University of New Haven

TAGLIATELA COLLEGE OF ENGINEERING

Electrical & Computer Engineering and Computer Science

Computer Engineering

Fraud Detection

CONTENTS

Executive Summary

Currently, insurance industries collect more than one trillion dollars of premiums each year. The total cost of an insurance fraud is estimated to be more than forty billion dollars. So, detection of an insurance fraud is a challenging problem for the insurance industry.
The auto insurance fraud is the most prominent type of insurance fraud, which can be done by fake accident claim. In this project, we are focusing on detecting the fraudulent claims by implementing various models using Machine learning techniques

Team Members:

Chandrasheker Bassetti - 00761756
Divya Madhuri Cheernapally - 00760724
Rishitha Ravikumar - 00762285

Questions?

Contact: Chandrasheker Bassetti
cbass6@unh.newhaven.edu
610-357-7103

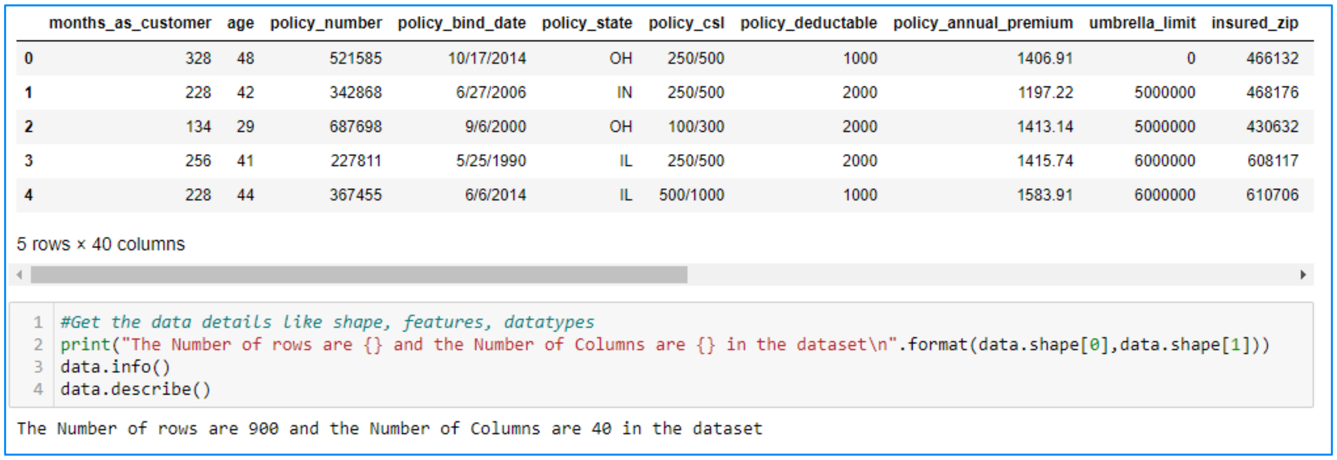
Problem Statement

The goal of the project is to build a model to detect the auto insurance fraud. The main challenge in building Machine Learning model is that the data for fraud samples is less compared with the legit insurance claims.

Our approach for this project is as per the following steps:

- Identify and collect data with necessary attributes
- EDA & Perform necessary data cleansing
- Segregate data into training set and testing set
- Select appropriate algorithm
- Executing best training algorithm with testing data
- Deployment

Identify and collect data with necessary attributes



	months_as_customer	age	policy_number	policy_bind_date	policy_state	policy_csl	policy_deductable	policy_annual_premium	umbrella_limit	insured_zip
0	328	48	521585	10/17/2014	OH	250/500	1000	1406.91	0	466132
1	228	42	342868	6/27/2006	IN	250/500	2000	1197.22	5000000	468176
2	134	29	687698	9/6/2000	OH	100/300	2000	1413.14	5000000	430632
3	256	41	227811	5/25/1990	IL	250/500	2000	1415.74	6000000	608117
4	228	44	367455	6/6/2014	IL	500/1000	1000	1583.91	6000000	610706

