
INSURANCE
FRAUD

Content

- Data Preparation and Cleaning
- Train Validation Split 70-30
- EDA on Training Data
- Feature Engineering
- Model Building
- Predicting and Model Evaluation

Data Preparation and Cleaning

❑ Import and inspect dataset

- ➢ Shape of dataset : (1000,40)

```
<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
```

Data Preparation and Cleaning

```
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          91
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
```

Since nulls in authorities_contacted account for around 10% (91 rows), hence replaced them with 'Unknown' prevents potential data loss

```
authorities_contacted
Police       292
Fire         223
Other        198
Ambulance    196
Unknown       91
Name: count, dtype: int64
```

_c39 contains only null values, so it was dropped from the dataset.

Redundant features like policy_number, insured_zip, insured_hobbies, and incident_location were dropped from the dataset.

Data Preparation and Cleaning

policy_bind_date	object
incident_date	object

Corrected the data type from object to datetime64[ns]

policy_bind_date	datetime64[ns]
incident_date	datetime64[ns]

Train Validation Split

```
# Put all the feature variables in X
X = df.drop('fraud_reported', axis=1)
# Put the target variable in y
y=  df['fraud_reported']
```

Independent variable

Dependent variable

```
# Split the dataset into 70% train and 30% validation and use stratification on the target variable
X_train, X_test, y_train, y_test = train_test_split (X, y, train_size=0.7,random_state=42 )
# Reset index for all train and test sets
X_train.shape,y_train.shape,X_test.shape,y_test.shape
```

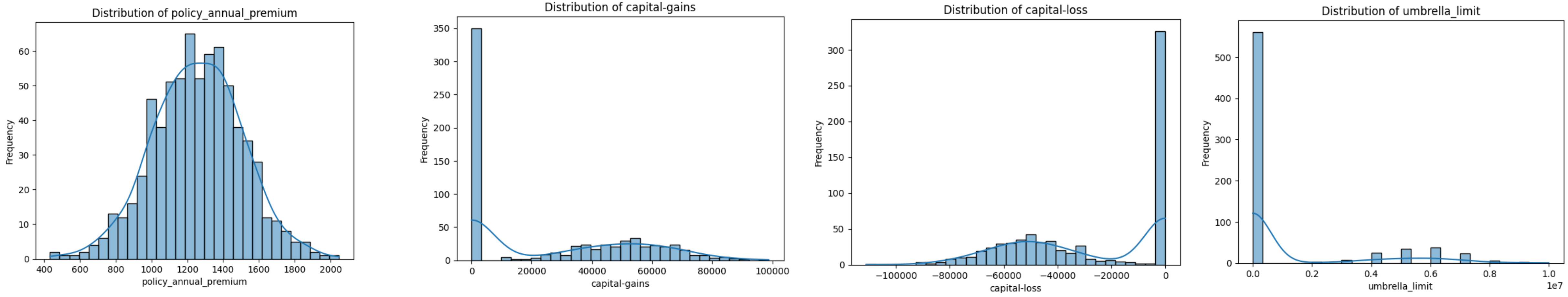
```
((700, 32), (700,), (300, 32), (300,))
```

Train size = 70%
Test size = 30%

EDA on Training Data

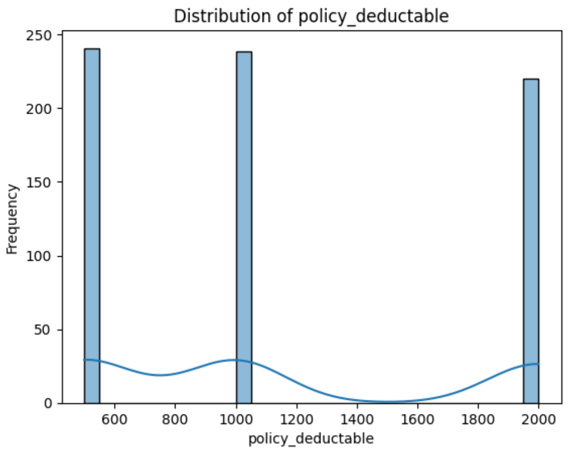
EDA on Numerical features

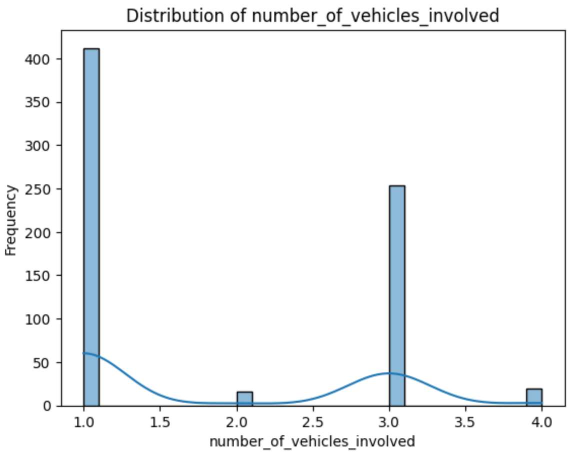
EDA on Training Data

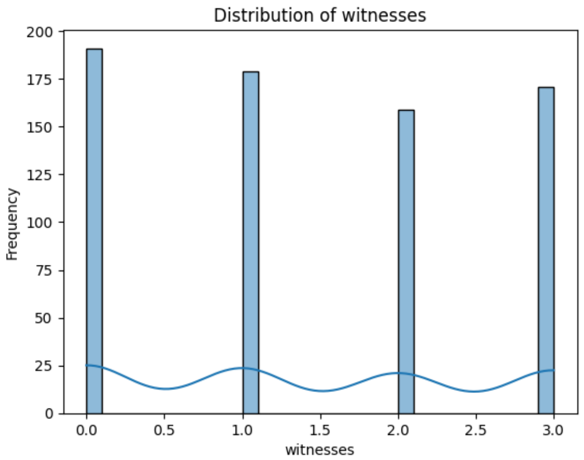


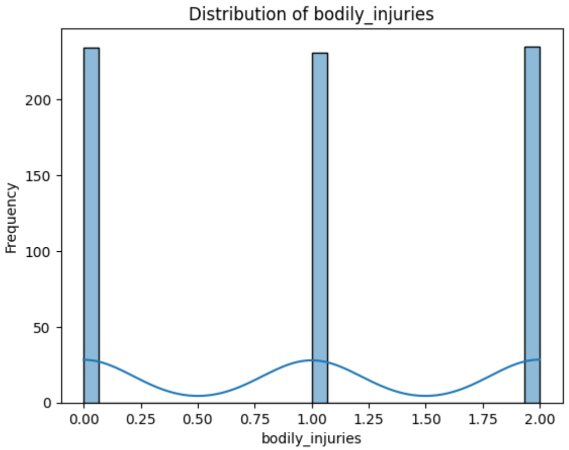
'umbrella_limit','capital-gains','capital-loss' are highly skewed toward zero, so they were deleted.

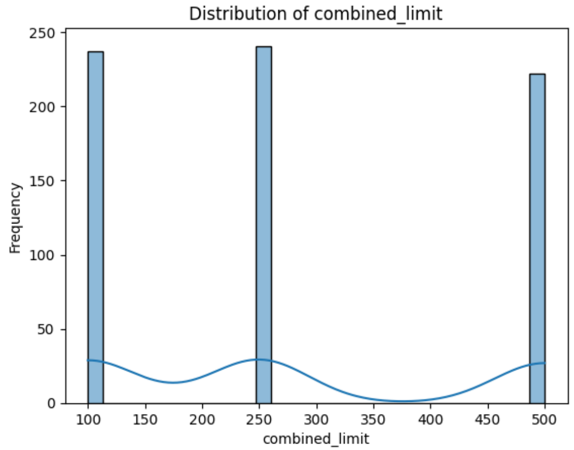
EDA on Training Data

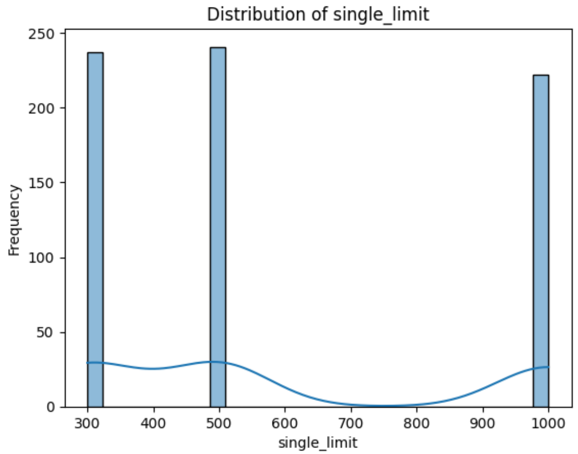






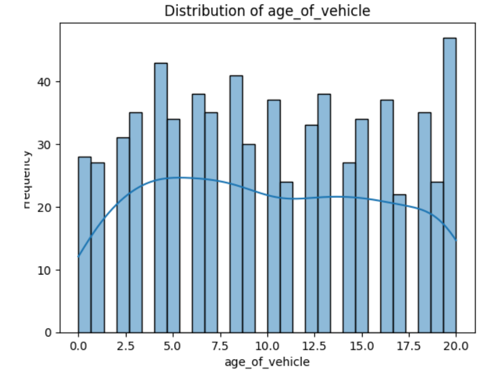


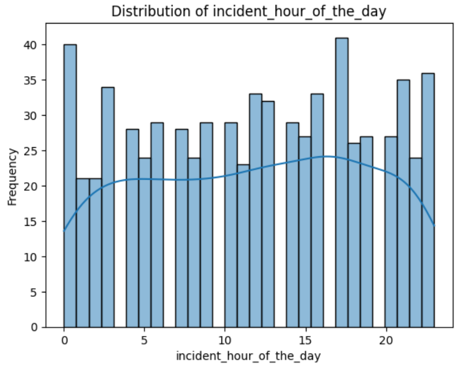


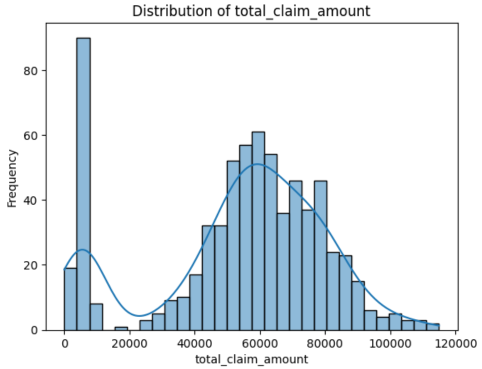


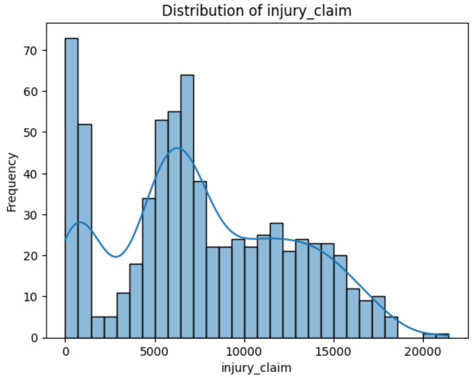
'policy_deductable', 'number_of_vehicles_involved', 'bodily_injuries' , 'witnesses', 'combined_limit' , 'single_limit' were initially marked as numerical, but since they exhibit categorical behavior, they were converted to object type.

