Fraudulent Claim Detection Report

Objective:

- Global Insure aims to enhance its ability to detect fraudulent insurance claims by leveraging historical claim data.
- The company seeks to identify patterns and key indicators that differentiate fraudulent claims from genuine ones.
- By developing a predictive model, it intends to assess the likelihood of fraud in incoming claims, enabling proactive fraud detection and reducing financial losses.

Submitted by:

- Balamurugan Chandran
- Deven Karla

1. Data Preparation

1.0 – Import libraries

1.1 Loaded the excel input data

- Using df.load(), identified authorities_contacted attribute has 91 empty values.
- `_c39` attribute `is empty column`
- All other attributes are not null
- `fraud_reported – is the predictive output variable.`

```
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
```

2. Data Cleaning

2.1 Handle null values

2.1.1 *Examine the columns to determine if any value or column needs to be treated*

- Checked number of missing values in each column using 100 * df.isna().mean()

2.1.2 Handle rows containing null values

Replaced null value with “No One” [Note: None is being treated as na]

df["authorities_contacted"].fillna("No One", inplace=True)

Check the value count to confirm

```
df["authorities_contacted"].value_counts()
```

```
authorities_contacted
Police       292
Fire         223
Other        198
Ambulance    196
No One        91
Name: count, dtype: int64
```

2.2 Identify and handle redundant values and columns

2.2.1 *Examine the columns to determine if any value or column needs to be treated*

Identified the unique values for each attributes, that won't contribute much for predictable variable

```
# Write code to display all the columns with their unique values and counts and check for redundant values
def unique_values_percentage_summary(df, columns):
    """
    A function to calculate the summary of categorical columns of a dataframe
        @param: df -> DataFrame
        @param: columns: list of categorical columns in the dataframe
        @returns: dictionary of summary of unique value counts and their percentages
    """
    results = {}
    for column in columns:
        # Count unique values and calculate percentages
        value_counts = df[column].value_counts()
        percentages = df[column].value_counts(normalize=True) * 100

        # Combine counts and percentages into a DataFrame
        unique_summary = pd.DataFrame(
            {"Count": value_counts, "Percentage (%)": percentages}
        )

        # Add the result to the dictionary
        results[column] = unique_summary

    return results
```

2.2.2 Identify and drop any columns that are completely empty

- Dropped _c39 empty column
- Based on data types, categorical_columns and numerical_columns are identified

2.2.3 Identify and drop rows where features have illogical or invalid values, such as negative values for features that should only have positive values

- Added a function to check if it has both positive and negative values in any single column
- Found **umbrella_limit** attribute has a single negative value
- Removing this illogical row.
- **collision_type** attribute's value ? is replaced with "No Collision" value

2.2.4 Identify and remove columns where a large proportion of the values are unique or near-unique, as these columns are likely to be identifiers or have very limited predictive power

- Identified attributes with 90% unique values are more than 90% of the rows present - so considered as limited predictive power to these columns
- limited_predictive_power = ['policy_number', 'policy_bind_date', 'policy_annual_premium', 'insured_zip', 'incident_location'] resulted attributes are returned
- Dropped these attributes as not making significance to predictable variable

2.3 Fix Data Types

- **incident_date** attribute datatype is changed to datetime
- Confirmed the change using dfAnalysis.info()

3.Train-Validation Split

- **fraud_reported** is the target predictive variable -> y
- All other dependent variables are stored as X
- Split the data using train_test_split with stratify =y and random_state=42

3.3 Split the data [3 Marks]

```
# 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, test_size=0.3, stratify=y, random_state=42
)
# Reset index for all train and test sets
X_train.reset_index(inplace=True, drop=True)
X_test.reset_index(inplace=True, drop=True)
y_train.reset_index(inplace=True, drop=True)
y_test.reset_index(inplace=True, drop=True)
```

4. EDA on training data

4.1 Perform univariate analysis

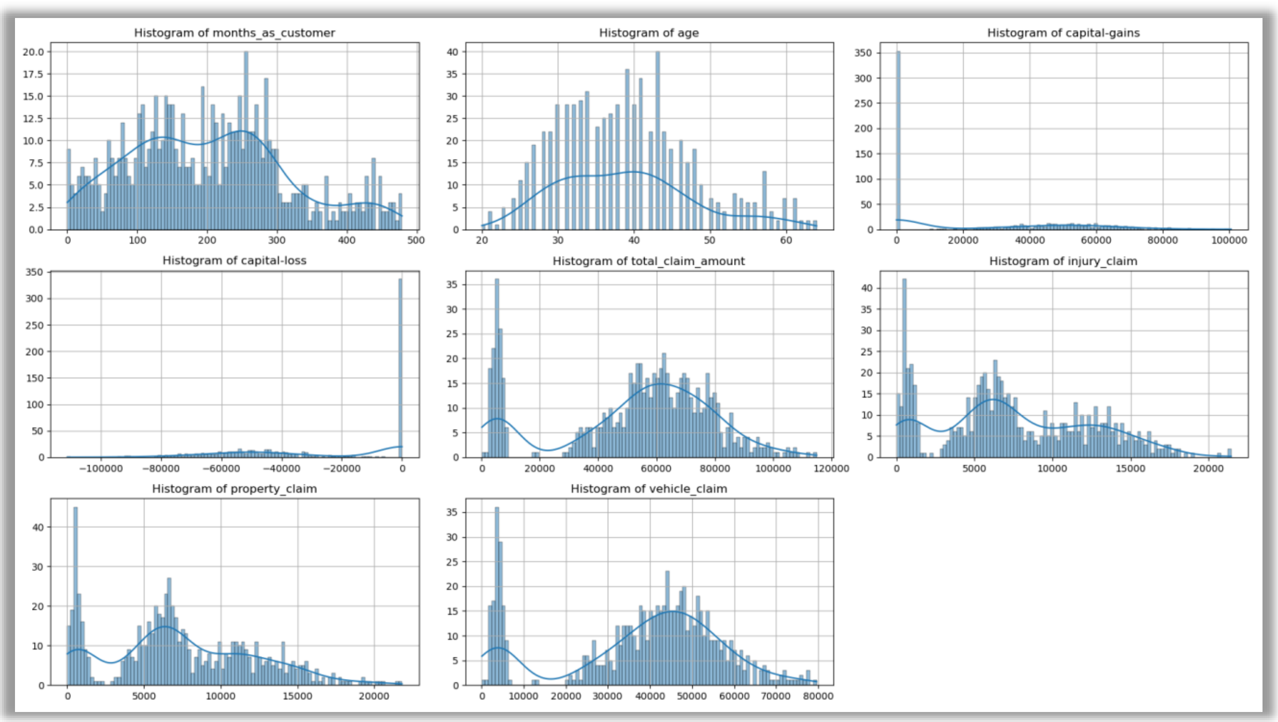
- Added a categorize(df,columns) function to take input data frame and select numerical and categorical columns based in data types
- Int64 and float64 are treated as numerical; object, category are treated as categorical
- Converted below numerical variables to categorical variables by changing the data type to category
    - "policy_deductable",
    - "witnesses",
    - "number_of_vehicles_involved",
    - "bodily_injuries",
    - "number_of_vehicles_involved",