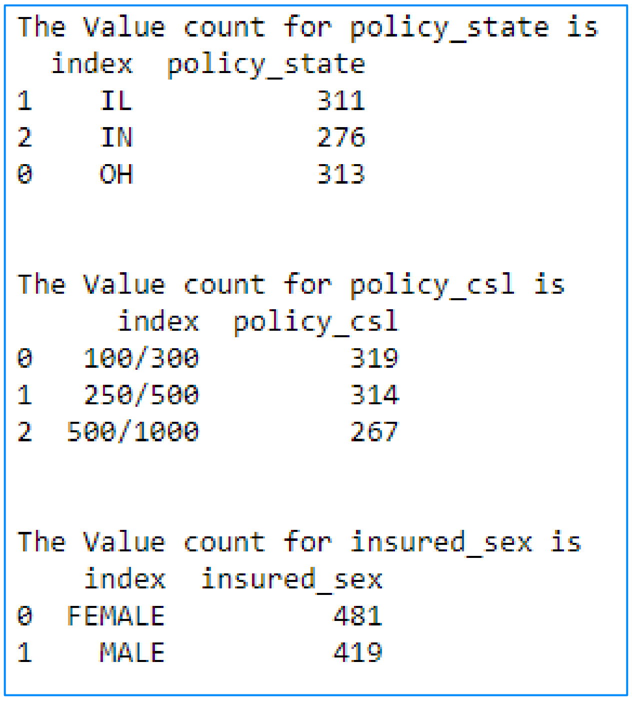
5 rows × 40 columns

```python
#Get the data details like shape, features, datatypes
print("The Number of rows are {} and the Number of Columns are {} in the dataset\n".format(data.shape[0],data.shape[1]))
data.info()
data.describe()
```

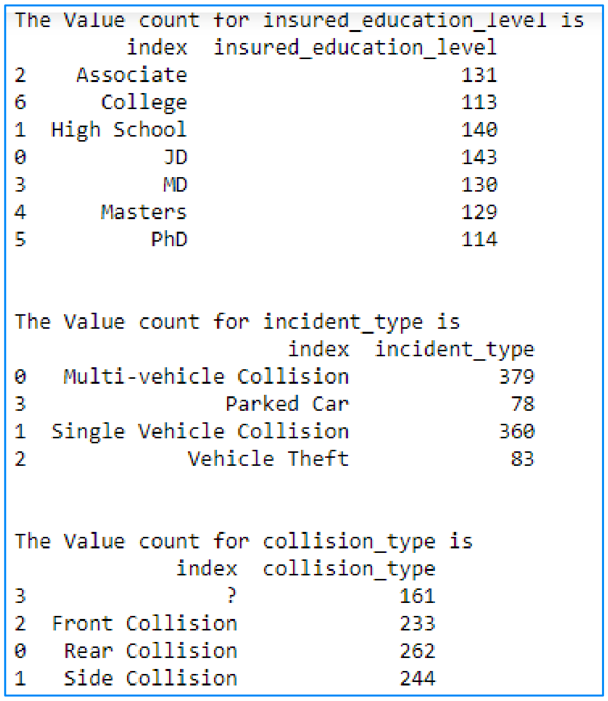
```
The Number of rows are 900 and the Number of Columns are 40 in the dataset
```

The Data set has 900 records having 40 columns including 39 features & 1 target variable. The column "_c39" has Null values and can be dropped. We can perform EDA and can remove some of the unwanted features like policy number, insured_zip etc.. which will not have any contribution to the target variable. The data contains Numeric, Date, categorical datatype variables. The average number of months customer with this company is 202 and the average age is 39.

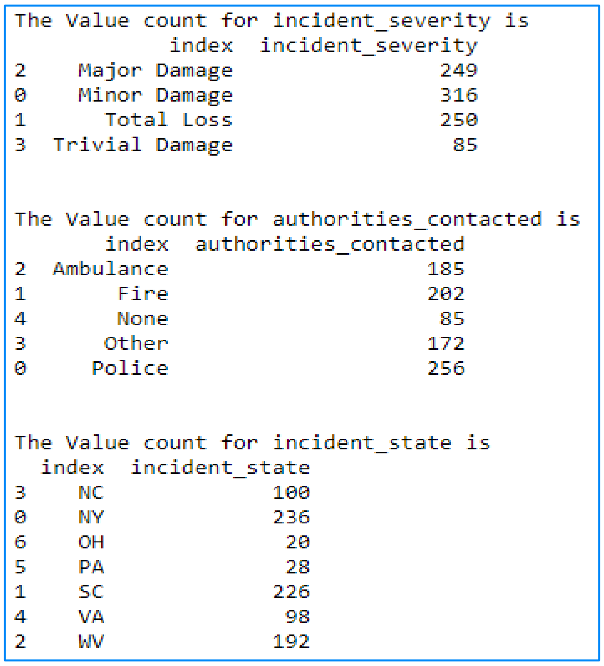
Exploratory Data Analysis & Data Cleaning