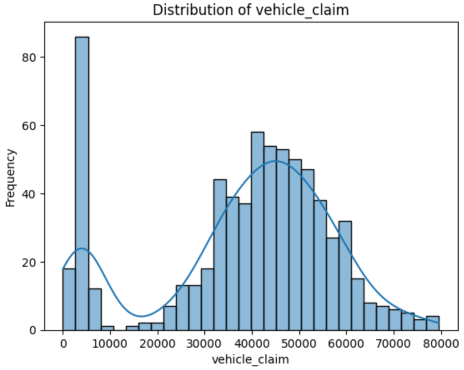
EDA on Training Data

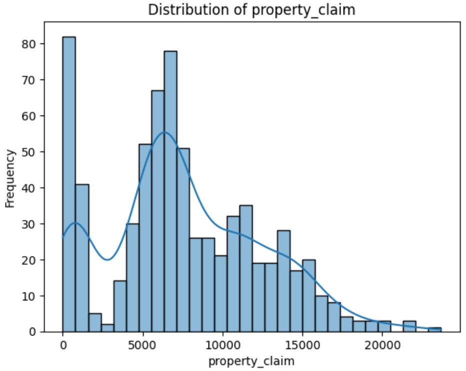


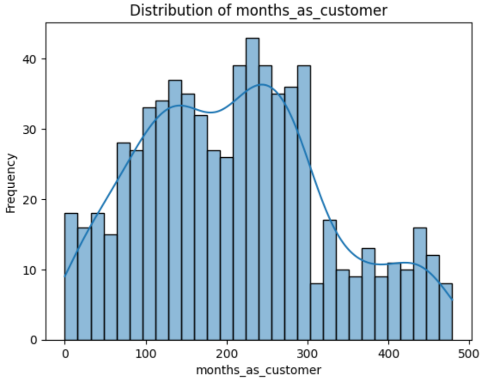


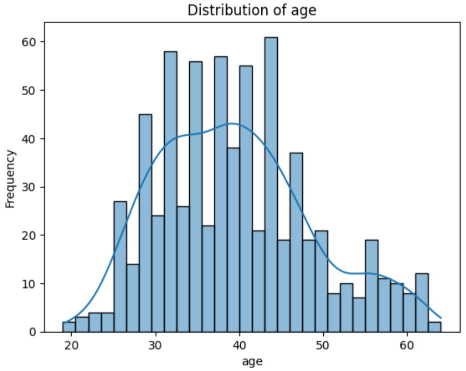










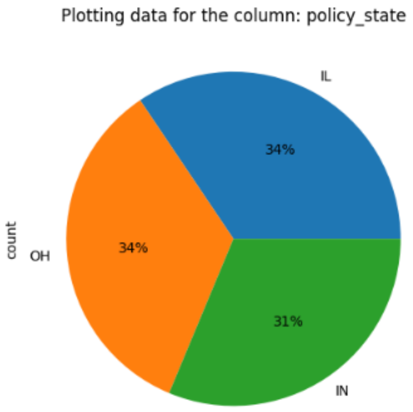


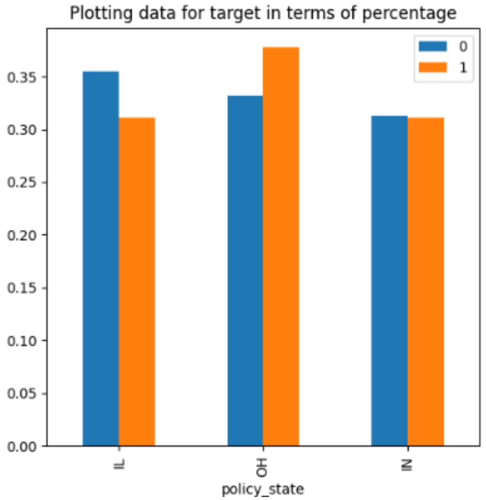
'incident_hour_of_the_day', 'total_claim_amount' , 'injury_claim', 'property_claim', 'age_of_vehicle', 'age' features are distributed across ranges, and to simplify analysis, these ranges have been grouped.

