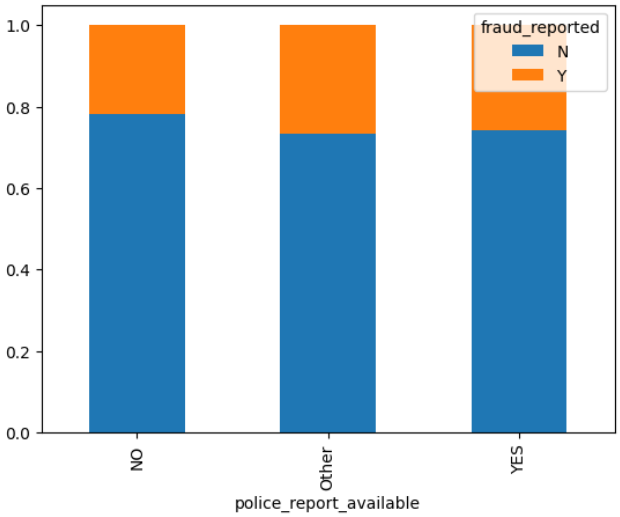

fraud_reported
N
Y
1.0
0.8
0.6
0.4
0.2
0.0
Major Damage
Minor Damage
Total Loss
Trivial Damage
incident_severity

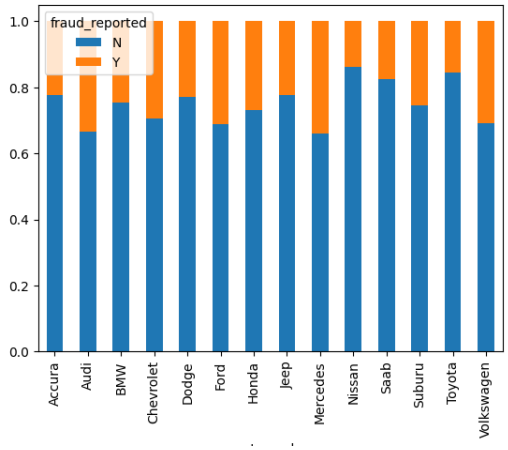

fraud_reported
N
Y
1.0
0.8
0.6
0.4
0.2
0.0
Ambulance
Fire
Other
Police
authorities_contacted

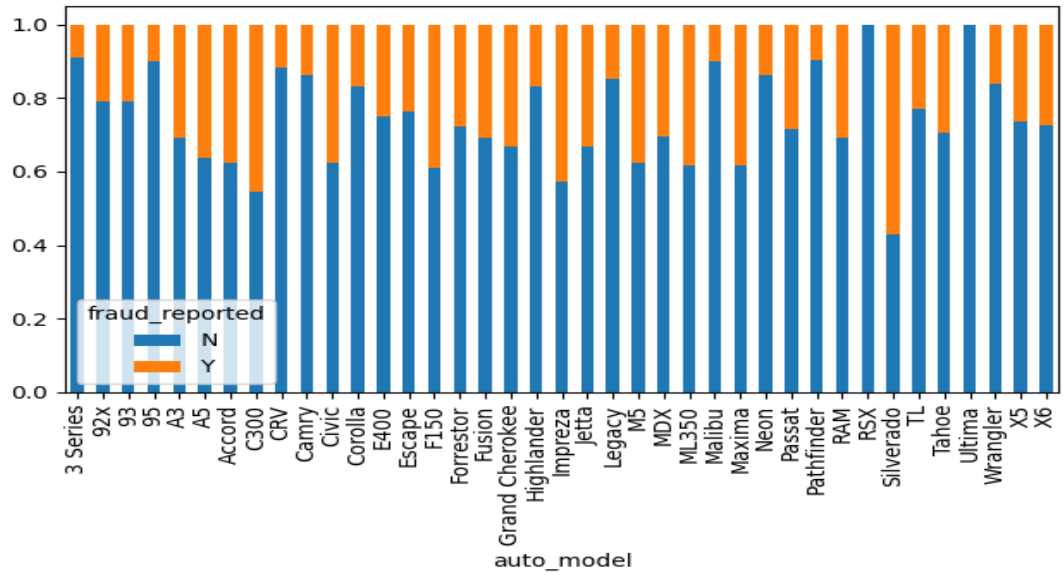

fraud_reported
N
Y
1.0
0.8
0.6
0.4
0.2
0.0
NC
NY
OH
PA
SC
VA
WV
incident_state


fraud_reported
N
Y
1.0
0.8
0.6
0.4
0.2
0.0
NO
Other
YES
property_damage


fraud_reported
N
Y
1.0
0.8
0.6
0.4
0.2
0.0
Arlington
Columbus
Hillsdale
Northbend
Northbrook
Riverwood
Springfield
incident_city