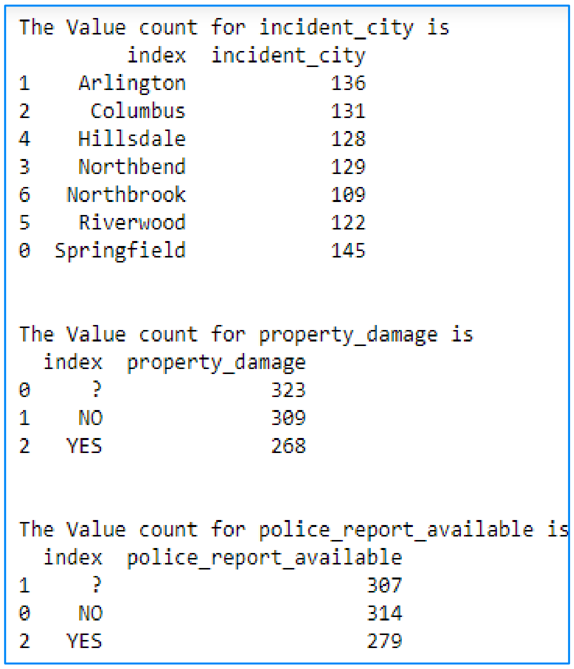
```
The Value count for policy_state is
  index  policy_state
1    IL           311
2    IN           276
0    OH           313


The Value count for policy_csl is
       index  policy_csl
0   100/300          319
1   250/500          314
2  500/1000          267


The Value count for insured_sex is
    index  insured_sex
0  FEMALE          481
1    MALE          419
```



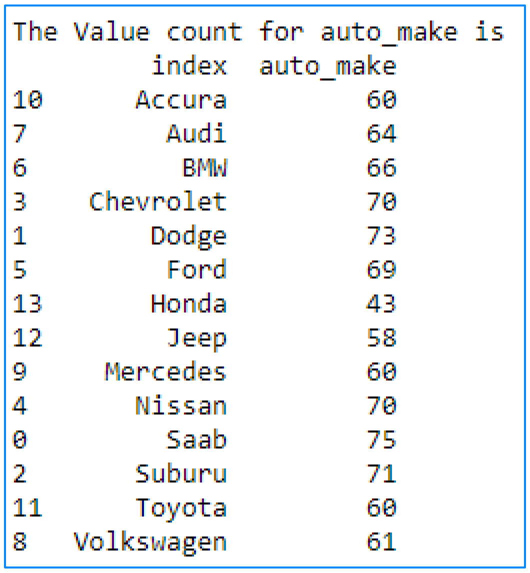
```
The Value count for insured_education_level is
         index  insured_education_level
2    Associate                      131
6      College                      113
1  High School                      140
0           JD                      143
3           MD                      130
4      Masters                      129
5          PhD                      114


The Value count for incident_type is
                      index  incident_type
0    Multi-vehicle Collision            379
3                 Parked Car             78
1   Single Vehicle Collision            360
2              Vehicle Theft             83


The Value count for collision_type is
             index  collision_type
3                ?             161
2  Front Collision             233
0   Rear Collision             262
1   Side Collision             244
```



```
The Value count for incident_severity is
            index  incident_severity
2    Major Damage                249
0    Minor Damage                316
1      Total Loss                250
3  Trivial Damage                 85


The Value count for authorities_contacted is
        index  authorities_contacted
2  Ambulance                     185
1       Fire                     202
4       None                      85
3      Other                     172
0     Police                     256


The Value count for incident_state is
  index  incident_state
3    NC             100
0    NY             236
6    OH              20
5    PA              28
1    SC             226
4    VA              98
2    WV             192
```