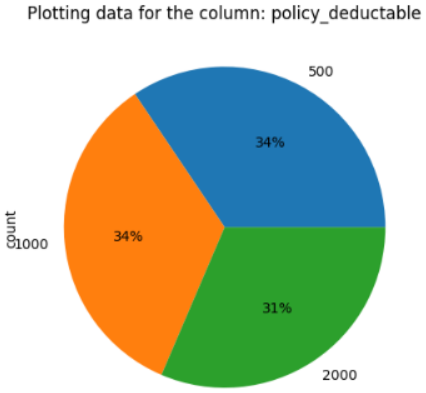
EDA on Training Data

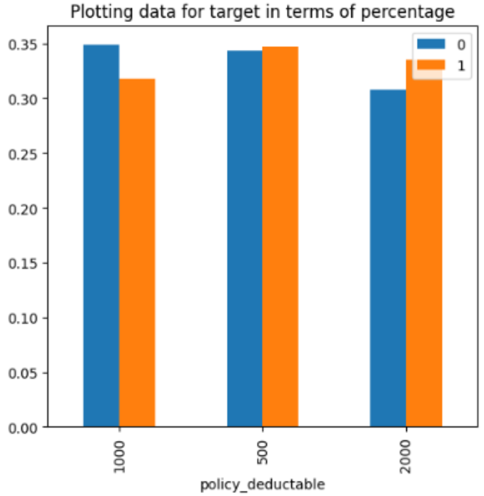
EDA on Categorical features

EDA on Training Data

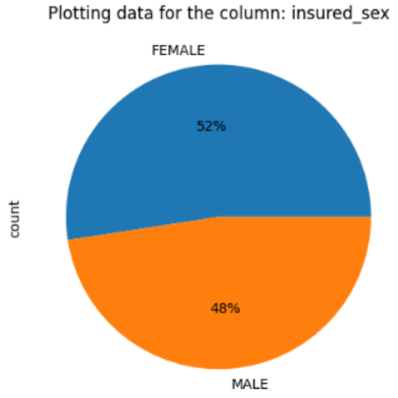


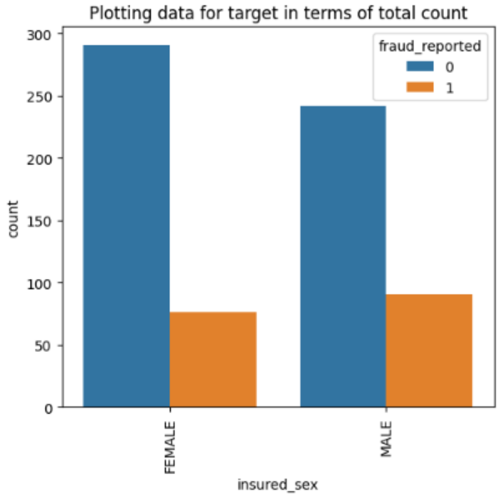


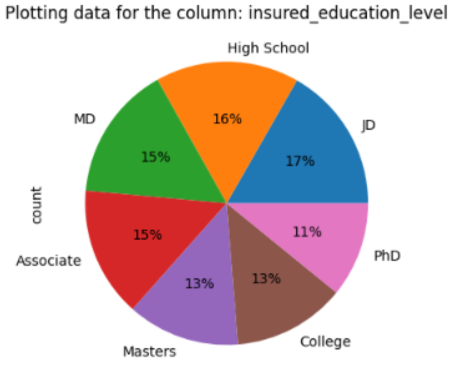


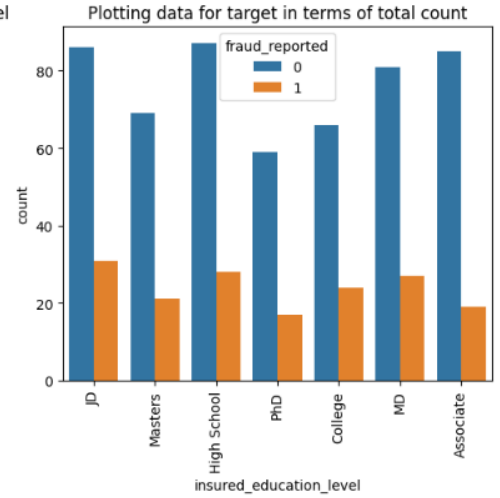


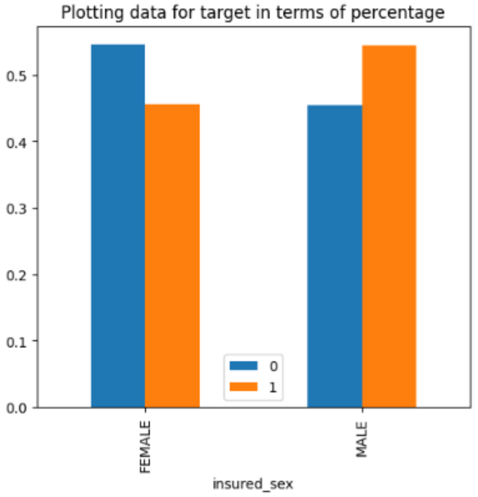
EDA on Training Data

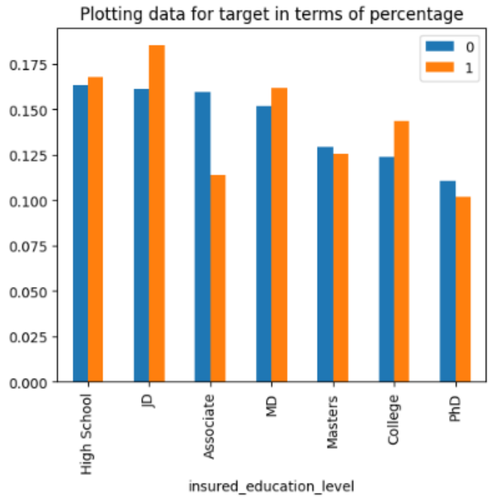




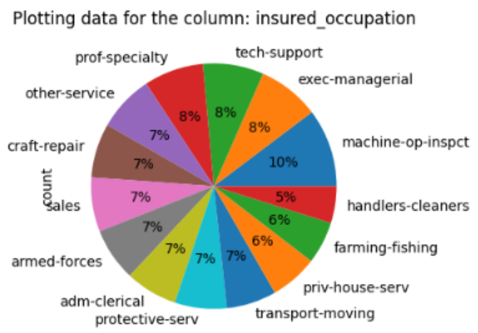


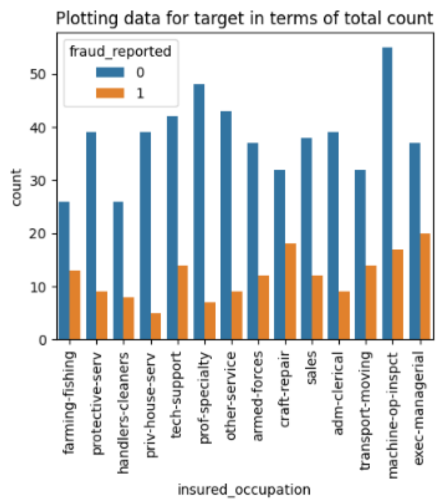


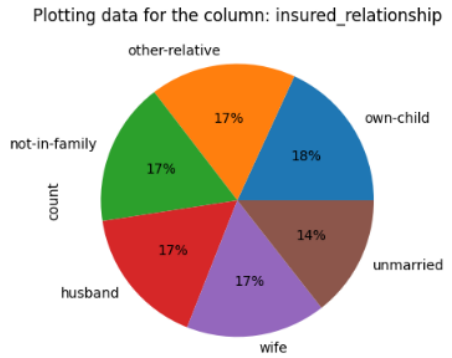


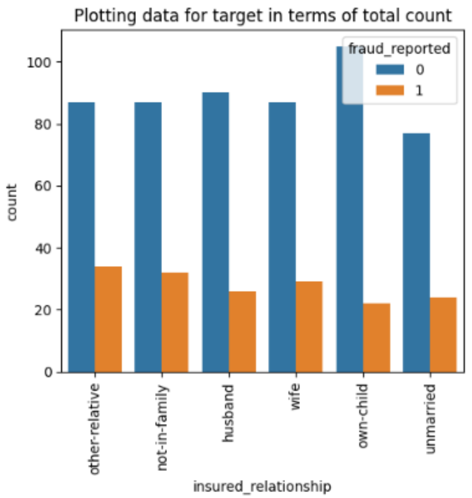


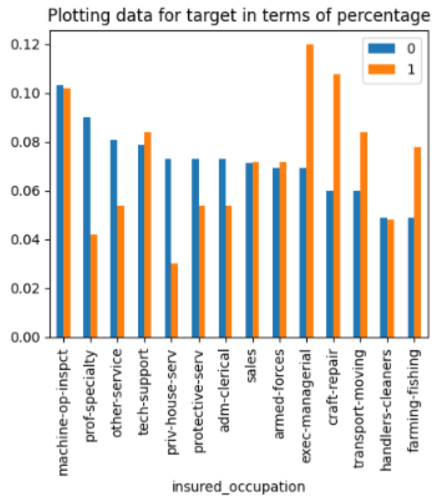
EDA on Training Data

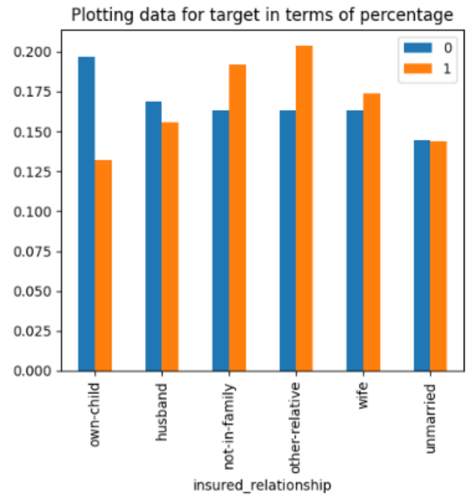




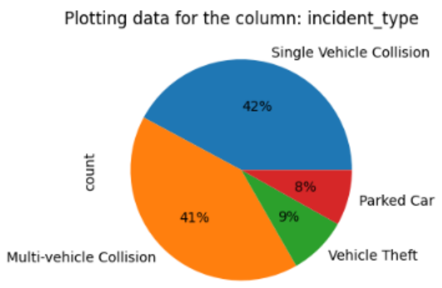


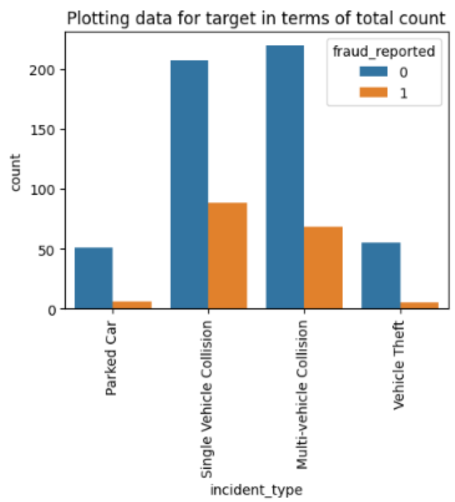


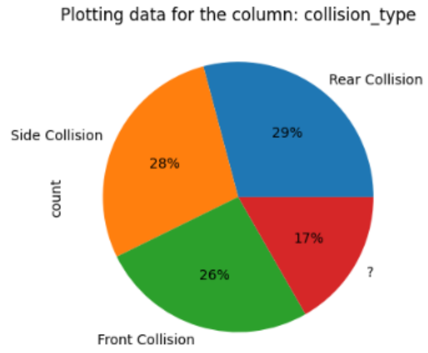


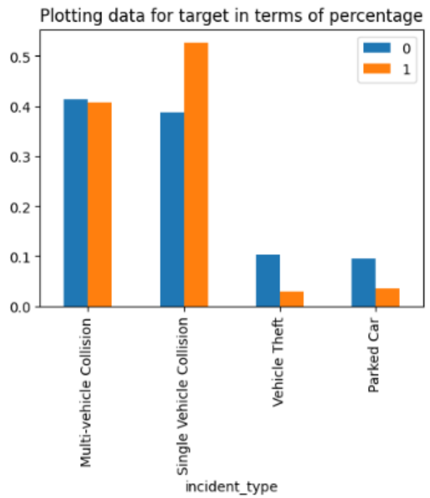


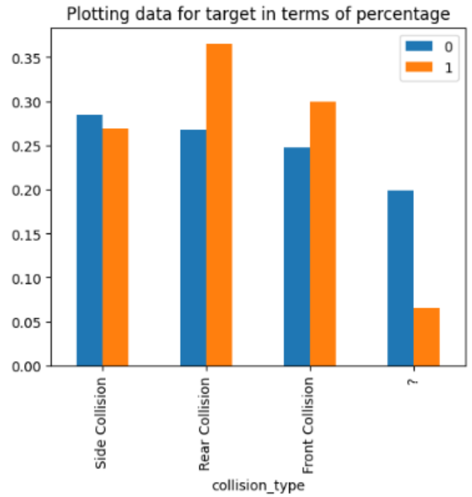
EDA on Training Data



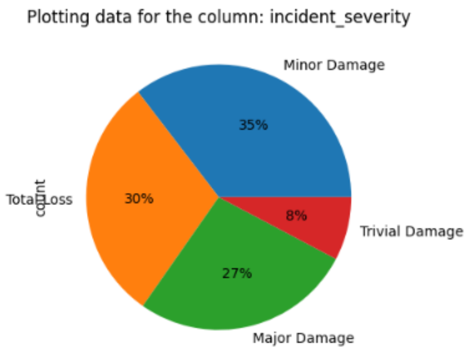