- "incident_hour_of_the_day",
- "umbrella_limit",
- "auto_year"

- **Numerical Columns:** ['months_as_customer', 'age', 'policy_number', 'policy_deductable', 'policy_annual_premium', 'umbrella_limit', 'insured_zip', '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', 'auto_year']
- **Categorical Columns:** ['policy_bind_date', 'policy_state', 'policy_csl', '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_location', 'property_damage', 'police_report_available', 'auto_make', 'auto_model', 'fraud_reported']

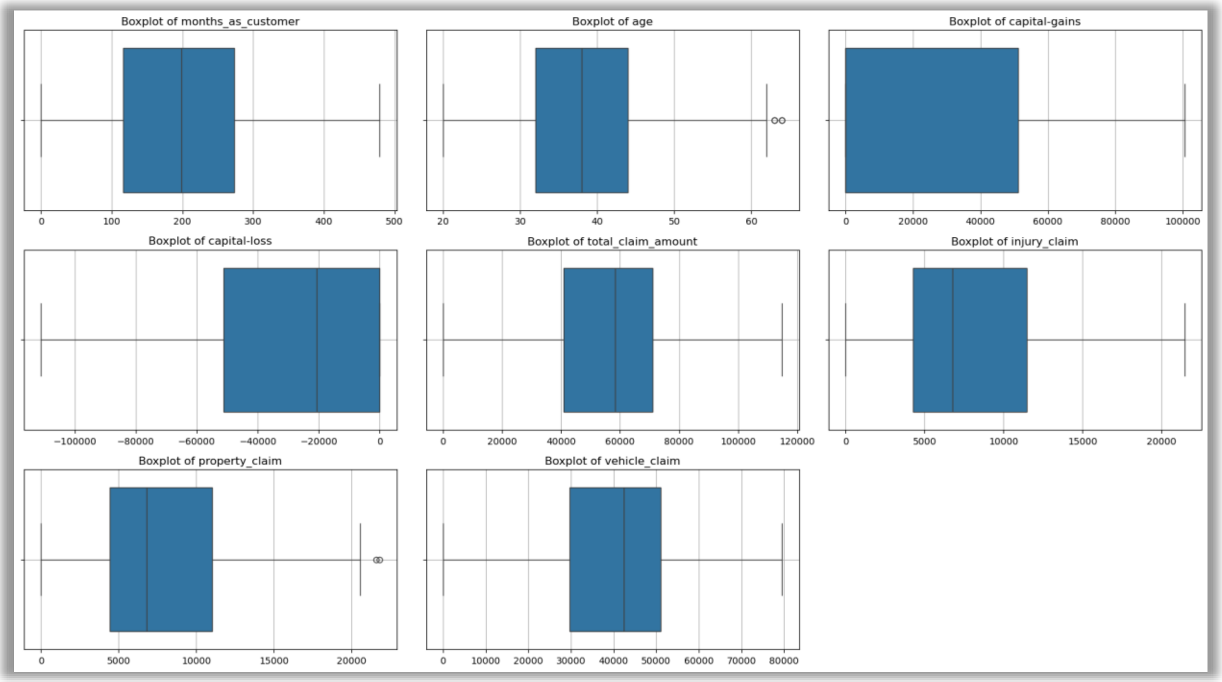
4.1.2 Visualize the distribution of selected numerical features using appropriate plots to understand their characteristics

Plotted the numerical variables in Histogram and BoxPlots



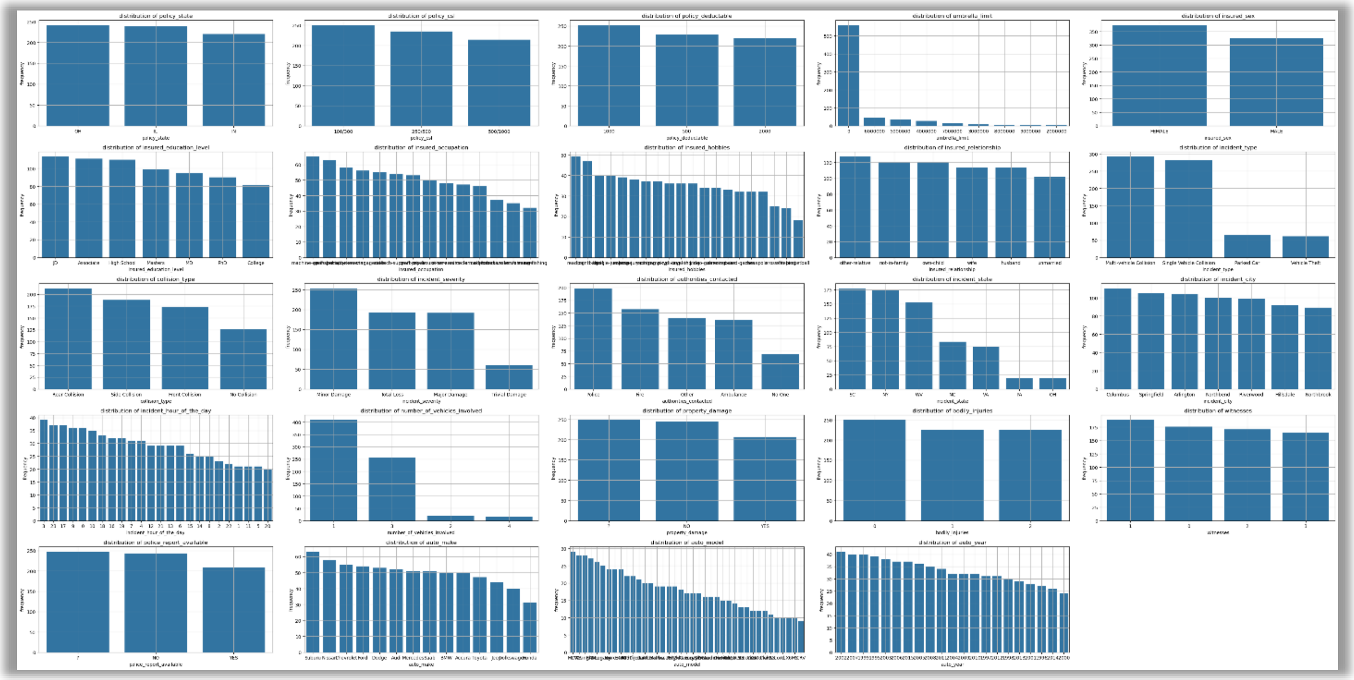
Observations: total_claim_amount, injury_claim, property_claim, vehicle_claim are very similar. It means it has multi collinearity. Considering only total_claim_amount would be better option.