fraud_reported
N
Y
1.0
0.8
0.6
0.4
0.2
0.0
NO
Other
YES
police_report_available

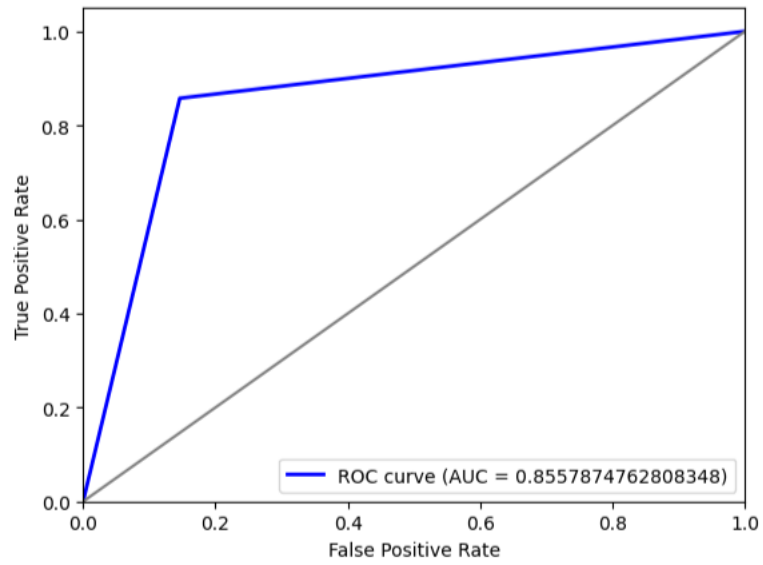

fraud_reported
N
Y
1.0
0.8
0.6
0.4
0.2
0.0
Accura
Audi
BMW
Chevrolet
Dodge
Ford
Honda
Jeep
Mercedes
Nissan
Saab
Suburu
Toyota
Volkswagen


fraud_reported
N
Y
1.0
0.8
0.6
0.4
0.2
0.0
3 Series
92x
93
95
A3
A5
Accord
C300
CRV
Camry
Civic
Corolla
E400
Escape
F150
Forrestor
Fusion
Grand Cherokee
Highlander
Impreza
Jetta
Legacy
M5
MDX
ML350
Malibu
Maxima
Neon
Passat
Pathfinder
RAM
RSX
Silverado
TL
Tahoe
Ultima
Wrangler
X5
X6
auto_model

Model Building

Logistic Regression

ROC Curve



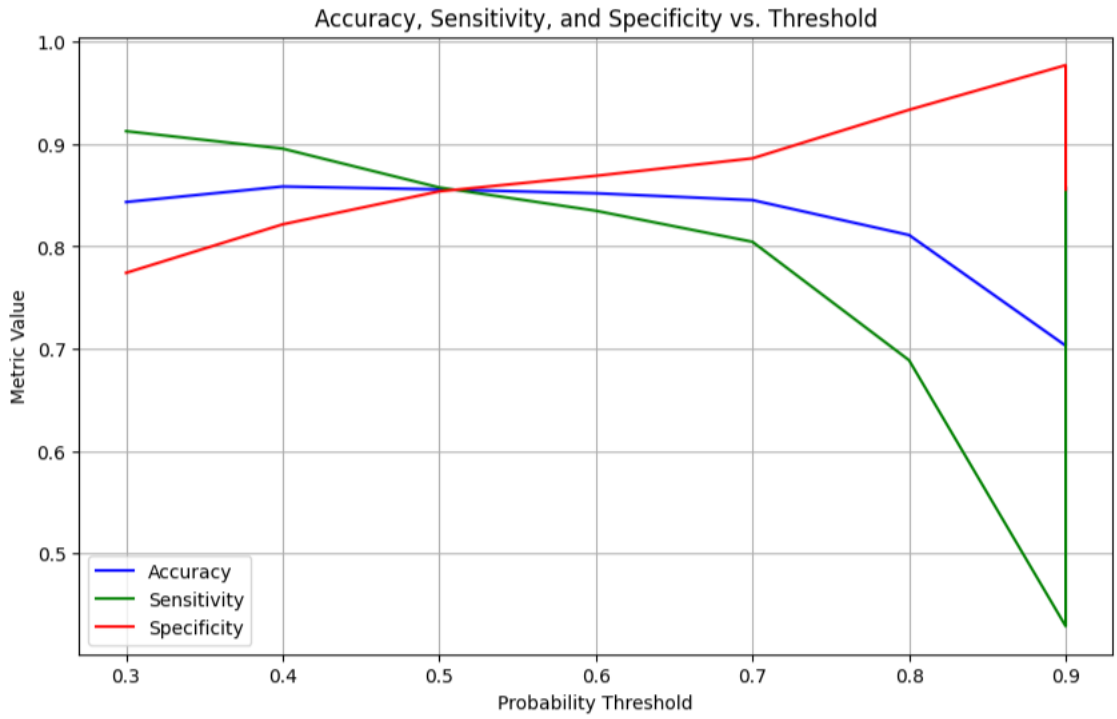
cutoff	accuracy	sensitivity	specificity
0.30	0.843454	0.912713	0.774194
0.40	0.858634	0.895636	0.821632
0.50	0.855787	0.857685	0.853890
0.60	0.851992	0.834915	0.869070
0.70	0.845351	0.804554	0.886148
0.80	0.811195	0.688805	0.933586
0.90	0.703036	0.428843	0.977230
0.52	0.856736	0.857685	0.855787
0.53	0.856736	0.857685	0.855787