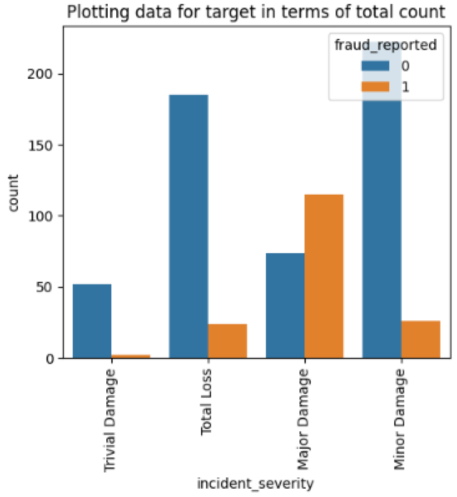


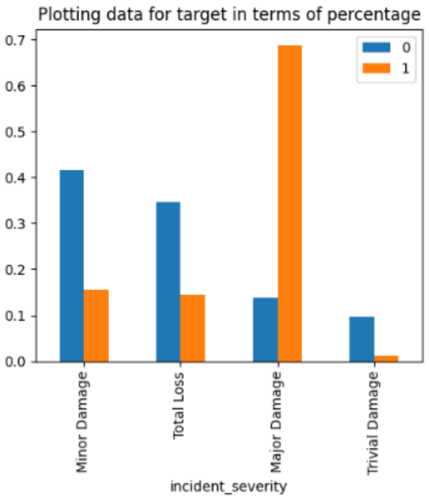


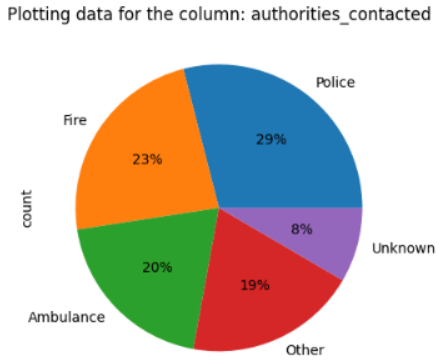


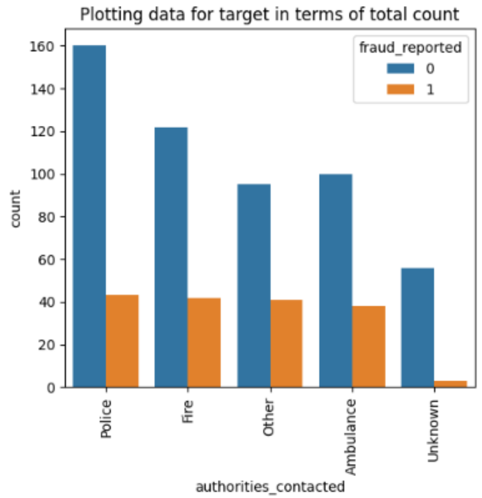
EDA on Training Data

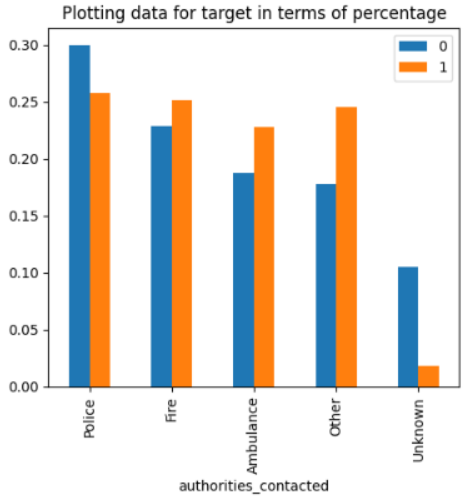




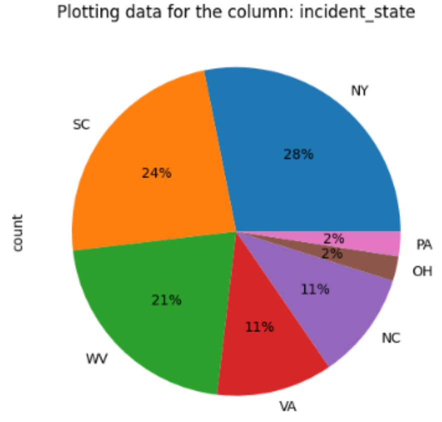


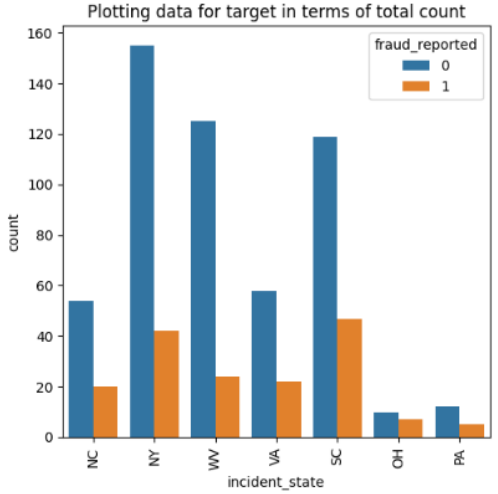


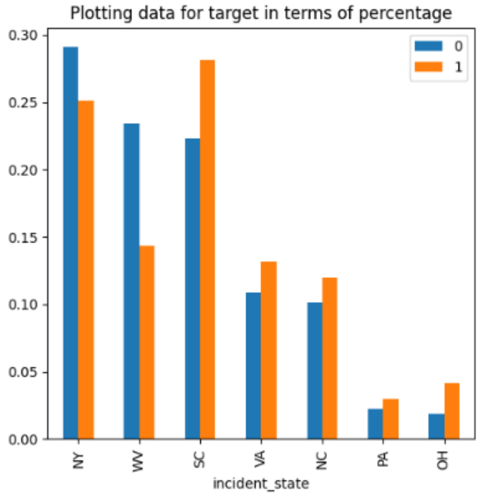


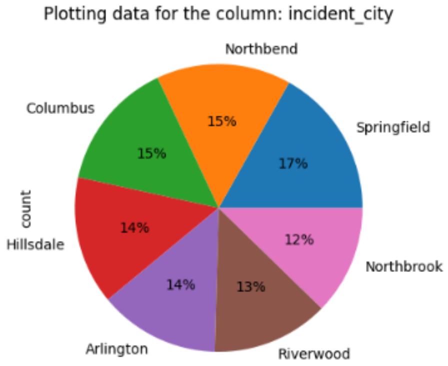


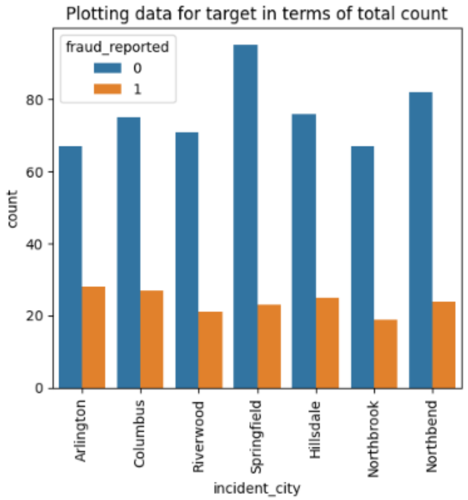
EDA on Training Data

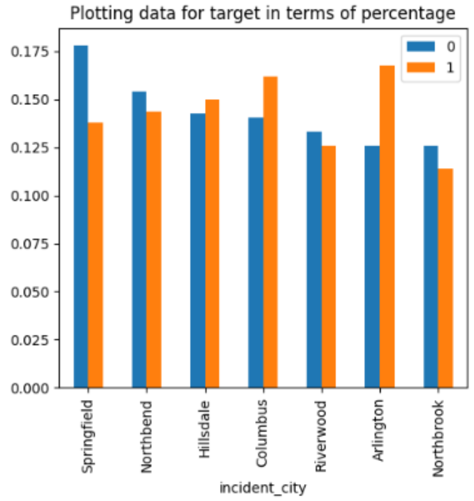




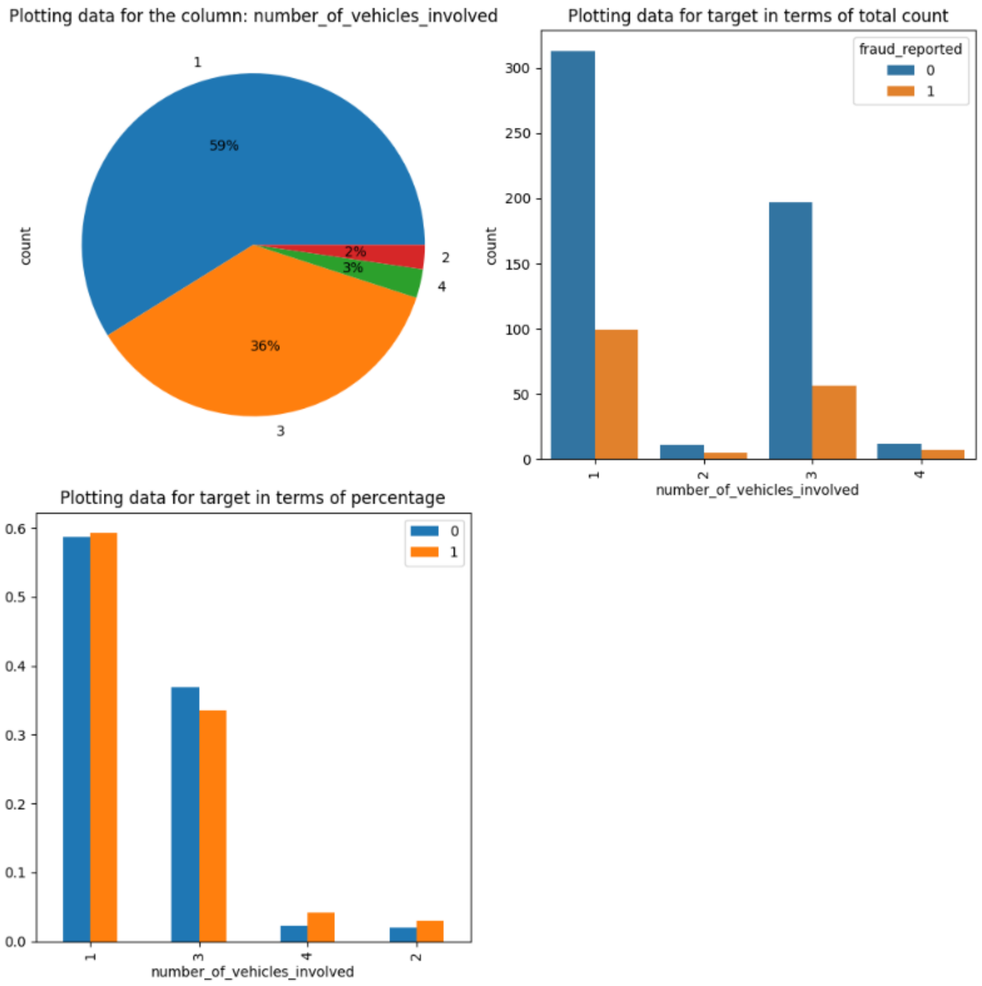


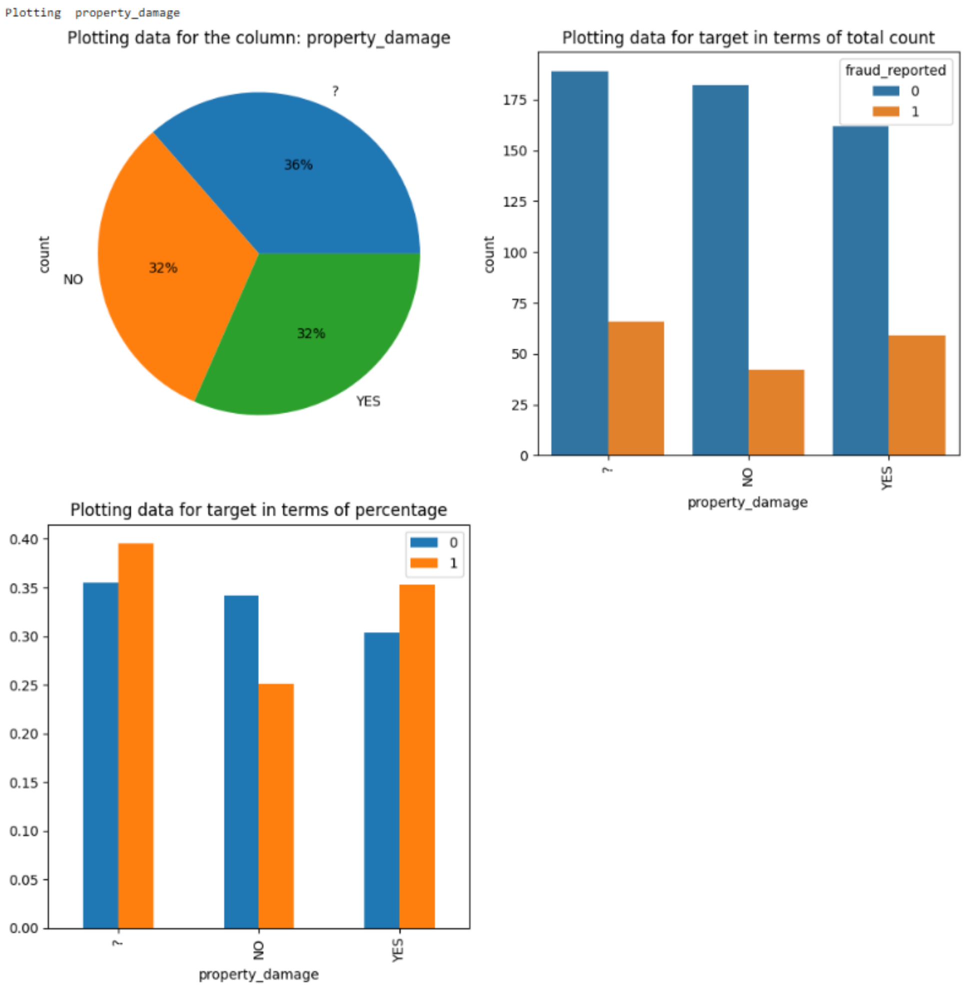




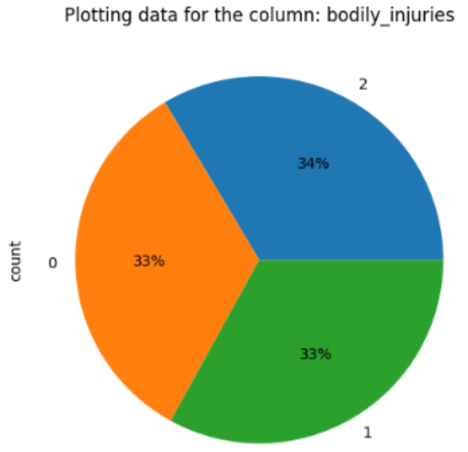


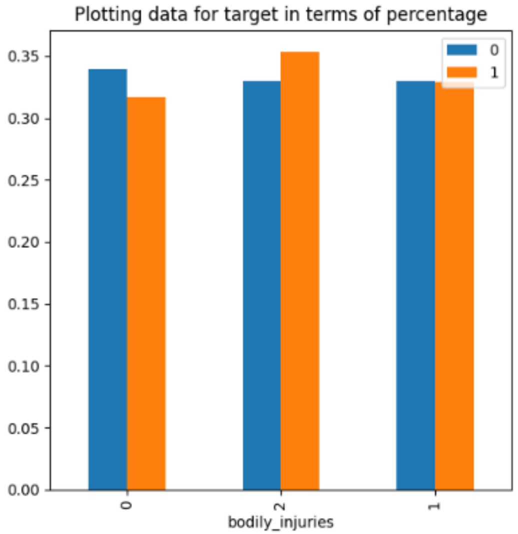
EDA on Training Data

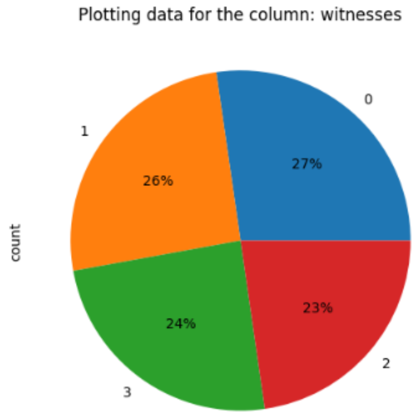


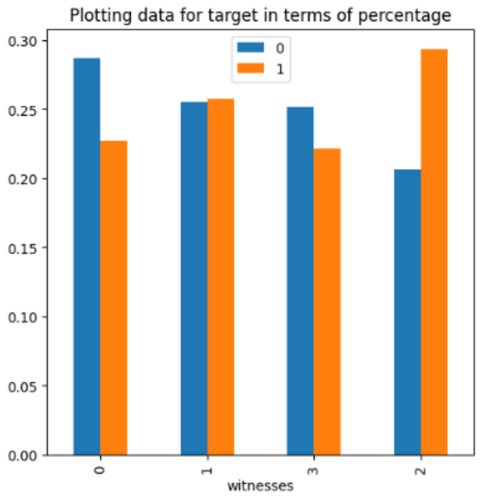


EDA on Training Data

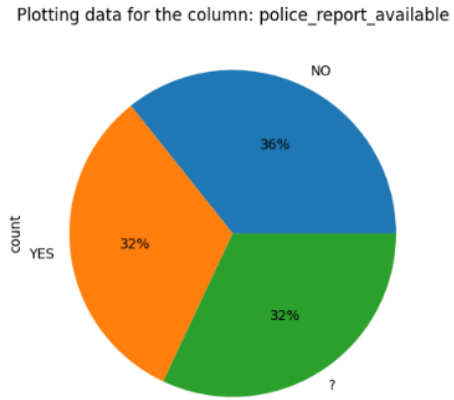


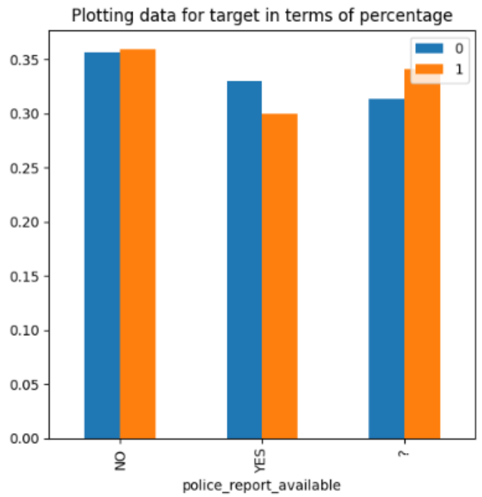


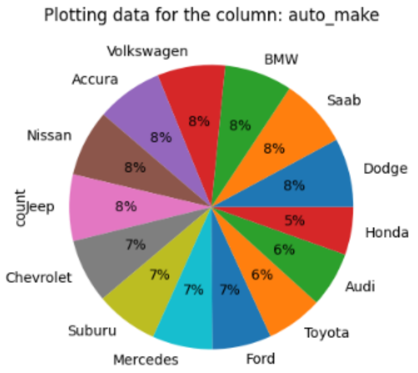


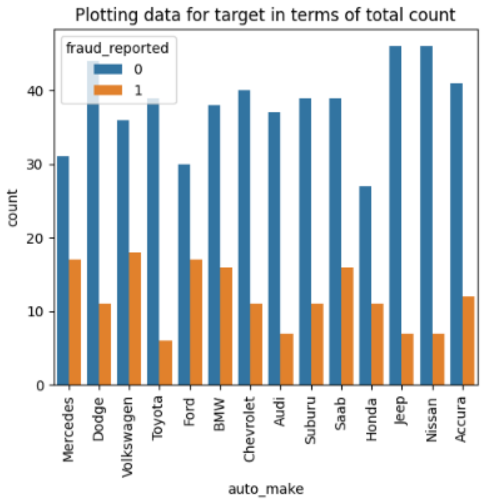