Capital_gain, capital_loss are having some spikes, not distributed normally.



Observations: Overall, there is not much of outliers; small outliers present in property_claim above 20000 and age above 60

Plotted the categorical variables in CountPlots



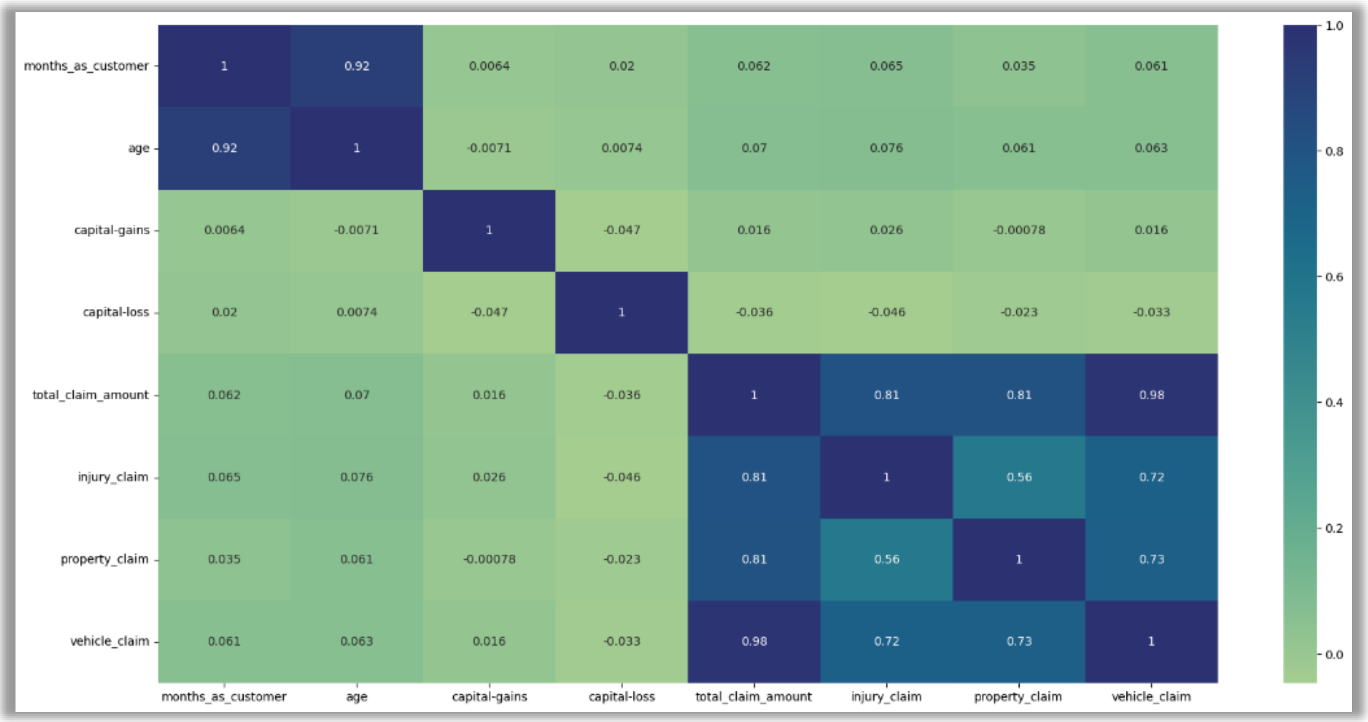
Observations:

Umbrella_limit – variable has significant percentage with zero values

Insured_Occupation, auto_model, incident_state & insured_hobbies show significant variations

4.2 Perform correlation analysis

Plotted heatmap for the numerical variables. sns.heatmap(corr_matt, annot=True, cmap="crest")



Observations:

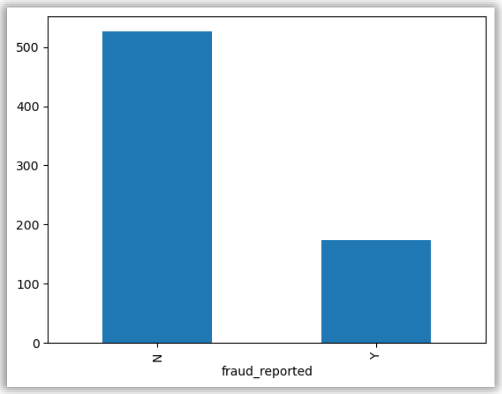
All claims (total_claim_amount, injury_claim, property_claim & vehicle_claim) are highly correlated.

4.3 Check class balance

Plot a bar chart to check class balance

y_train.value_counts().plot.bar()

plt.show()



Observations: Approximately close to 30% of incidents seem to be fraud reported

4.4 Perform bivariate analysis

4.4.1 Target likelihood analysis for categorical variables.

- Written a function **level_wise_analysis** to calculate and analyse likelihood for categorical features

- Written a function **target_likelihood_analysis** to calculate target_likelihood, coefficient of variation and low contribution flag.

- When the coeff_of_variation is less than 0.1, considered it as low contributing feature.

Observations:

['incident_city', 'insured_education_level', 'policy_deductable', 'insured_sex',

'bodily_injuries', 'policy_state', 'police_report_available'] variables are low contributing features. It can be eliminated for model preparation

(target_analysis[(target_analysis["Target_Likelihood"] > .35) & (target_analysis["Coeff_of_Variation"] > .2)]).sort_values(by="Target_Likelihood", ascending=False).head(20)