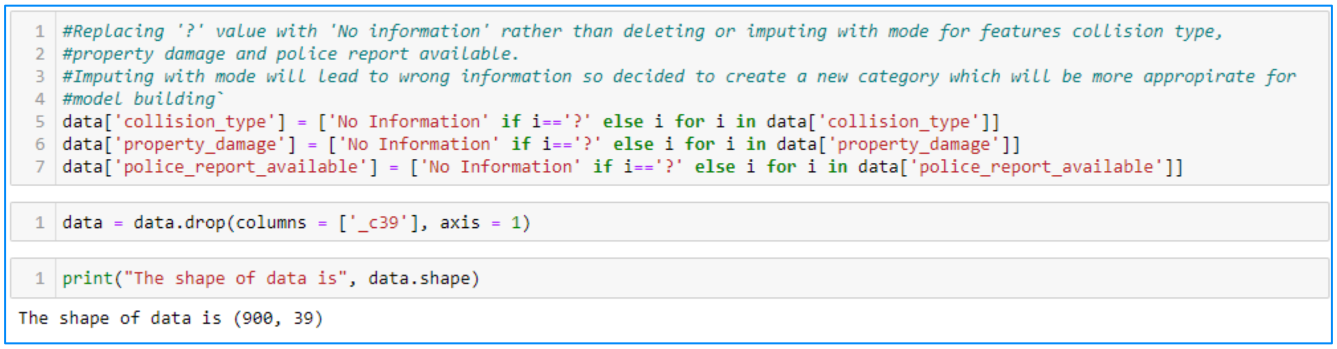
```
The Value count for incident_city is
        index  incident_city
1   Arlington           136
2    Columbus           131
4   Hillsdale           128
3   Northbend           129
6  Northbrook           109
5   Riverwood           122
0 Springfield           145


The Value count for property_damage is
  index  property_damage
0     ?              323
1    NO              309
2   YES              268


The Value count for police_report_available is
  index  police_report_available
1     ?                      307
0    NO                      314
2   YES                      279
```



```
The Value count for auto_make is
         index  auto_make
10      Accura         60
7         Audi         64
6          BMW         66
3    Chevrolet         70
1        Dodge         73
5         Ford         69
13       Honda         43
12        Jeep         58
9     Mercedes         60
4       Nissan         70
0         Saab         75
2       Suburu         71
11      Toyota         60
8   Volkswagen         61
```

When we explore into each category, we get the value counts as shown above. With this we have also made the following observations.

- There are No null values in the dataset except _c39 column which will be removed.
- Features like collision type, property damage, police report available has '?' for some of the records which needs to be imputed with proper data.
- Policy is purchased in IL, IN and OH states with almost equal distribution.
- Both Male & Female are the policy owners.
- Policy Owners Age range from 19 to 64.
- Most of the Claims involved vehicle collision rather than Theft or in Parking.
- Majority of the claims involved the Major Damage/Total Loss.
- More Incidents of fraud occurred at New York and South Carolina states.
- Irrespective of the Car Model, claims were reported.
- 70% of the claims were reported as not fraud.



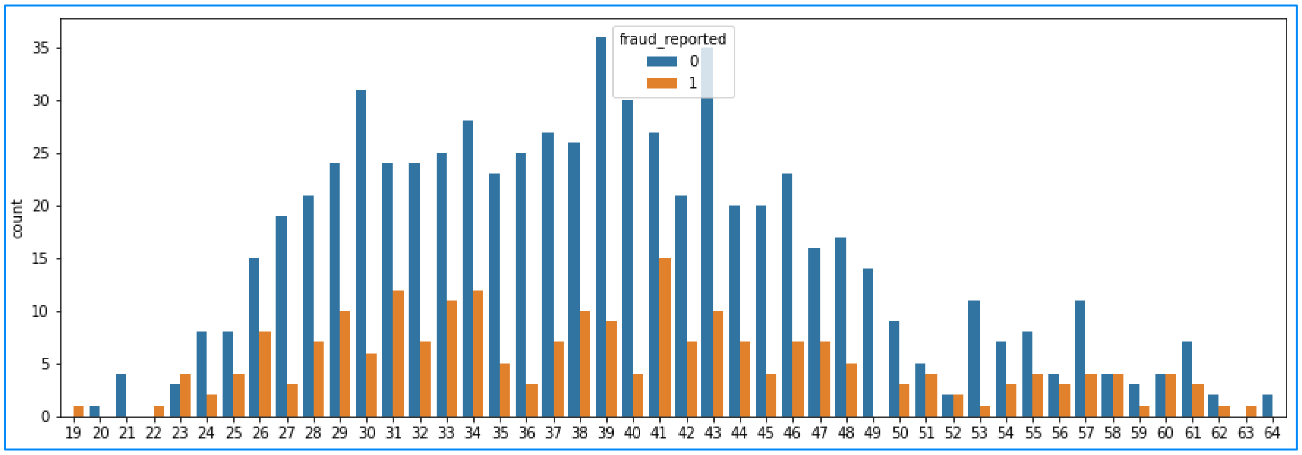
```
#Replacing '?' value with 'No information' rather than deleting or imputing with mode for features collision type,
#property damage and police report available.
#Imputing with mode will lead to wrong information so decided to create a new category which will be more appropirate for
#model building`
data['collision_type'] = ['No Information' if i=='?' else i for i in data['collision_type']]
data['property_damage'] = ['No Information' if i=='?' else i for i in data['property_damage']]
data['police_report_available'] = ['No Information' if i=='?' else i for i in data['police_report_available']]
```

```
data = data.drop(columns = ['_c39'], axis = 1)
```