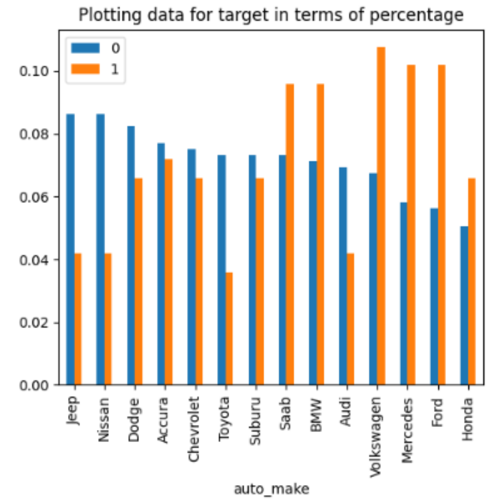
EDA on Training Data



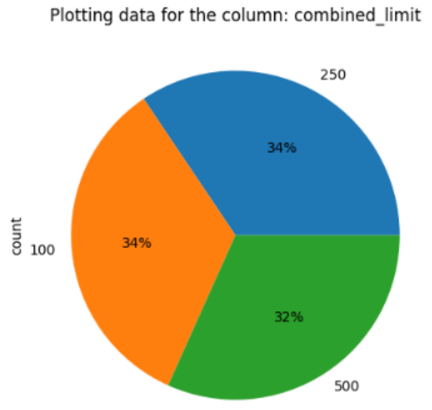


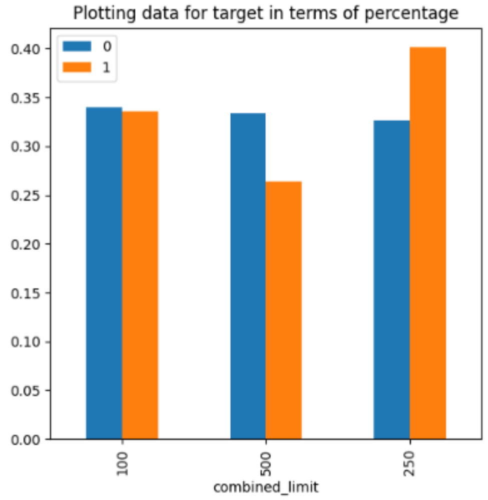


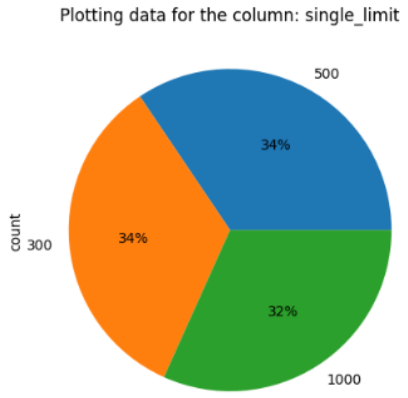


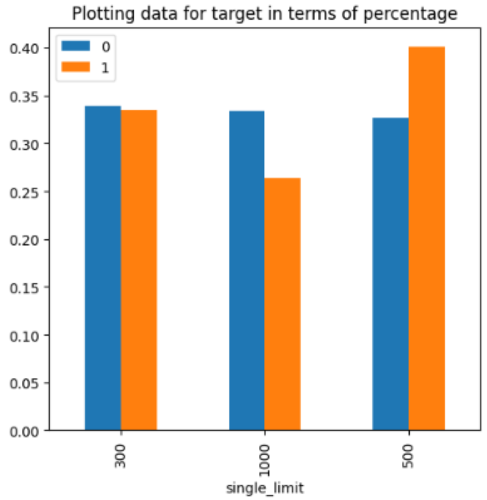


EDA on Training Data

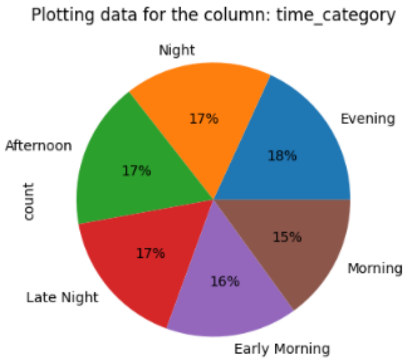


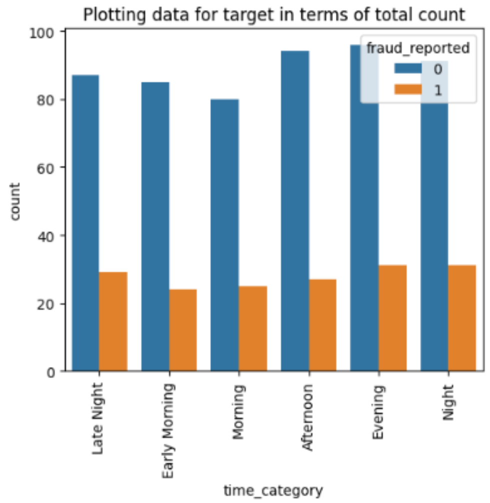


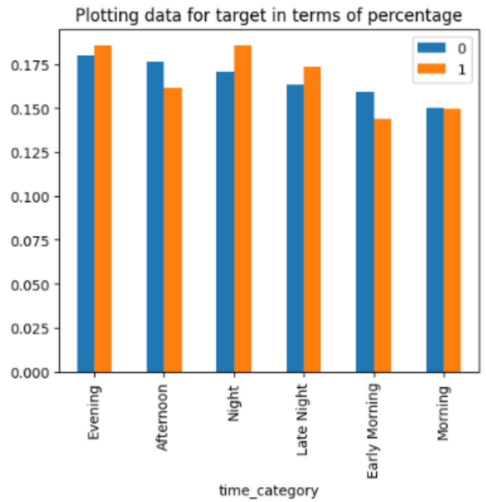


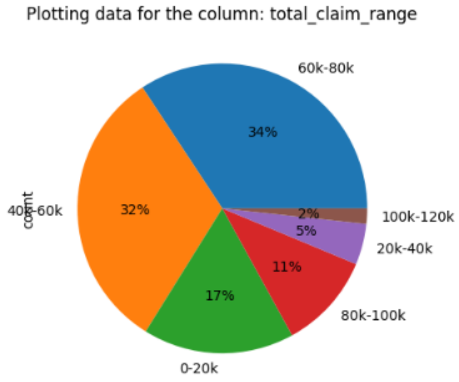


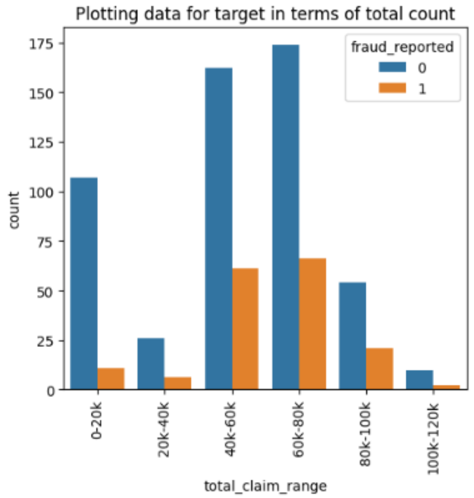
EDA on Training Data

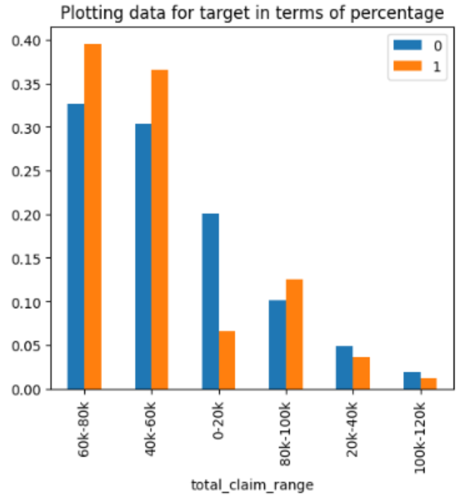




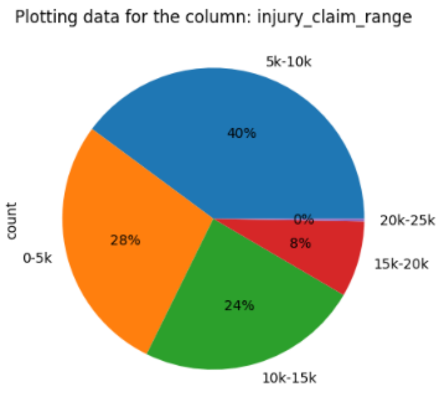


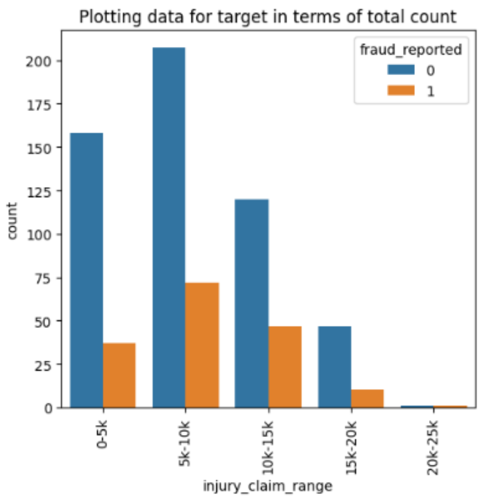


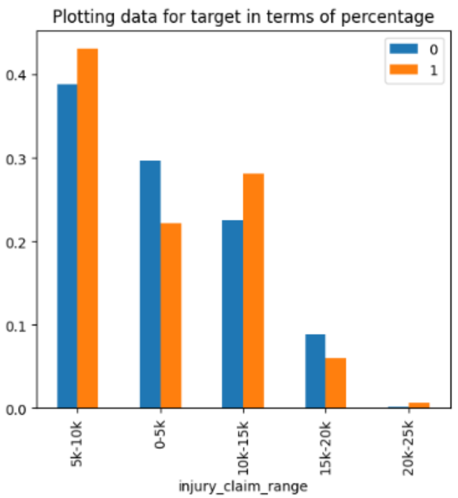


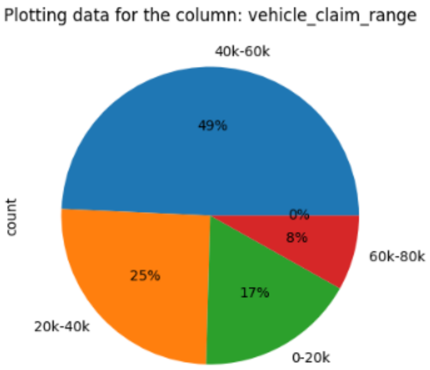


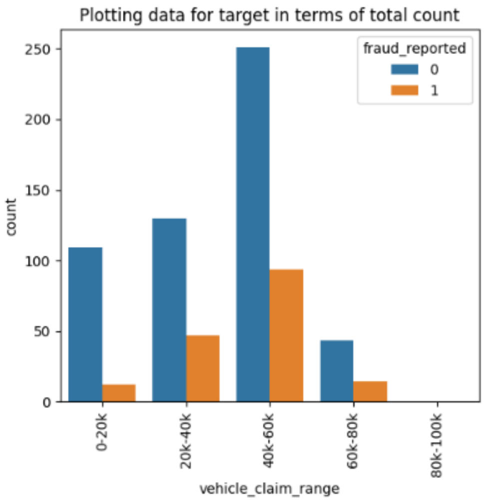
EDA on Training Data

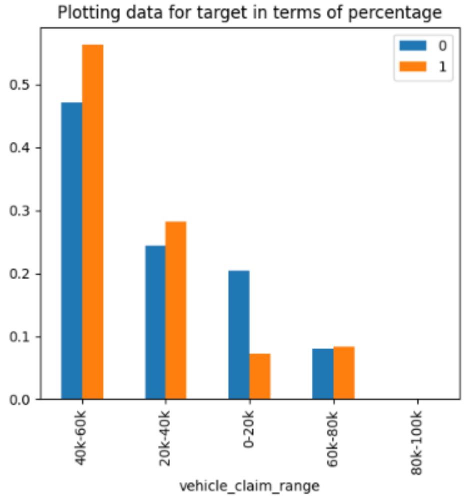




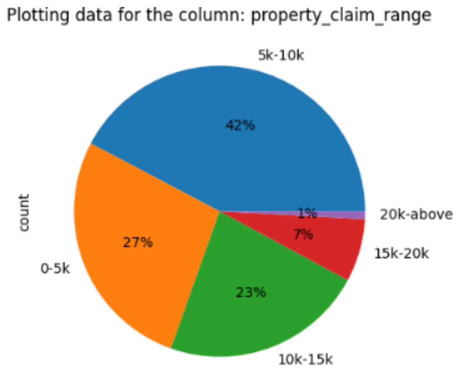


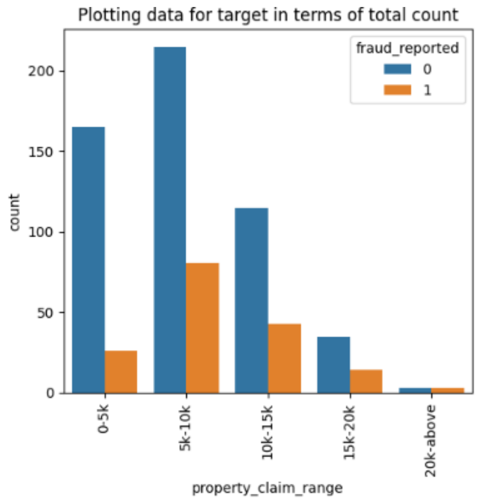


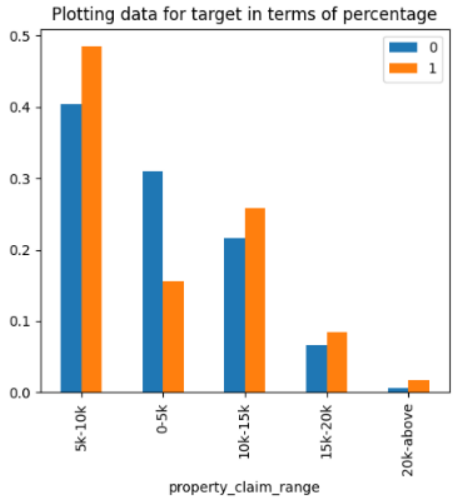


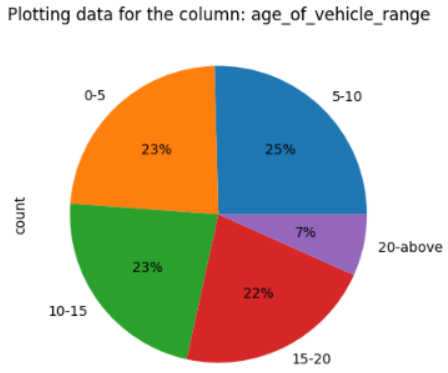