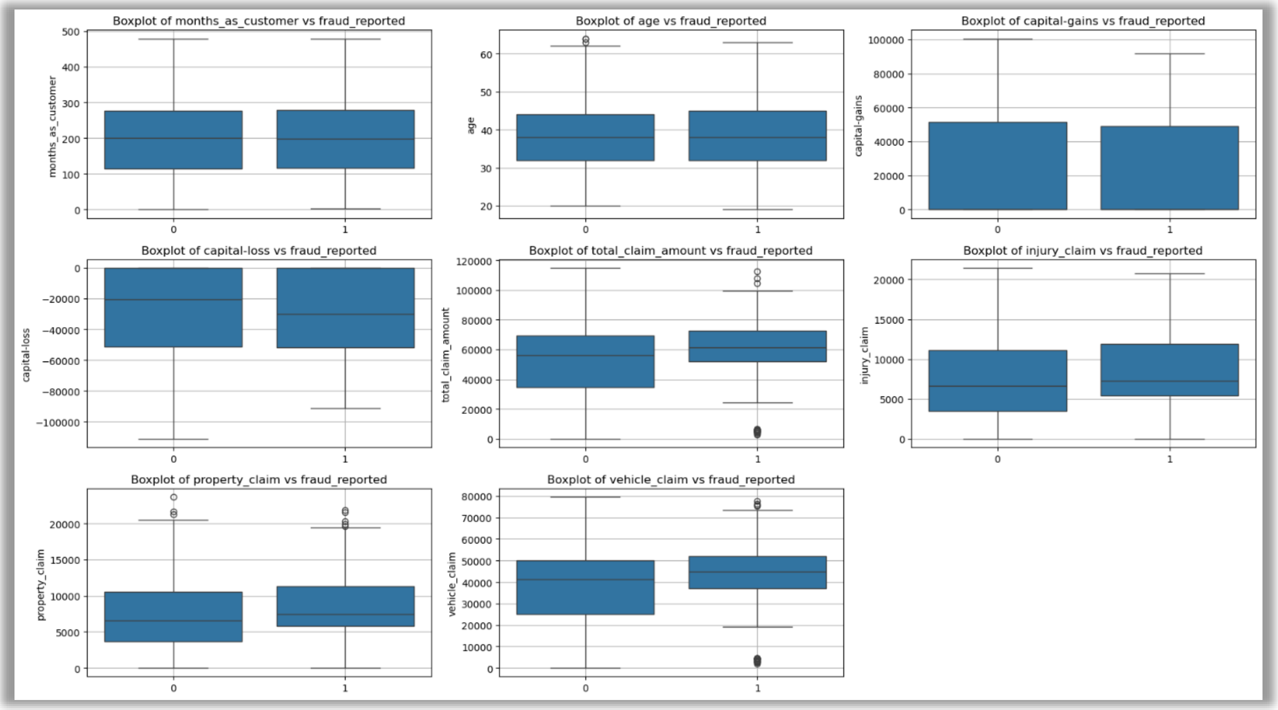
Listed all the target variables with coeff_of_variation > .2 and target_likelihood > .35 to understand the most significant contributor for the target variable.

	Feature	Category	Target_Likelihood	Sample_Size	Coeff_of_Variation	Low_Contribution
47	insured_hobbies	chess	0.83	46	0.74	False
48	insured_hobbies	cross-fit	0.74	35	0.74	False
10	umbrella_limit	2000000	0.67	3	0.41	False
76	incident_severity	Major Damage	0.61	275	1.12	False
18	umbrella_limit	10000000	0.50	2	0.41	False
168	auto_model	X6	0.44	16	0.39	False
87	incident_state	OH	0.43	23	0.30	False
162	auto_model	Silverado	0.41	22	0.39	False
178	auto_year	2004	0.41	39	0.28	False
17	umbrella_limit	9000000	0.40	5	0.41	False
154	auto_model	ML350	0.40	20	0.39	False
137	auto_model	C300	0.39	18	0.39	False
170	auto_year	1996	0.39	36	0.28	False
16	umbrella_limit	8000000	0.38	8	0.41	False
164	auto_model	Tahoe	0.38	24	0.39	False
144	auto_model	F150	0.37	27	0.39	False
31	insured_occupation	exec-managerial	0.37	76	0.24	False
140	auto_model	Civic	0.36	22	0.39	False

- Clearly some significance importance there for hobbies with chess and cross-fit
- incident severity as Major Damage has high likelihood and Coeff
- insured_occupation as exec-managerial has significance
- auto_model -> Civic, F150, Tahoe, C300 ,ML350 ,Silverado ,X6 has high likelihood to result in strong predictor

4.4.2 Explore the relationships between numerical features and the target variable to understand their impact on the target outcome using appropriate visualization techniques to identify trends and potential interactions

Done boxplot for numerical variables against target variable – **fraud_reported**



Observations:

- Total_claim_amount has slightly higher when it's fraudulent reported incident
- Similar case applied to other claims (Multicollinearity exists) (vehile_claim , property_claim, injury_claim will be contributed in total_claim_amount)
- Not much inference on other features (Features months_as_customer, age, capital-gains and capital-loss are not significantly explaining the fraud)

6. Feature Engineering

6.1 Perform resampling

RandomOverSampler technique used to balance the data and handle class imbalance.

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

# Perform resampling on training data
ros = RandomOverSampler(random_state=42)
X_train1, y_train1 = ros.fit_resample(X_train, y_train)
print(X_train1.shape)
print(y_train1.shape)
```

6.2 Feature Creation

- Deriving new feature **is_weekday** from **incident_date.**
- Applied dt.dayofweek function to categorize 0 to 4 as weekday (1) and 5 & 6 are non-weekday (0)
- Applied another logic on **incident_hour_of_the_day**, to categorize

  5 to 11 as Morning; 11 to 17 as Afternoon; 17 to 23 as Evening; 23 to 5 as Night

- Mapped target variable Y to 1 and N to 0

6.3 Handle redundant columns

"injury_claim", "property_claim", "vehicle_claim" variables are removed because of total_claim_amount has full influencing on them

"months_as_customer", "age", "capital-gains", "capital-loss" - As per section 4.4.2- there is not much variance for these variable to explain the predictability. Hence their variables are removed

"policy_csl", "auto_year" – As per section 4.1.2 count plots, there is not much variance for these variable to explain the predictability. Hence their variables are removed.

Also removing all the low_contributions variables identified in earlier section 4.4.1 – Target likelihood analysis

In summary, below features are dropped.

```
features_to_drop

['injury_claim',
 'property_claim',
 'vehicle_claim',
 'months_as_customer',
 'age',
 'capital-gains',
 'capital-loss',
 'policy_csl',
 'auto_year',
 'incident_city',
 'insured_education_level',
 'policy_deductable',
 'insured_sex',
 'bodily_injuries',
 'policy_state',
 'police_report_available']
```

6.4 Combine values in Categorical Columns

umbrella_limit → As we have seen earlier, this variable has significant percentage with zero values. Non zero values can be combined effort on target variable

incident_type → Vehicle Theft and Parked Car are treated as Others. Remaing inicident types are like collisions with another vehicle.

insured_hobbies → As we have seen in target likelihood analysis, **chess** and **cross-fit** have very significance, so other categories are combined as "Others" having comparatively less significance.

insured_occupation→ As we have seen in target likelihood analysis, **exec-managerial** has very significance, so other categories are combined as "Others" having comparatively less significance.

auto_model → As we have seen in target likelihood analysis, **"Civic", "F150", "Tahoe", "C300","ML350","Silverado" ,"X6"** have very significance, so other categories are combined as "Others" having comparatively less significance.

incident_major_severity→ As we have seen in target likelihood analysis, **Major Damage** has very significance, so other categories are combined as "Others" having comparatively less significance.

witness_more_than_one → New variable is introduced; 0 & 1 are treated in one category(NO) and 2 & 3 are treated in another category (YES) to reduce the features.

authorities_contacted → Other and **No One** can be combined together as Others.

Drop the duplicate / irrelevant columns **auto_make, incident_severity, witnesses, incident_date, incident_hour_of_the_day.**

Final revised features list is below