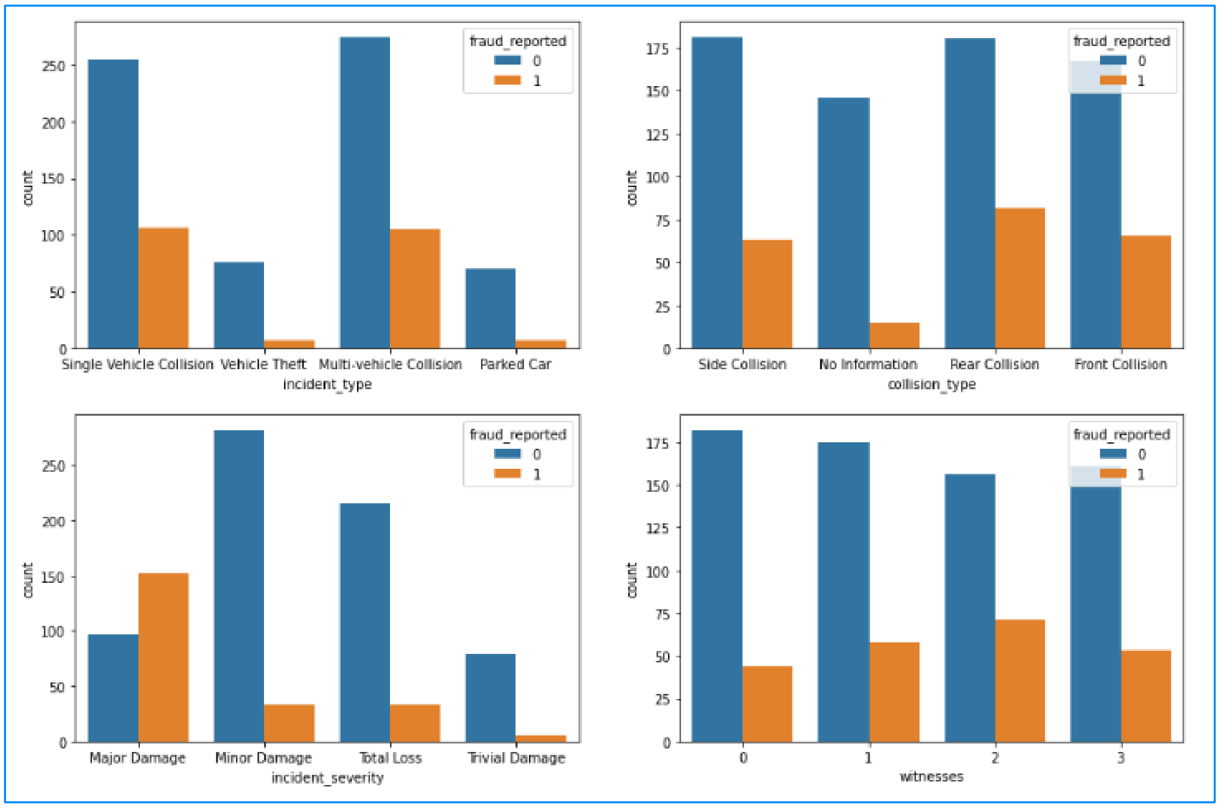
```
print("The shape of data is", data.shape)
```

```
The shape of data is (900, 39)
```