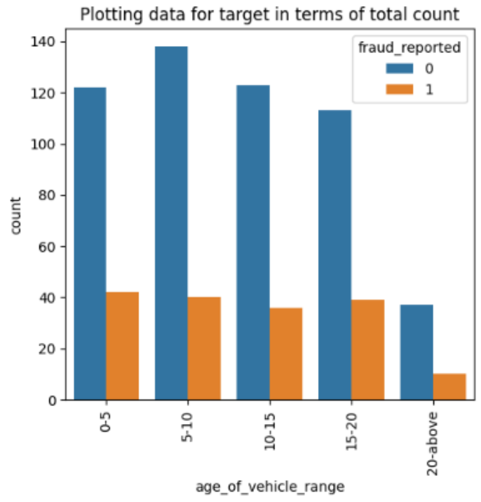
EDA on Training Data

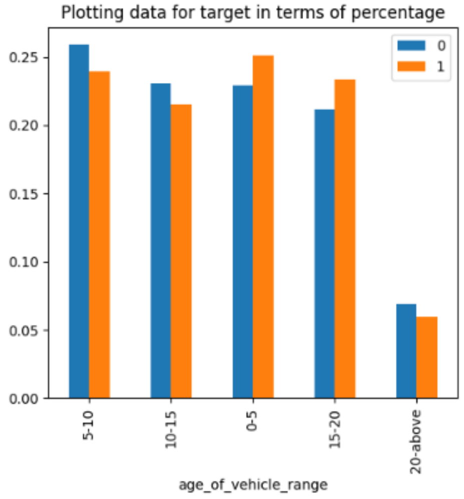




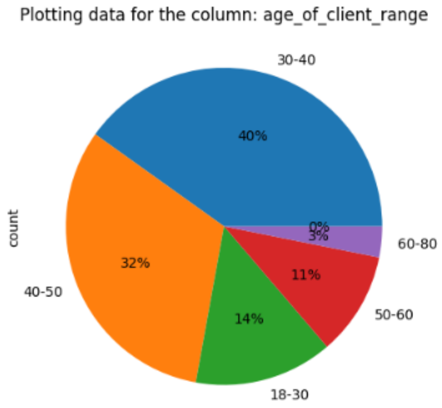


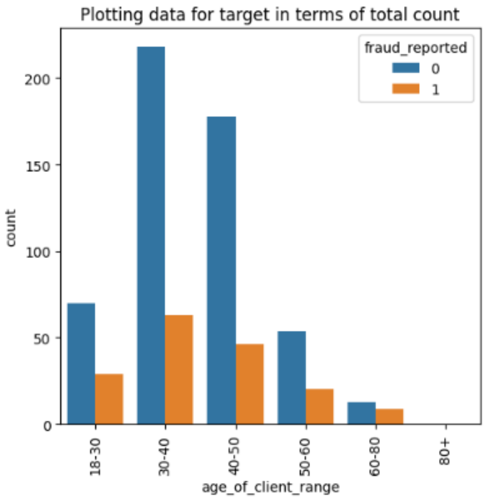


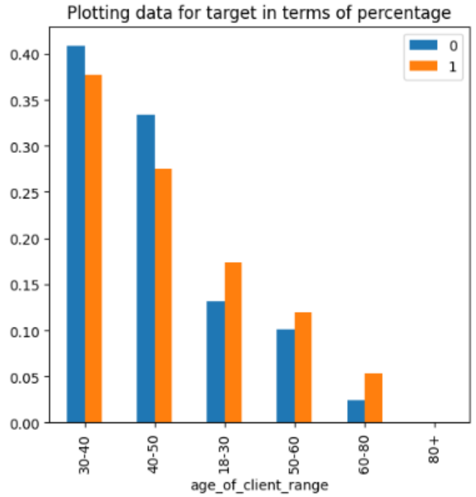


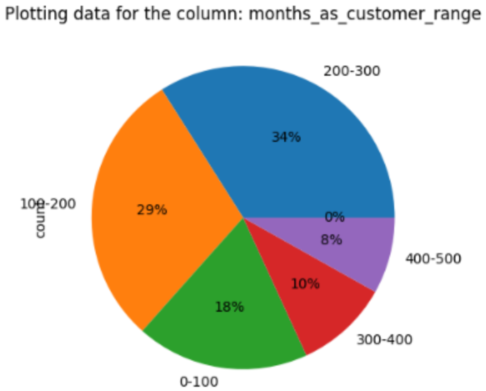


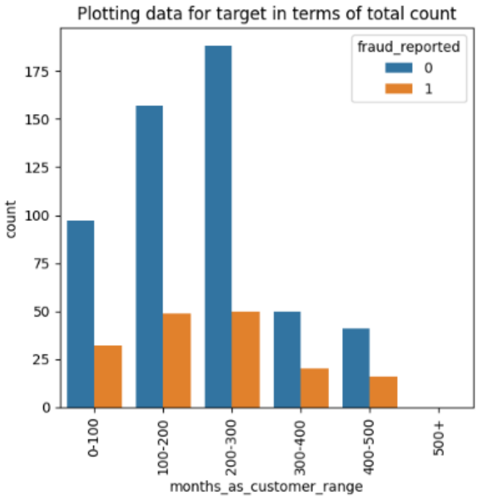
EDA on Training Data

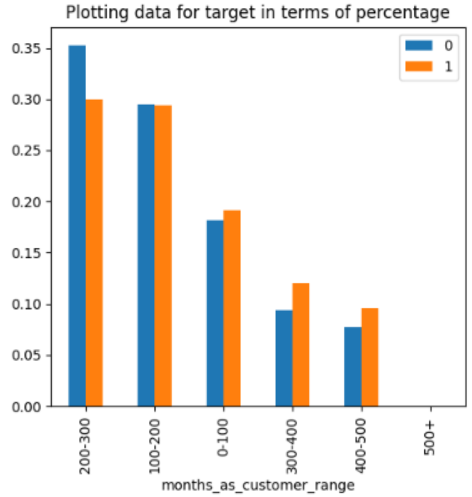












Feature Engineering

policy_csl
250/500
250/500
100/300
250/500
500/1000

single_limit
500
500
300
500
1000

combined_limit
250
250
100
250
500

policy_csl contains combined limits. To facilitate analysis, we'll split it into separate columns and drop policy_csl feature

```
df['age_of_vehicle'] = df['incident_year']-df['auto_year']
```

incident_year
2015
2015
2015
2015
2015

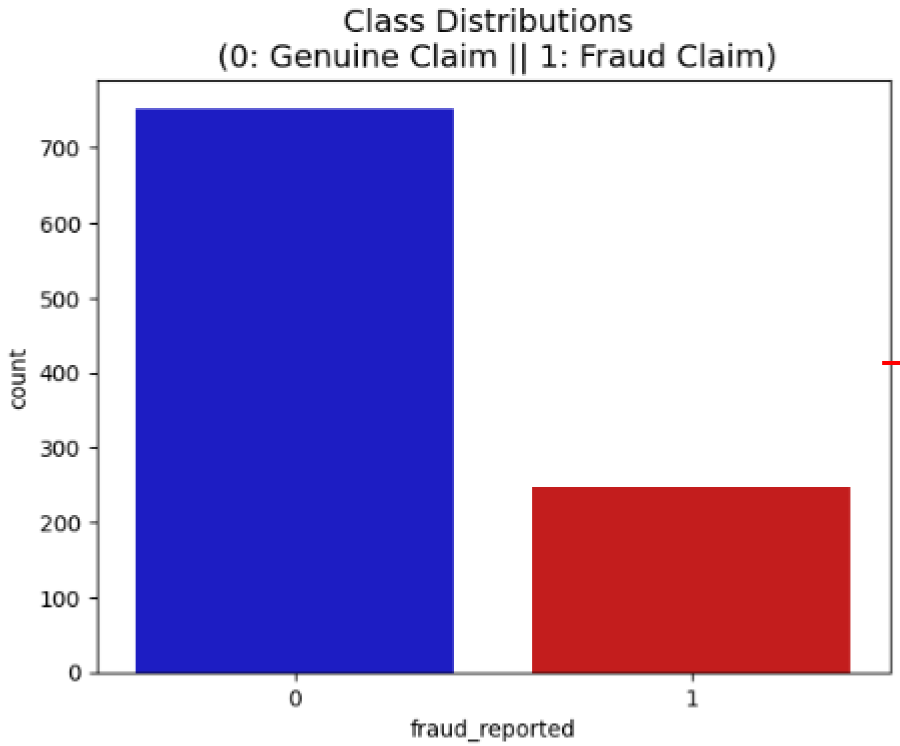
auto_year
2004
2007
2007
2014
2009

age_of_vehicle
11
8
8
1
6

Subtracting auto_year from incident_year provides the age_of_vehicle at the time of the incident