```
numerical_columns

['umbrella_limit', 'total_claim_amount']

categorical_columns

['insured_occupation',
 'insured_hobbies',
 'insured_relationship',
 'incident_type',
 'collision_type',
 'authorities_contacted',
 'incident_state',
 'number_of_vehicles_involved',
 'property_damage',
 'auto_model',
 'is_weekday',
 'incident_time',
 'incident_major_severity',
 'witness_more_than_one']
```

6.5.2 Create dummy variables for categorical columns in training data

Create dummy variables using the 'get_dummies' for categorical columns in training data

```python
# Create dummy variables using the 'get_dummies' for categorical columns in training data
dummies = pd.get_dummies(X_train1[categorical_columns], dtype=int, drop_first=True)
X_train2 = pd.concat([X_train1, dummies], axis=1)

X_train2.drop(categorical_columns, axis=1, inplace=True)
X_train2.head()
```

```
Index(['umbrella_limit', 'total_claim_amount',
       'insured_occupation_exec-managerial', 'insured_hobbies_chess',
       'insured_hobbies_cross-fit', 'insured_relationship_not-in-family',
       'insured_relationship_other-relative', 'insured_relationship_own-child',
       'insured_relationship_unmarried', 'insured_relationship_wife',
       'incident_type_Others', 'incident_type_Single Vehicle Collision',
       'collision_type_No Collision', 'collision_type_Rear Collision',
       'collision_type_Side Collision', 'authorities_contacted_Fire',
       'authorities_contacted_Others', 'authorities_contacted_Police',
       'incident_state_NY', 'incident_state_OH', 'incident_state_PA',
       'incident_state_SC', 'incident_state_VA', 'incident_state_WV',
       'number_of_vehicles_involved_2', 'number_of_vehicles_involved_3',
       'number_of_vehicles_involved_4', 'property_damage_NO',
       'property_damage_YES', 'auto_model_Civic', 'auto_model_F150',
       'auto_model_ML350', 'auto_model_Others', 'auto_model_Silverado',
       'auto_model_Tahoe', 'auto_model_X6', 'is_weekday_1',
       'incident_time_Evening', 'incident_time_Morning', 'incident_time_Night',
       'incident_major_severity_YES', 'witness_more_than_one_YES'],
      dtype='object')
```

In total it turns out to be 42 features.

6.5.3 Create dummy variables for categorical columns in validation data

Repeated the same process as above for X_test1

6.5.3 Create dummy variables for categorical columns in validation data [2 Marks]

```python
# Create dummy variables using the 'get_dummies' for categorical columns in validation data
dummies_test = pd.get_dummies(X_test[categorical_columns], dtype=int, drop_first=True)
X_test1 = pd.concat([X_test, dummies_test], axis=1)

X_test1.drop(categorical_columns, axis=1, inplace=True)
```

6.5.4 Create dummy variable for dependent feature in training and validation data

Mapped 1 for Y and 0 for N

```python
# Create dummy variable for dependent feature in training data
def binary_map(X):
    return X.map({"Y": 1, "N": 0})

# Create dummy variable for dependent feature in validation data
y_test = binary_map(y_test)
```

6.6 Feature scaling

Imported StandardScaler

And performed numerical features scaling for train and test data

```
# Import the necessary scaling tool from scikit-learn
from sklearn.preprocessing import StandardScaler  # (var-mean)/(std)

# Scale the numeric features present in the training data
scaler = StandardScaler()
X_train2[numerical_columns] = scaler.fit_transform(X_train2[numerical_columns])

# Scale the numeric features present in the validation data
X_test1[numerical_columns] = scaler.transform(X_test1[numerical_columns])
```

7. Model Building

7.1 Feature selection

- Imported all necessary libraries

7.1.2 Perform feature selection

logreg = LogisticRegression(random_state=42)

Created logisticregression with random_state=42, n_features_to_select=50 and min_features_to_select=12

Got the rfe supported features

```
X_train2.columns[rfecv.support_  ]
```

```
Index(['insured_hobbies_chess', 'insured_hobbies_cross-fit',
       'insured_relationship_own-child', 'collision_type_Rear Collision',
       'incident_state_PA', 'incident_state_VA',
       'number_of_vehicles_involved_2', 'number_of_vehicles_involved_4',
       'property_damage_NO', 'auto_model_Civic', 'auto_model_Others',
       'incident_major_severity_YES', 'witness_more_than_one_YES'],
      dtype='object')
```

7.1.2 Retain the selected features

```
# Put columns selected by RFECV into variable 'col'
col = X_train2.columns[rfecv.support_]
print(col)
```

```
Index(['insured_hobbies_chess', 'insured_hobbies_cross-fit',
       'insured_relationship_own-child', 'collision_type_Rear Collision',
       'incident_state_PA', 'incident_state_VA',
       'number_of_vehicles_involved_2', 'number_of_vehicles_involved_4',
       'property_damage_NO', 'auto_model_Civic', 'auto_model_Others',
       'incident_major_severity_YES', 'witness_more_than_one_YES'],
      dtype='object')
```

7.2 Build Logistic Regression Model

Imported statsmodels.api, add constant