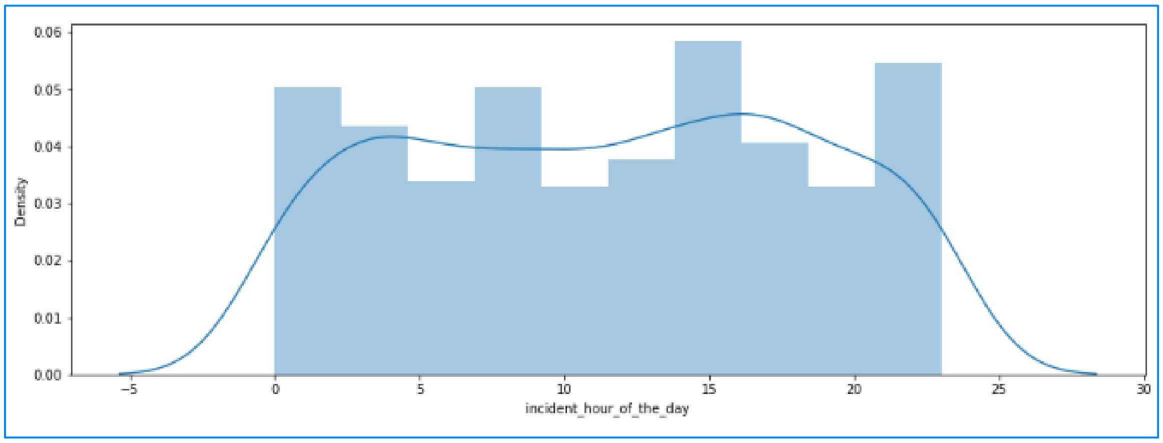
As a part of data cleaning process, categorical features having “?”is replaced with “No Information” as New Category instead of replacing with Mode as this information will be critical and may lead to wrong analysis.



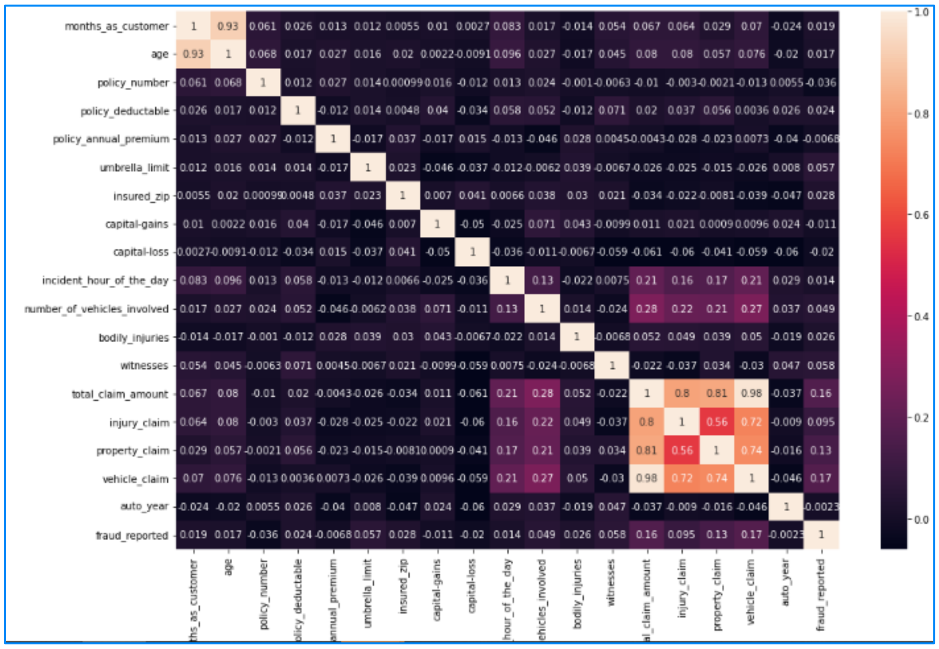
When we do a count plot for fraud cases across different ages, we observed that fraud cases are maximum at the age of 41



Similarly, when we do a count plot for incident_type, collision_type , Incident_severity and witness across fraud cases we observed the following results.