Model Building

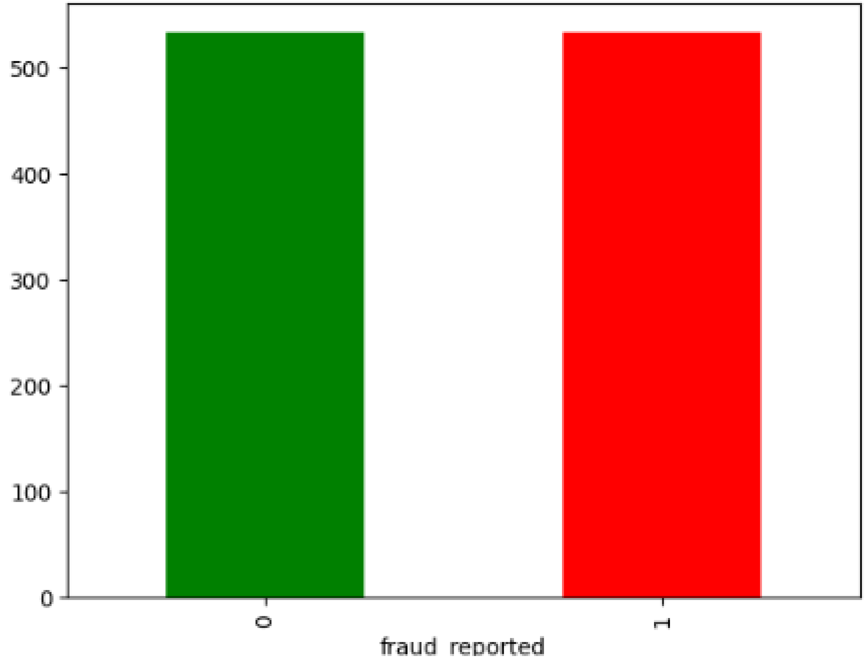
Class balance check




```python
# Import RandomOverSampler from imblearn library
from imblearn.over_sampling import RandomOverSampler

# Perform resampling on training data
Over_sample = RandomOverSampler(random_state = 5)
X_resample_os, y_resample_os = Over_sample.fit_resample(X_train, y_train)
```

Resampling



Model Building

MACHINE LEARNING

Dummy variable creation and scaling

```
# Identify the categorical columns for creating dummy variables
categorical_cols = X_resample_os.select_dtypes(include=['object','category']).columns.tolist()
print('categorical_cols', categorical_cols)
```

```
# Create dummy variables using the 'get_dummies' for categorical columns in training data
dummy = pd.get_dummies(X_resample_os[categorical_cols], columns=categorical_cols, drop_first=True).astype('int')

X_resample_os = pd.concat([X_resample_os, dummy], axis=1)

X_resample_os.drop(categorical_cols, axis=1, inplace=True)
```

```
# Import the necessary scaling tool from scikit-learn
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
# Scale the numeric features present in the training data
X_resample_os[Numerical_cols] = scaler.fit_transform(X_resample_os[Numerical_cols])
# Scale the numeric features present in the validation data
X_test[Numerical_cols] = scaler.fit_transform(X_test[Numerical_cols])
```

Model Building : Logistic regression

Feature selection

```python
# Apply RFECV to identify the most relevant features
rfecv = RFECV(estimator=LogisticRegression(solver='liblinear'), step=1, cv=StratifiedKFold(5), scoring='accuracy')
rfecv.fit(X_train, y_train)
```

```
▸ RFECV
▸ estimator: LogisticRegression
  ▸ LogisticRegression
```

	Feature	Selected	Ranking
0	policy_annual_premium	True	1
1	policy_state_IN	True	1
2	policy_state_OH	True	1
4	policy_deductable_2000	True	1
6	insured_education_level_College	True	1
5	insured_sex_MALE	True	1
7	insured_education_level_High School	True	1
8	insured_education_level_JD	True	1
14	insured_occupation_exec-managerial	True	1
9	insured_education_level_MD	True	1
10	insured_education_level_Masters	True	1
12	insured_occupation_armed-forces	True	1
13	insured_occupation_craft-repair	True	1
19	insured_occupation_priv-house-serv	True	1
17	insured_occupation_machine-op-inspct	True	1
16	insured_occupation_handlers-cleaners	True	1
30	incident_type_Parked Car	True	1
25	insured_relationship_not-in-family	True	1
26	insured_relationship_other-relative	True	1
20	insured_occupation_prof-specialty	True	1

Model Building : Logistic regression

Best Logistic Regression Model

```python
X_train_l = X_train[top_features]
X_train_l5 = sm.add_constant(X_train_l)
logm5 = sm.GLM(y_train,(sm.add_constant(X_train_l)),family = sm.families.Binomial())
logm5.fit().summary()
```

Generalized Linear Model Regression Results

Dep. Variable:	fraud_reported	**No. Observations:**	1066
Model:	GLM	**Df Residuals:**	1044
Model Family:	Binomial	**Df Model:**	21
Link Function:	Logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-688.48
Date:	Sat, 14 Jun 2025	**Deviance:**	1377.0
Time:	18:18:41	**Pearson chi2:**	1.08e+03
No. Iterations:	4	**Pseudo R-squ. (CS):**	0.09026
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
const	-0.0093	0.064	-0.145	0.885	-0.136	0.117
policy_annual_premium	-0.1441	0.066	-2.190	0.029	-0.273	-0.015
policy_state_IN	-0.0695	0.074	-0.939	0.348	-0.215	0.076
policy_state_OH	0.0622	0.075	0.830	0.407	-0.085	0.209
policy_deductable_2000	0.0609	0.066	0.924	0.355	-0.068	0.190
insured_education_level_College	0.1097	0.071	1.544	0.123	-0.030	0.249
insured_sex_MALE	0.2484	0.066	3.788	0.000	0.120	0.377
insured_education_level_High School	0.0665	0.073	0.913	0.361	-0.076	0.209
insured_education_level_JD	0.0833	0.073	1.144	0.253	-0.059	0.226
insured_occupation_exec-managerial	0.1091	0.069	1.573	0.116	-0.027	0.245
insured_education_level_MD	0.0501	0.072	0.693	0.488	-0.092	0.192
insured_occupation_armed-forces	0.0490	0.066	0.741	0.459	-0.081	0.179
insured_occupation_craft-repair	0.1532	0.069	2.227	0.026	0.018	0.288
insured_occupation_priv-house-serv	-0.2364	0.076	-3.129	0.002	-0.384	-0.088
incident_type_Parked Car	-0.1985	0.074	-2.686	0.007	-0.343	-0.054
insured_relationship_not-in-family	0.1349	0.068	1.988	0.047	0.002	0.268
insured_relationship_other-relative	0.2079	0.068	3.057	0.002	0.075	0.341
insured_occupation_prof-specialty	-0.1385	0.069	-2.005	0.045	-0.274	-0.003
insured_occupation_sales	0.0473	0.067	0.702	0.482	-0.085	0.179
insured_occupation_protective-serv	-0.0560	0.067	-0.837	0.402	-0.187	0.075
insured_occupation_transport-moving	0.1062	0.067	1.580	0.114	-0.026	0.238
incident_type_Single Vehicle Collision	0.2418	0.067	3.583	0.000	0.110	0.374

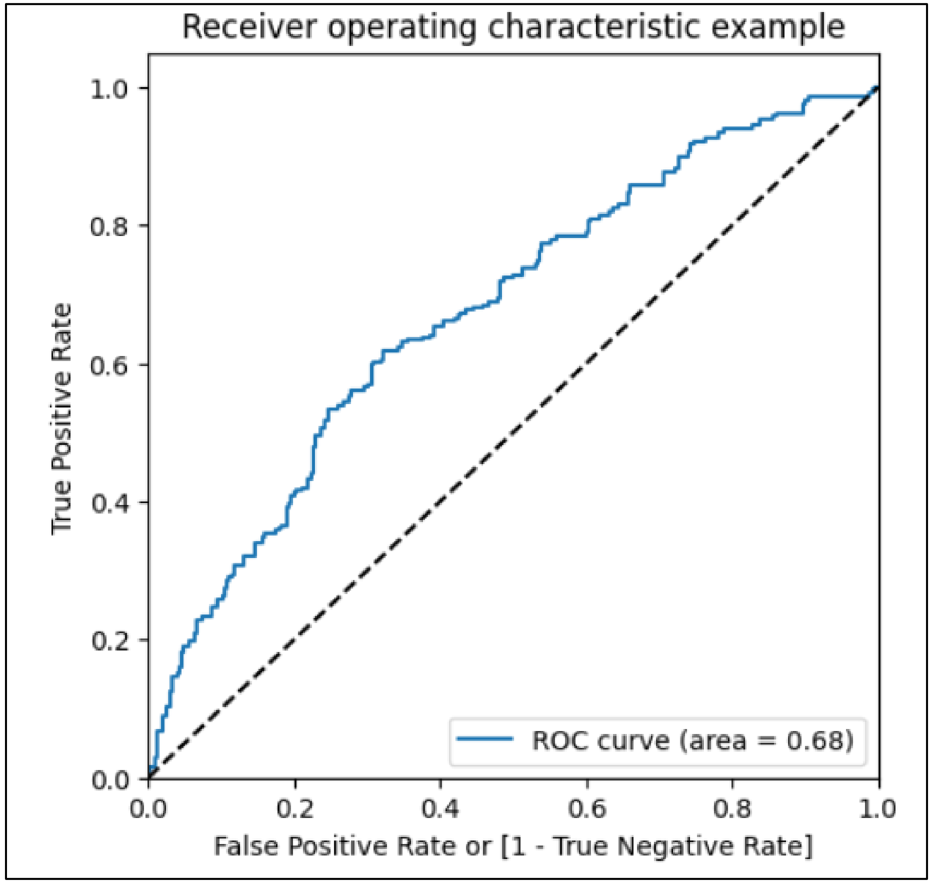
	Feature	VIF
2	policy_state_OH	1.36
1	policy_state_IN	1.33
6	insured_education_level_High School	1.29
9	insured_education_level_MD	1.27
7	insured_education_level_JD	1.26
4	insured_education_level_College	1.23
8	insured_occupation_exec-managerial	1.16
19	insured_occupation_transport-moving	1.12
17	insured_occupation_sales	1.12
11	insured_occupation_craft-repair	1.12
16	insured_occupation_prof-specialty	1.12
15	insured_relationship_other-relative	1.12
14	insured_relationship_not-in-family	1.12
20	incident_type_Single Vehicle Collision	1.11
10	insured_occupation_armed-forces	1.10
18	insured_occupation_protective-serv	1.10
13	incident_type_Parked Car	1.09
12	insured_occupation_priv-house-serv	1.07
0	policy_annual_premium	1.05
3	policy_deductable_2000	1.05
5	insured_sex_MALE	1.04

Model Building : Logistic regression

```
y_train_pred_final['Predicted'] = y_train_pred_final.Prdicted_Prob.map(lambda x: 1 if x > 0.5 else 0)
```

```
[322, 211]
[185, 348]
```