```
# Fit a logistic Regression model on X_train after adding a constant and output the summary
X_train_sm = sm.add_constant(X_train_sm)
ols_rfe = sm.GLM(y_train1, X_train_sm, family=sm.families.Binomial())
res_rfe = ols_rfe.fit()
pprint(res_rfe.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:         fraud_reported   No. Observations:                 1052
Model:                            GLM   Df Residuals:                     1038
Model Family:                Binomial   Df Model:                           13
Link Function:                  Logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -364.63
Date:                Sun, 10 Aug 2025   Deviance:                       729.27
Time:                        03:17:34   Pearson chi2:                 5.45e+03
No. Iterations:                     7   Pseudo R-squ. (CS):             0.5000
Covariance Type:            nonrobust
==================================================================================================
                                     coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------------------------
const                             -2.3961      0.330     -7.268      0.000      -3.042      -1.750
insured_hobbies_chess              5.9952      0.643      9.324      0.000       4.735       7.255
insured_hobbies_cross-fit          3.9656      0.496      7.998      0.000       2.994       4.937
insured_relationship_own-child    -0.6985      0.273     -2.556      0.011      -1.234      -0.163
collision_type_Rear Collision      0.7988      0.211      3.786      0.000       0.385       1.212
incident_state_PA                 -0.9687      0.657     -1.475      0.140      -2.256       0.319
incident_state_VA                  0.7616      0.321      2.372      0.018       0.132       1.391
number_of_vehicles_involved_2      0.9567      0.477      2.004      0.045       0.021       1.892
number_of_vehicles_involved_4     -1.0816      0.609     -1.775      0.076      -2.276       0.113
property_damage_NO                -0.6285      0.215     -2.928      0.003      -1.049      -0.208
auto_model_Civic                   2.0419      0.634      3.219      0.001       0.799       3.285
auto_model_Others                 -0.3266      0.270     -1.211      0.226      -0.855       0.202
incident_major_severity_YES        3.7492      0.218     17.230      0.000       3.323       4.176
witness_more_than_one_YES          0.6996      0.196      3.570      0.000       0.315       1.084
==================================================================================================
```

Model is built, looking into summary, **"number_of_vehicles_involved_4", "auto_model_Others", "incident_state_PA"** feature's P value is high, so we can remove and re-run the model

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:         fraud_reported   No. Observations:                 1052
Model:                            GLM   Df Residuals:                     1041
Model Family:                Binomial   Df Model:                           10
Link Function:                  Logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -368.01
Date:                Sun, 10 Aug 2025   Deviance:                       736.03
Time:                        03:17:34   Pearson chi2:                 4.34e+03
No. Iterations:                     7   Pseudo R-squ. (CS):             0.4967
Covariance Type:            nonrobust
====================================================================================================
                                       coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------------------------
const                               -2.6954      0.237    -11.374      0.000      -3.160      -2.231
insured_hobbies_chess                5.7795      0.620      9.318      0.000       4.564       6.995
insured_hobbies_cross-fit            3.8729      0.485      7.985      0.000       2.922       4.823
insured_relationship_own-child      -0.6737      0.271     -2.490      0.013      -1.204      -0.143
collision_type_Rear Collision        0.7540      0.209      3.609      0.000       0.344       1.163
incident_state_VA                    0.8146      0.316      2.581      0.010       0.196       1.433
number_of_vehicles_involved_2        0.9520      0.474      2.007      0.045       0.022       1.882
property_damage_NO                  -0.5989      0.212     -2.822      0.005      -1.015      -0.183
auto_model_Civic                     2.3383      0.591      3.960      0.000       1.181       3.496
incident_major_severity_YES          3.7461      0.214     17.502      0.000       3.327       4.166
witness_more_than_one_YES            0.6790      0.194      3.503      0.000       0.299       1.059
====================================================================================================
```

Now all P values are good. Final GLM model – res_rfe1

7.2.3 Evaluate VIF of features to assess multicollinearity

None of the value is above 5, so good to proceed

	Features	VIF
0	const	4.74
9	incident_major_severity_YES	1.07
1	insured_hobbies_chess	1.06
7	property_damage_NO	1.04
2	insured_hobbies_cross-fit	1.02
5	incident_state_VA	1.02
10	witness_more_than_one_YES	1.02
3	insured_relationship_own-child	1.01
4	collision_type_Rear Collision	1.01
6	number_of_vehicles_involved_2	1.01
8	auto_model_Civic	1.01

7.2.4 Make predictions on training data

Predict the probabilities on the training data

y_train_pred = res_rfe1.predict(X_train_sm1)

output → array([0.84936937, 0.12548411, 0.06815427, ..., 0.85871162, 0.13289262,

0.93152731])

7.2.5 Create a DataFrame that includes actual fraud reported flags, predicted probabilities, and a column indicating predicted classifications based on a cutoff value of 0.5

```
# Create a new DataFrame containing the actual fraud reported flag and the probabilities predicted by the model

y_train_pred_final = pd.DataFrame(
    {"fraud_reported": y_train1.values, "fraud_reported_probabilities": y_train_pred}
)

# Create new column indicating predicted classifications based on a cutoff value of 0.5
y_train_pred_final["predicted"] = y_train_pred_final[
    "fraud_reported_probabilities"
].apply(lambda x: 1 if x > 0.5 else 0)
y_train_pred_final.head()
```

	fraud_reported	fraud_reported_probabilities	predicted
0	0	0.849369	1
1	0	0.125484	0
2	0	0.068154	0
3	0	0.068164	0
4	0	0.220543	0

7.2.6 Check the accuracy of the model

Accuracy comes to 0.877 which is good model prediction.

7.2.7 Create a confusion matrix based on the predictions made on the training data

array([[446, 80],

[49, 477]], dtype=int64)

7.2.8 Create variables for true positive, true negative, false positive and false negative

Create variables for true positive, true negative, false positive and false negative

tn, fp, fn, tp = confusion.ravel()

print(tn, fp, fn, tp)

446 80 49 477

__7.2.9__ Calculate sensitivity, specificity, precision, recall and F1-score

Sensitivity : 0.91

Specificity : 0.85

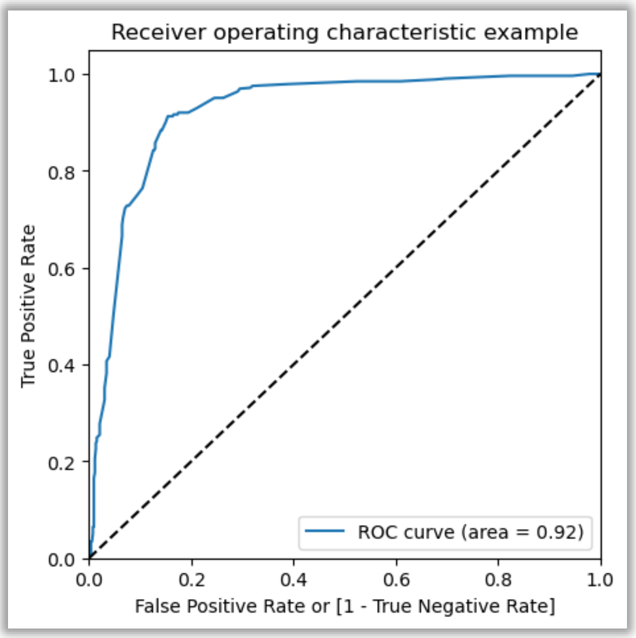
Precision: 0.86

Recall: 0.91

F1 Score: 0.88

7.3 Find the Optimal Cutoff

__7.3.1__ Plot ROC Curve to visualize the trade-off between true positive rate and false positive rate across different classification thresholds