Confusion Matrix

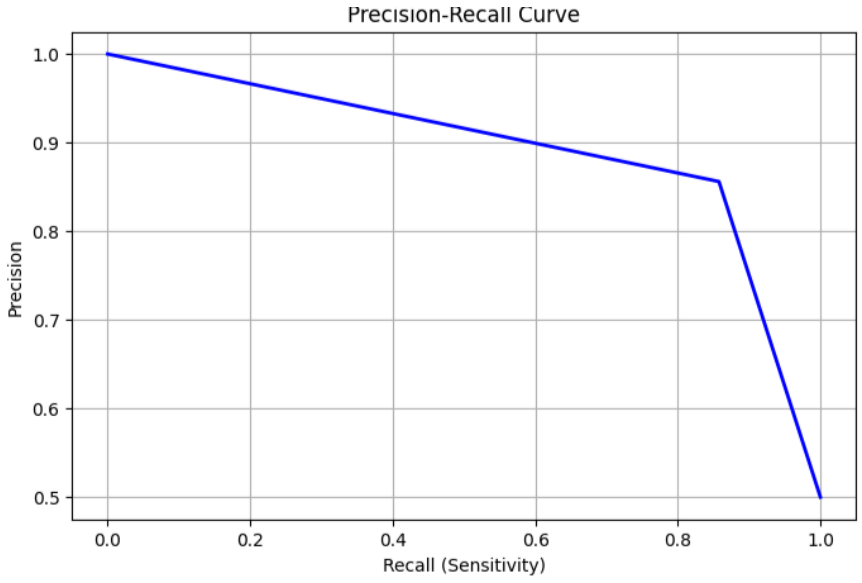
```
: # Create the confusion matrix once again
  cm=metrics.confusion_matrix(df_results_lr['Actual'],df_results_lr['Predict_prob_0.53_cutoff'])
  cm
```

```
: array([[451,  76],
         [ 75, 452]])
```

Identifying the sensitivity vs specificity threshold



Precision Recall Curve



Random Forest

Confusion Matrix

7.4.7 Create confusion matrix [1 Mark]

```
[364]: # Create the confusion matrix to visualise the performance
       cm=metrics.confusion_matrix(y_train_final,y_train_preds_rf)
       cm
```

```
[364]: array([[405, 122],
              [ 42, 485]])
```

```
sensitivity= 0.920303605313093
specificity= 0.7685009487666035
precision= 0.7990115321252059
recall= 0.920303605313093
f1_score= 0.855379188712522
```

Validation data

Logistic Regression

```
: # Check the accuracy
  metrics.accuracy_score(df_validation_lr['Actual'],df_validation_lr['Final_preds'])
```

```
: 0.8666666666666667
```

8.1.8 Calculate sensitivity, specificity, precision, recall and f1 score of the model [2 Marks]

```
: # Calculate the sensitivity
  sensitivity=TP/(TP+FN)  #Same as recall
  print("sensitivity=",sensitivity)
  # Calculate the specificity
  specificity=TN/(TN+FP)
  print("specificity=",specificity)
  # Calculate Precision
  precision=TP/(TP+FP)
  print("precision=",precision)
  # Calculate Recall
  recall=TP/(TP+FN)
  print("recall=",recall)
  # Calculate F1 Score
  f1_score=(2*precision*recall)/(precision+recall)
  print("f1_score=",f1_score)
```

```
sensitivity= 0.7972972972972973
specificity= 0.8893805309734514
precision= 0.7023809523809523
recall= 0.7972972972972973
f1_score= 0.7468354430379748
```

Random Forest

```
: # Check accuracy
  metrics.accuracy_score(y_test_final,preds_rf)
```

```
: 0.6733333333333333
```

```
sensitivity= 0.527027027027027
specificity= 0.7212389380530974
precision= 0.38235294117647056
recall= 0.527027027027027
f1_score= 0.4431818181818181
```