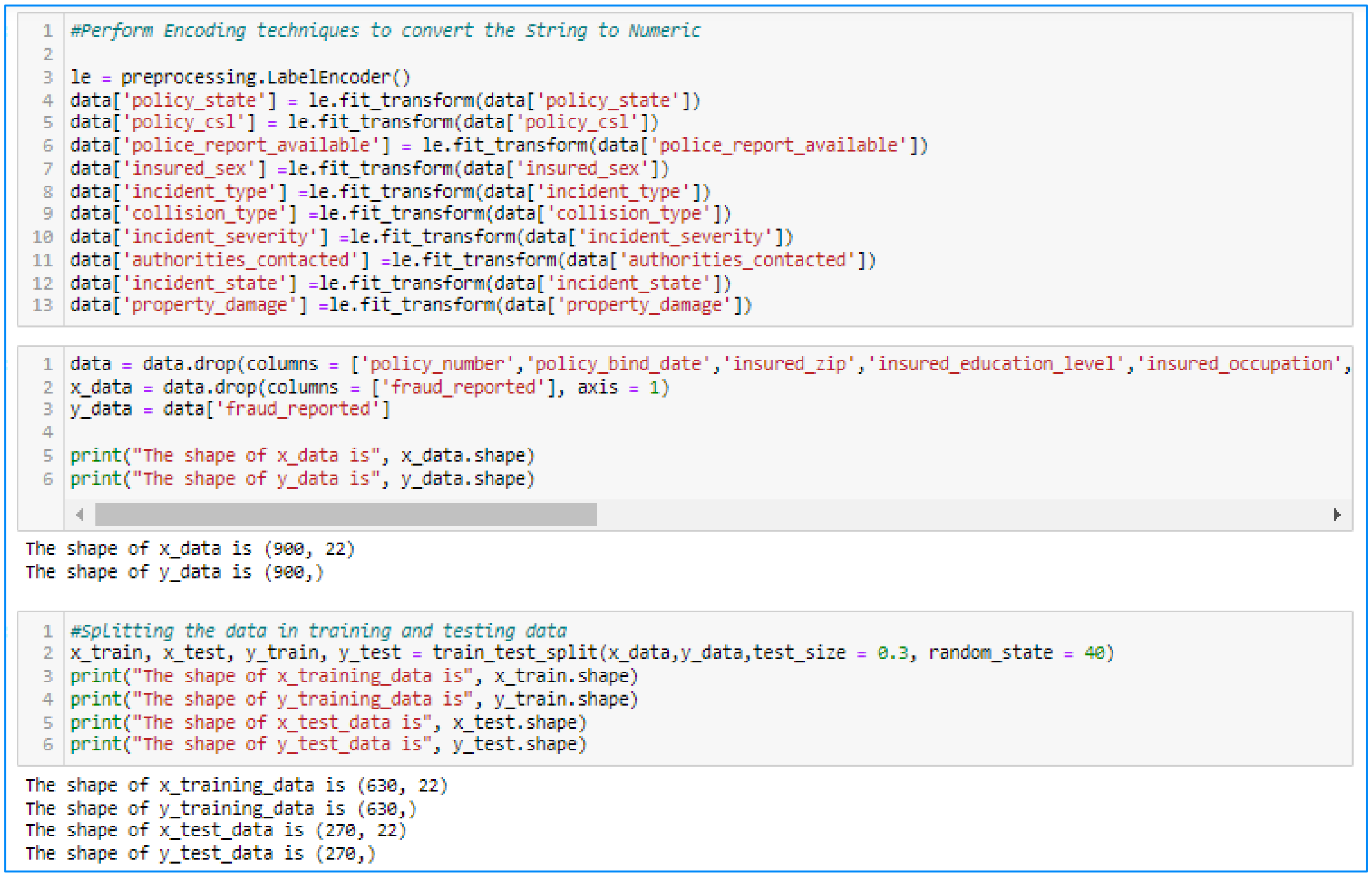
The above plot shows the distribution of incidents happening across the hour of the day.



The above heat map and correlation table represents the bivariate analysis which shows the correlation of one category with the other category. It can be done either by a Heat map or Pair Plot.
The lightest colour represents the highest co-relation and the darkest colour represent the least co-relation. Overall, through EDA we can make the following observations.

- We can see some of the outliers in Months as customers column
- Months as customer and Age features follow near normal distribution with mean as 220 & 39 respectively.
- Customers who doesn't have umbrella limit were more doing frauds.
- Customers who are in the age between 30-40 were claiming fraud auto insurance claims
- As per the data, customers having all level of education were registering fraud claims but Junior degree were bit on higher side.
- Most of the Incidents happened in the night time.
- More fraud claims were registered in South Carolina state and New York states.
- Observed that the claims which doesn't have the policy report were most likely towards the fradulent claims
- Features like Policy Number, Policy bind date, insured zip, insured education level, insured occupation, incident city, incident location, auto make, auto model auto year were not contributing to target variable and hence dropped
- Features like Total claim, property claim, vehicle claim, injury claim were highly correlated independent features hence considering only total claim.