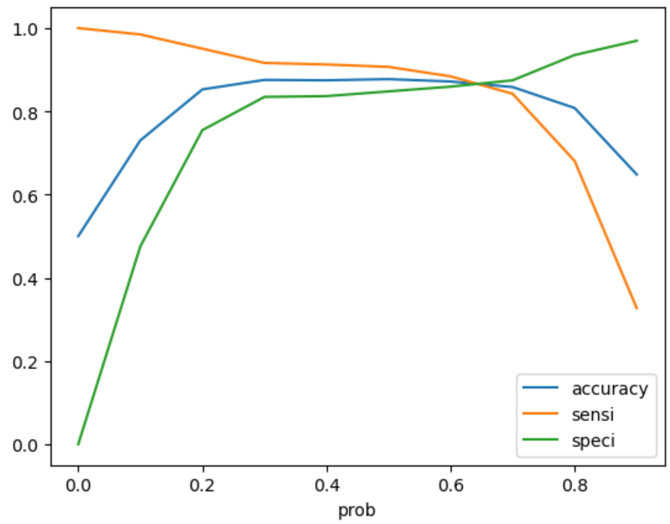
ROC cure are is 0.92

__7.3.2__ Predict on training data at various probability cutoffs

Finding probability at a cut off, by measuring between 0, 0.1, .2 …….,0.9

7.3.3 Plot accuracy, sensitivity, specificity at different values of probability cutoffs

	prob	accuracy	sensi	speci
0.0	0.0	0.500000	1.000000	0.000000
0.1	0.1	0.730038	0.984791	0.475285
0.2	0.2	0.852662	0.950570	0.754753
0.3	0.3	0.875475	0.916350	0.834601
0.4	0.4	0.874525	0.912548	0.836502
0.5	0.5	0.877376	0.906844	0.847909
0.6	0.6	0.871673	0.884030	0.859316
0.7	0.7	0.858365	0.842205	0.874525
0.8	0.8	0.807985	0.680608	0.935361
0.9	0.9	0.648289	0.326996	0.969582



By looking at the plot, 0.6 would be optimal cut off.

7.3.4 Create a column for final prediction based on optimal cutoff

	fraud_reported	fraud_reported_probabilities	predicted	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	final_predicted
0	0	0.849369	1	1	1	1	1	1	1	1	1	1	0	1
1	0	0.125484	0	1	1	0	0	0	0	0	0	0	0	0
2	0	0.068154	0	1	0	0	0	0	0	0	0	0	0	0
3	0	0.068164	0	1	0	0	0	0	0	0	0	0	0	0
4	0	0.220543	0	1	1	1	0	0	0	0	0	0	0	0

7.3.5 Calculate the accuracy

Accuracy comes to 0.87

7.3.6 Create confusion matrix

array([[452, 74],

[61, 465]], dtype=int64)

7.3.7 Create variables for true positive, true negative, false positive and false negative

tn, fp, fn, tp = cm2.ravel()

7.3.8 Calculate sensitivity, specificity, precision, recall and F1-score of the model

Sensitivity : 0.88

Specificity : 0.86

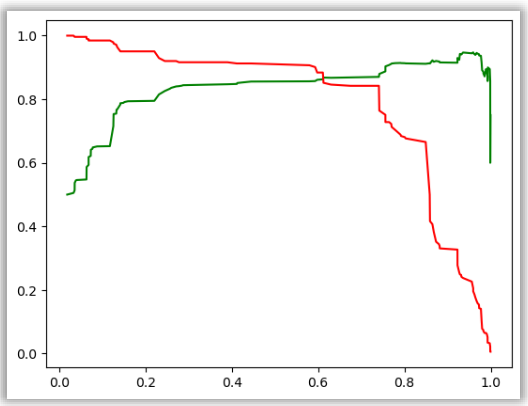
Precision: 0.86

Recall: 0.88

F1 Score: 0.87

7.3.9 Plot precision-recall curve

Here again 0.6 is the cut off for predictability



Precision recall threshold cuts over around 0.6 (likely little less);

Let's consider best cut off as 0.5 for prediction model

7.4 Build Random Forest Model

7.4.2 Build the random forest model

RandomForestClassifier

RandomForestClassifier(n_jobs=-1, oob_score=True, random_state=42)

7.4.3 Get feature importance scores and select important features

	Varname	Imp
0	incident_major_severity_YES	0.227161
1	total_claim_amount	0.119309
2	insured_hobbies_chess	0.096347
3	insured_hobbies_cross-fit	0.031293
4	witness_more_than_one_YES	0.026690
5	is_weekday_1	0.023967
6	property_damage_NO	0.022632
7	incident_state_SC	0.019429
8	incident_time_Night	0.018784
9	incident_state_WV	0.018681
10	umbrella_limit	0.018311
11	insured_relationship_own-child	0.018168
12	auto_model_Others	0.018029
13	property_damage_YES	0.017953
14	collision_type_Rear Collision	0.017618
15	incident_type_Single Vehicle Collision	0.017544
16	insured_relationship_not-in-family	0.017368
17	authorities_contacted_Police	0.017244

Select features with high importance scores (imp >=0.015)

best_features = imp_df[imp_df["Imp"] >= 0.015]["Varname"]

```
best_features

0                 incident_major_severity_YES
1                          total_claim_amount
2                       insured_hobbies_chess
3                   insured_hobbies_cross-fit
4                   witness_more_than_one_YES
5                                is_weekday_1
6                          property_damage_NO
7                           incident_state_SC
8                         incident_time_Night
9                           incident_state_WV
10                             umbrella_limit
11             insured_relationship_own-child
12                          auto_model_Others
13                        property_damage_YES
14              collision_type_Rear Collision
15     incident_type_Single Vehicle Collision
16         insured_relationship_not-in-family
17               authorities_contacted_Police
18                      incident_time_Evening
19                number_of_vehicles_involved_3
20               authorities_contacted_Others
21               collision_type_Side Collision
22                  insured_relationship_wife
23                 authorities_contacted_Fire
24                          incident_state_NY
Name: Varname, dtype: object
```

7.4.4 *Train the model with selected features*

```
# Fit the model on the training data with selected features

rf = RandomForestClassifier(random_state=42, n_jobs=-1, oob_score=True)

rf.fit(X_train_best_features, y_train1)
```

rf.oob_score_ = **0.9306083650190115**

7.4.5 Generate predictions on the training data

The accuracy of the model comes to 1.0

7.4.7 *Create confusion matrix*

```
array([[526,   0],
       [  0, 526]], dtype=int64)
```

7.4.9 *Calculate sensitivity, specificity, precision, recall and F1-score of the model*

Sensitivity : 1.0

Specificity : 1.0

Precision: 1.0

Recall: 1.0

F1 Score: 1.0

7.4.10 *Check if the model is overfitting training data using cross validation*

[0.88625592 0.93364929 0.93333333 0.92380952 0.94285714]

mean accuracy = 0.9239810426540286 (**so no overfitting here**)