Methodology

- Data Cleaning was performed
  - Remove redundant columns, impute missing values, change the datatype of the variables as necessary
  - Multicollinearity was identified using heat maps and those variables were dropped
- Data was segmented into train and evaluation sets
  - A 70-30 split was used to segment the data into stratified train and test sets.
  - Class balance was 75-25. Hence, the train data was up sampled to ensure that the 2 target classes are balanced
- Numerical , categorical and date variables were analyzed separately
  - Numerical variables – Standardization technique was used to scale the numerical variables so that no variable is given more importance than the other. This is especially important for Logistic regression
  - Categorical variables- There were 2 type of categorical variables: Variables with less than 3 levels – these were dummy value encoded directly.
  - For the categorical variables , with many levels, the fraud distribution for each level of the variable was observed . Based on this , this, the similar values were combined thus decreasing the cardinality of the variable . Ex: Hobbies, Education , Type, Severity etc. Dummy variable encoding was used on the rolled-up values.
  - Derived variables/interaction variables were created where necessary
  - Month and year features were extracted from datetime variables
  - Capital Profile and loss variables were added for a particular candidate to get their overall profile/loss
  - Standardization was performed on the numerical data before modelling and its relationship with target variables observed

- Standard Scalar was performed on the numerical variables to scale the variables to bring them to the same range
- All the above transformations were fit only using the train set to avoid data leakage
- The transformed data was fed into 2 different algorithms – Logistic Regression and Random Forest
- The performance of the algorithm was evaluated basis the validation set and the final model was selected
- All the above transformations were fit only using the train set to avoid data leakage

Model Methodology

- The transformed data was fed into 2 different algorithms – Logistic Regression and Random Forest
- **Logistic Regression was first tried:**
- RFECV (Recursive feature elimination along with cross validation) was used to select the important features. A CV fold of 5 was used .
- P values of the selected variables were observed and VIF was also calculated to ensure there is no multi collinearity on the selected variables
- Predictions were calculated for different threshold values and then the best threshold value was selected based on the various metrics like accuracy, sensitivity and specificity
- ROC curve was plotted to understand AUC
- Accuracy, Sensitivity and specificity was also plotted across various threshold values to confirm the optimal threshold value visually
- Precision recall curve was also plotted to understand these values.
- **Random Forest model was then tried:**
- A random forest model was calculated with default hyperparameters. This model overfit on the data

- GridSearch CV was then used to get the best hyperparameters
- The selected hyperparameters were then used to construct the final model.
- Top features (whose threshold value >.04) were selected to build the model

Insights / Observations:

- GridSearchCV helps to reduce overfitting by choosing the best params and testing them over different sets of data using Cross validation techniques
- Logistic regression model had similar accuracy on train and test and did not overfit
- Random Forest default model overfit here and tuned model performed better on the train set but again overfit on the test set
- Therefore, logistic regression was chosen as the final model

Reasons for choosing Logistic regression as the final model

- Logistic regression model was selected as the best model due to its
  - Better performance on the evaluation set
  - Interpretability of the model coefficients and confidence basis p-values

Logistic Regression

accuracy=87%

sensitivity= 79%

specificity= 89%

precision= 70%

recall= 80%

f1_score= 75%

```
sensitivity= 0.7972972972972973
specificity= 0.8893805309734514
precision= 0.7023809523809523
recall= 0.7972972972972973
f1_score= 0.7468354430379748
```

	coef	std err	z	P>|z|	[0.025	0.975]
const	6.5914	0.431	15.298	0.000	5.747	7.436
authorities_contacted_Police	0.4765	0.219	2.178	0.029	0.048	0.905
incident_severity_updated_Total Loss	-2.8803	0.245	-11.760	0.000	-3.360	-2.400
incident_state_updated_Low	-0.6440	0.191	-3.380	0.001	-1.017	-0.271
auto_model_updated_Low	-0.7424	0.208	-3.571	0.000	-1.150	-0.335
incident_severity_updated_Other	-3.6191	0.245	-14.764	0.000	-4.100	-3.139
insured_hobbies_updated_Medium	-3.6625	0.355	-10.330	0.000	-4.357	-2.968
insured_occupation_updated_Low	-0.6273	0.186	-3.378	0.001	-0.991	-0.263

Random Forest

accuracy=68%

sensitivity= 53%

specificity= 73%

precision= 39%

recall= 53%

f1_score= 45%

	Feature	Importance
24	incident_severity_updated_Other	0.083143
43	total_claim_amount	0.067327
38	insured_zip	0.055175
36	policy_annual_premium	0.048481
34	months_as_customer	0.047573
44	Time_between_policy_bind_date_incident_date	0.047474
46	Total_profit_loss	0.044312
19	insured_hobbies_updated_Low	0.044279
39	incident_hour_of_the_day	0.042857

```
sensitivity= 0.527027027027027
specificity= 0.7300884955752213
precision= 0.39
recall= 0.527027027027027
f1_score= 0.4482758620689655
```