Metrics	value	Remarks
accuracy	0.628517824	**62.85%**
sensitivity/Recall	0.652908068	**65.3%** of actual positives were correctly predicted
specificity	0.60412758	**60.4%** of actual negatives were correctly predicted
Precision	0.62254025	**62.2%** of predicted positives were correct
f1_score	0.637362637	Balance between precision and recall

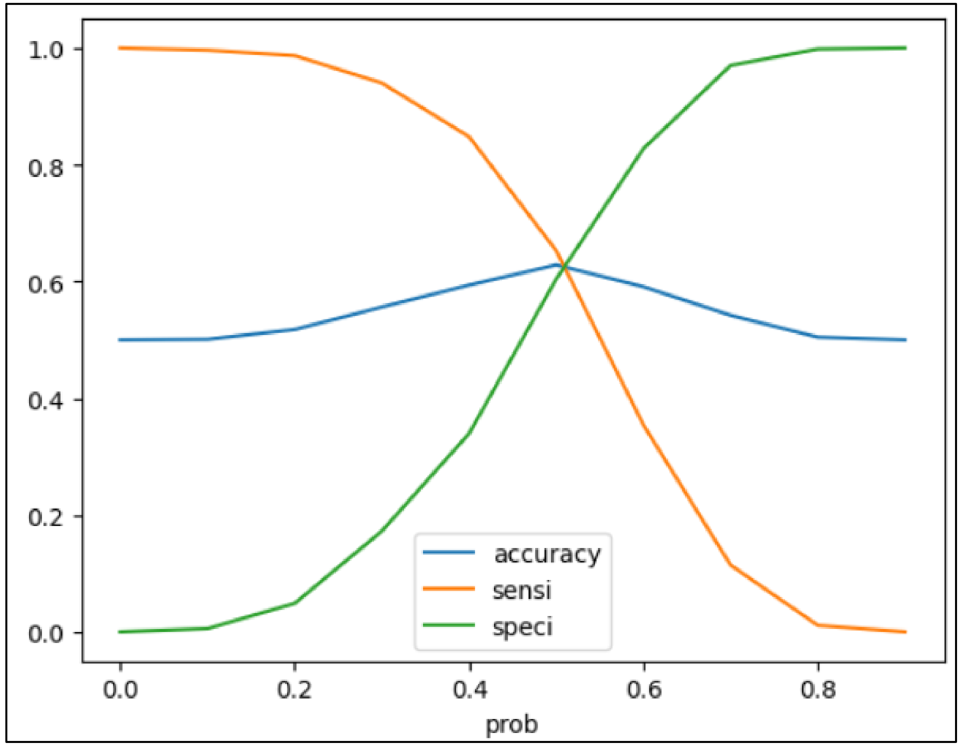


Model Building : Logistic regression

```
numbers = [float(x)/10 for x in range(10)]
for i in numbers:
    y_train_pred_final[i]= y_train_pred_final.Prdicted_Prob.map(lambda x: 1 if x > i else 0)
```

	Actual	Prdicted_Prob	Predicted	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0	0	0.313163	0	1	1	1	1	0	0	0	0	0	0
1	0	0.578618	1	1	1	1	1	1	1	0	0	0	0
2	1	0.569165	1	1	1	1	1	1	1	0	0	0	0
3	0	0.582130	1	1	1	1	1	1	1	0	0	0	0
4	0	0.294940	0	1	1	1	0	0	0	0	0	0	0

	prob	accuracy	sensi	speci
0.0	0.0	0.500000	1.000000	0.000000
0.1	0.1	0.500938	0.996248	0.005629
0.2	0.2	0.517824	0.986867	0.048780
0.3	0.3	0.556285	0.939962	0.172608
0.4	0.4	0.593809	0.848030	0.339587
0.5	0.5	0.628518	0.652908	0.604128
0.6	0.6	0.590994	0.354597	0.827392
0.7	0.7	0.542214	0.114447	0.969981
0.8	0.8	0.504690	0.011257	0.998124
0.9	0.9	0.500000	0.000000	1.000000

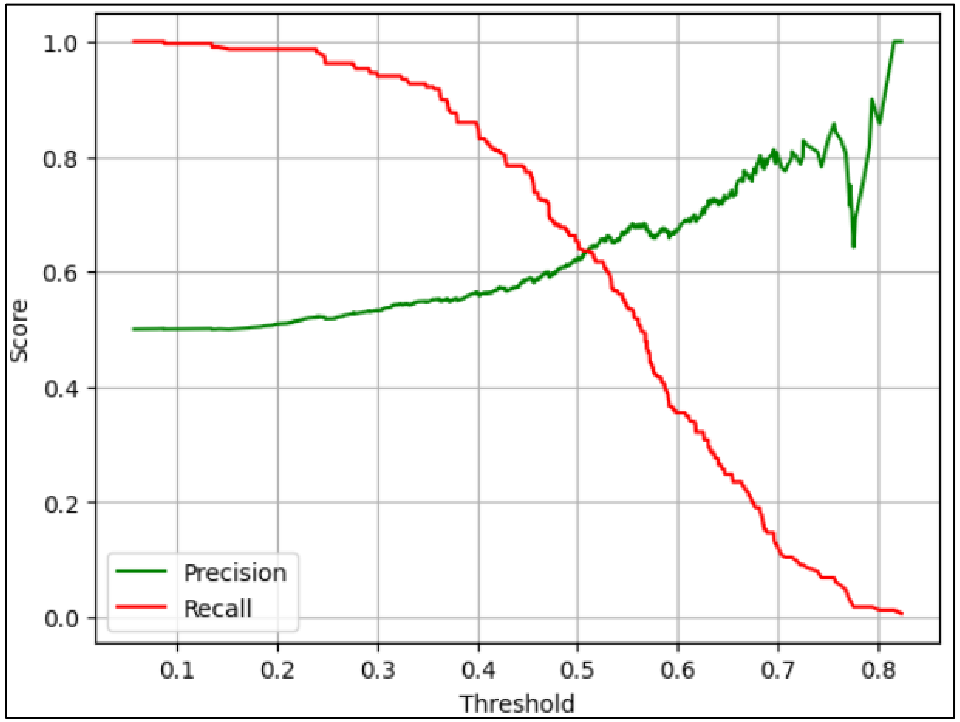


Model Building : Logistic regression

```python
# Create a column for final prediction based on the optimal cutoff
y_train_pred_final['Optimal_Predicted'] = y_train_pred_final.Prdicted_Prob.map(lambda x: 1 if x > 0.45 else 0)
```

Metrics	value	Remarks
accuracy	0.614446529	**61.44%**
sensitivity/Recall	0.772983114	**77.3%** of actual positives were correctly predicted
specificity	0.455909944	**45.6%** of actual negatives were correctly predicted
Precision	0.586894587	**58.7%** of predicted positives were correct
f1_score	0.667206478	Balance between precision and recall

precision-recall curve



Model Building : Random forest

```python
# Build a base random forest model
rf = RandomForestClassifier(n_estimators=100, max_depth=4, max_features=5,random_state=100,oob_score=True)
rf.fit(X_train,y_train)
```

```
RandomForestClassifier
RandomForestClassifier(max_depth=4, max_features=5, oob_score=True,
                       random_state=100)
```

	features	Imp
36	incident_severity_Minor Damage	0.088142
37	incident_severity_Total Loss	0.057772
42	authorities_contacted_Unknown	0.036399
38	incident_severity_Trivial Damage	0.035396
0	policy_annual_premium	0.033319
35	collision_type_Unknown	0.024426
97	vehicle_claim_range_40k-60k	0.023208
27	insured_relationship_own-child	0.021713
26	insured_relationship_other-relative	0.020784
79	combined_limit_500	0.019508
5	insured_sex_MALE	0.019384
30	incident_type_Parked Car	0.019108
32	incident_type_Vehicle Theft	0.017087
19	insured_occupation_priv-house-serv	0.015043
100	property_claim_range_5k-10k	0.014860

Metrics	value	Remarks
accuracy	0.815196998	**81.51%**
sensitivity/Recall	0.791744841	**79.17%** of actual positives were correctly predicted
specificity	0.838649156	**83.9%** of actual negatives were correctly predicted
Precision	0.830708661	**83.1%** of predicted positives were correct
f1_score	0.810758886	Balance between precision and recall

```python
# Use cross validation to check if the model is overfitting
cv_scores = cross_val_score(rf, X_train_rf, y_train, cv=5, scoring='accuracy')

print("Cross-Validation Accuracy Scores:", cv_scores)
print("Mean CV Accuracy:", cv_scores.mean())
```

```
Cross-Validation Accuracy Scores: [0.72897196 0.7370892  0.79342723 0.82629108 0.84507042]
Mean CV Accuracy: 0.7861699793778246
```

Training Accuracy	81.51%
Mean Cross-Validation Accuracy	78.61%

The accuracy gap is just 2.9% — which is small and acceptable and model is NOT significantly overfitting

Model Building : Random forest

Hyperparameter Tuning

```
# Use grid search to find the best hyperparamter values
Classifier_rf = RandomForestClassifier(random_state=42, n_jobs=-1)
# Best Hyperparameters
params = {
    'max_depth': [1, 2, 5, 10, 20],
    'min_samples_leaf': [5, 10, 20, 50, 100],
    'max_features': [2,3,4],
    'n_estimators': [10, 30, 50, 100, 200]
}
```

```
# Building random forest model based on results of hyperparameter tuning
grid_search = GridSearchCV(estimator=Classifier_rf, param_grid= params,cv=4, n_jobs=-1, verbose=1, scoring='accuracy')
```

```
# Make predictions on training data
grid_search.fit(X_train_rf,y_train)
```

```
Fitting 4 folds for each of 375 candidates, totalling 1500 fits
GridSearchCV
best_estimator_: RandomForestClassifier
RandomForestClassifier
RandomForestClassifier(max_depth=20, max_features=4, min_samples_leaf=5,
                       n_estimators=30, n_jobs=-1, random_state=42)
```

Metrics	value	Remarks
accuracy	0.873358349	**87.30%**
sensitivity/Recall	0.853658537	**85.36%** of actual positives were correctly predicted
specificity	0.889305816	**88.9%** of actual negatives were correctly predicted
Precision	0.885214008	**88.5%** of predicted positives were correct
f1_score	0.869149952	Balance between precision and recall

Model Evaluation

Logistic regression

Metrics	value	Remarks
accuracy	0.403333333	**40.30%**
sensitivity/Recall	0.75	**75 %** of actual positives were correctly predicted
specificity	0.277272727	**27.7%** of actual negatives were correctly predicted
Precision	0.273972603	**27.4%** of predicted positives were correct
f1_score	0.401337793	Balance between precision and recall

Random forest

Metrics	value	Remarks
accuracy	0.773333333	**77.30%**
sensitivity/Recall	0.75	**75%** of actual positives were correctly predicted
specificity	0.8227	**82.3%** of actual negatives were correctly predicted
Precision	0.5666	**56.6%** of predicted positives were correct
f1_score	0.6	Balance between precision and recall

Conclusion

Logistic regression

Metrics	Train Data	Test Data
Accuracy	0.6144	0.4033
Sensitivity/Recall	0.7730	0.7500
Specificity	0.4559	0.2773
Precision	0.5869	0.2740
f1_score	0.6672	0.4013

- The logistic regression model is overfitting — performing reasonably well on the training data but very poorly on the test set.
- The recall remains high, which means the model still finds most positive cases, but it sacrifices precision and specificity, resulting in many false positives.
- Model is unreliable for deployment in its current form

Random forest:

Metrics	Train Data	Test Data
Accuracy	0.8734	0.7733
Sensitivity/Recall	0.8537	0.7500
Specificity	0.8893	0.8227
Precision	0.8852	0.5666
f1_score	0.8691	0.6000

- The Random Forest model demonstrates strong and well-rounded performance on both training and test datasets.
- It generalizes fairly well with a good balance of sensitivity (recall) and specificity, making it reliable for binary classification tasks.
- While test precision is lower, it's still usable — but we may need to adjust thresholds depending on business needs (e.g., favoring precision over recall or vice versa).
- Overall, this model is a good candidate for deployment or further tuning.