7.5 Hyperparameter Tuning

Use grid search to find the best hyperparamter values

params = {

"max_depth": [3, 5, 10, 15, 18, 20, 22],

"min_samples_leaf": [9, 10, 11, 12, 15, 20, 23, 50, 100, 200],

"n_estimators": [10, 25, 50, 100],

"max_features": [10, 15, 20, 25,30,40,50],

}

rf_best - model

```
Fitting 5 folds for each of 1960 candidates, totalling 9800 fits
GridSearchCV
best_estimator_: RandomForestClassifier
RandomForestClassifier
RandomForestClassifier(max_depth=10, max_features=20, min_samples_leaf=9,
                       n_estimators=10, n_jobs=-1, random_state=42)
```

7.5.3 Make predictions on training data

train_accuracy is 0.8982889733840305 or 0.90

7.5.5 Create confusion matrix

array([[453, 73],

[34, 492]], dtype=int64)

7.5.7 Calculate sensitivity, specificity, precision, recall and F1-score of the model

Sensitivity : 0.94

Specificity : 0.86

Precision: 0.87

Recall: 0.94

F1 Score: 0.90

8. Prediction and Model Evaluation

8.1 Make predictions over validation data using logistic regression model

8.1.1 Select relevant features for validation data and add constant

GLM Model – **res_rfe1**

```
Index(['insured_hobbies_chess', 'insured_hobbies_cross-fit',
       'insured_relationship_own-child', 'collision_type_Rear Collision',
       'incident_state_VA', 'number_of_vehicles_involved_2',
       'property_damage_NO', 'auto_model_Civic', 'incident_major_severity_YES',
       'witness_more_than_one_YES'],
      dtype='object')
```

```
# Select the relevant features for validation data
X_test_best = X_test1[col]

# Add constant to X_validation
X_test_sm = sm.add_constant(X_test_best)
```

8.1.2 Make predictions over validation data

	Test_actual	Validation
0	0	0.231141
1	0	0.063243
2	0	0.117486
3	0	0.068154
4	0	0.134535
...	...	...
295	0	0.989197
296	1	0.126060
297	0	0.125484
298	0	0.063243
299	0	0.063554

300 rows × 2 columns

8.1.4 Make final prediction based on cutoff value

8.1.4 Make final prediction based on cutoff value [1 Mark]

```
# Make final predictions on the validation data using the optimal cutoff
predictions["Predicted"] = predictions["Validation"].apply(
    lambda x: 1 if x > 0.5 else 0
)
predictions.head()
```

	Test_actual	Validation	Predicted
0	0	0.231141	0
1	0	0.063243	0
2	0	0.117486	0
3	0	0.068154	0
4	0	0.134535	0

8.1.5 Check the accuracy of logistic regression model on validation data

Test_accuracy is 0.83

8.1.6 Create confusion matrix

array([[185, 41],

[10, 64]], dtype=int64)

8.1.8 Calculate sensitivity, specificity, precision, recall and f1 score of the model

Sensitivity : 0.86

Specificity : 0.82

Precision: 0.61

Recall: 0.86

F1 Score: 0.72

8.2 Make predictions over validation data using random forest model

8.2.1 Select the important features and make predictions over validation data

8.2.1 Select the important features and make predictions over validation data

```
# Select the relevant features for validation data
X_test_best = X_test1[X_train_best_features.columns]

# Make predictions on the validation data
y_test_rf = rf_best.predict(X_test_best)
y_test_rf[:10]
```

```
array([0, 0, 0, 0, 0, 1, 0, 0, 0, 1], dtype=int64)
```

```
X_train_best_features.columns
```

```
Index(['incident_major_severity_YES', 'total_claim_amount',
       'insured_hobbies_chess', 'insured_hobbies_cross-fit',
       'witness_more_than_one_YES', 'is_weekday_1', 'property_damage_NO',
       'incident_state_SC', 'incident_time_Night', 'incident_state_WV',
       'umbrella_limit', 'insured_relationship_own-child', 'auto_model_Others',
       'property_damage_YES', 'collision_type_Rear Collision',
       'incident_type_Single Vehicle Collision',
       'insured_relationship_not-in-family', 'authorities_contacted_Police',
       'incident_time_Evening', 'number_of_vehicles_involved_3',
       'authorities_contacted_Others', 'collision_type_Side Collision',
       'insured_relationship_wife', 'authorities_contacted_Fire',
       'incident_state_NY'],
      dtype='object')
```

test_accuracy = 0.84

8.2.5 Calculate sensitivity, specificity, precision, recall and F1-score of the model

Sensitivity : 0.82; Specificity : 0.85;Precision: 0.64; Recall: 0.82; F1 Score: 0.72

9. Best Model Conclusion

Logistic Regression	Random Forest
Base model: res_rfe1 **(Section 7.2)** Train_accuracy = 0.88 Sensitivity : 0.91 Specificity : 0.85 Precision: 0.86 Recall: 0.91 F1 Score: 0.88 **Optimal cut off at 0.6 (Section 7.3.5)** Model: res_rfe1 Train_accuracy = 0.87 Sensitivity : 0.88 Specificity : 0.86 Precision: 0.86 Recall: 0.88 F1 Score: 0.87	Base model: rf **(Section 7.4)** oob_score_ = 0.93 Train_accuracy = 1 Sensitivity : 1.0 Specificity : 1.0 Precision: 1.0 Recall: 1.0 F1 Score: 1.0 mean accuracy = 0.92 **Hyperparameter Tuning (Section 7.5)** Model: rf_best Train_accuracy = 0.90 Sensitivity : 0.94 Specificity : 0.86 Precision: 0.87 Recall: 0.94 F1 Score: 0.90
GLM Prediction (Section: 8.1) Model: res_rfe1 Test_accuracy = 0.83 Sensitivity : 0.86 Specificity : 0.82 Precision: 0.61 Recall: 0.86 F1 Score: 0.72	**RF Prediction (Section: 8.2)** Model: rf_best Test_accuracy = 0.84 Sensitivity : 0.82 Specificity : 0.85 Precision: 0.64 Recall: 0.82 F1 Score: 0.72

Inference:

- We have got very close result in logistic and random forest models. Both are close to 0.85 to .90 accuracy even in training and test data.
- Test accuracy for logistic and random forest prediction is also almost same .83 and .84 respectively.
- It will be a close call to take right decision. We are performing fraud deduction which means, identifying fraud person is very important though we include some false positive identified person. This implies **Sensitivity** should be higher
- **GLM Prediction sensitivity** (0.86) is higher than **RF Prediction sensitivity** (0.81), so I suggest to go with ***LGR mode rse_rfe1 as best model for consideration***