Segregating data into Training & Testing



```
#Perform Encoding techniques to convert the String to Numeric

le = preprocessing.LabelEncoder()
data['policy_state'] = le.fit_transform(data['policy_state'])
data['policy_csl'] = le.fit_transform(data['policy_csl'])
data['police_report_available'] = le.fit_transform(data['police_report_available'])
data['insured_sex'] =le.fit_transform(data['insured_sex'])
data['incident_type'] =le.fit_transform(data['incident_type'])
data['collision_type'] =le.fit_transform(data['collision_type'])
data['incident_severity'] =le.fit_transform(data['incident_severity'])
data['authorities_contacted'] =le.fit_transform(data['authorities_contacted'])
data['incident_state'] =le.fit_transform(data['incident_state'])
data['property_damage'] =le.fit_transform(data['property_damage'])
```

```
data = data.drop(columns = ['policy_number','policy_bind_date','insured_zip','insured_education_level','insured_occupation',
x_data = data.drop(columns = ['fraud_reported'], axis = 1)
y_data = data['fraud_reported']

print("The shape of x_data is", x_data.shape)
print("The shape of y_data is", y_data.shape)
```

```
The shape of x_data is (900, 22)
The shape of y_data is (900,)
```

```
#Splitting the data in training and testing data
x_train, x_test, y_train, y_test = train_test_split(x_data,y_data,test_size = 0.3, random_state = 40)
print("The shape of x_training_data is", x_train.shape)
print("The shape of y_training_data is", y_train.shape)
print("The shape of x_test_data is", x_test.shape)
print("The shape of y_test_data is", y_test.shape)
```

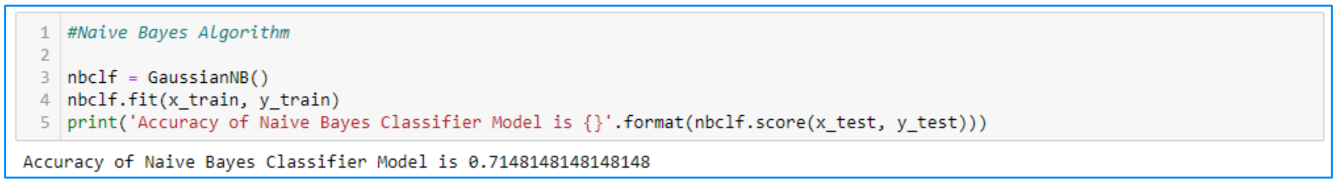
```
The shape of x_training_data is (630, 22)
The shape of y_training_data is (630,)
The shape of x_test_data is (270, 22)
The shape of y_test_data is (270,)
```

We need to segregate the whole dataset into the training **(70%)** and test data **(30%)** so that model will learn the data patterns. Before segregating the data, since we have some of the features in categorical, we need to convert into numeric data before we feed into the Model. This can be achieved using Label Encoder as all these are Nominal Categorical features and the process is called as encoding.

Later, we can drop some of the unwanted features as these features were not contributing much information to detect whether claim is real or fraud.

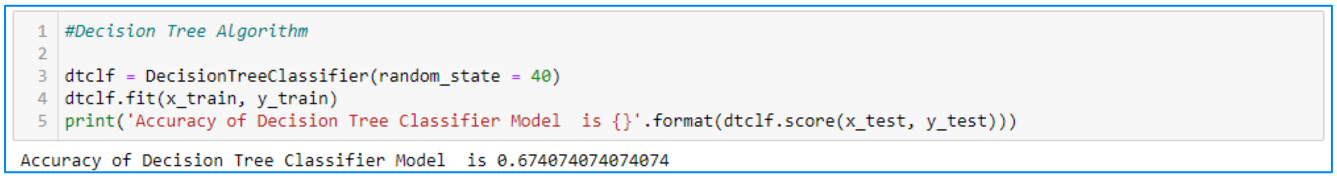
Selecting Approximate Algorithm

Once the data is prepared, we can feed the data to various classification algorithms and train the models. In this project, we build Naïve Bayes classifier, Decision Tree Classifier, Random Forest Classifier, Support Vector Machine (SVM), Adaboost Classifier and XGBoost Classifier models. Since the research paper focused mainly on Naïve Bayes, Decision Tree and Random Forest, we too focused on these models and found that Decision Tree and Random Forest performs well compared to Naïve Bayes Model.



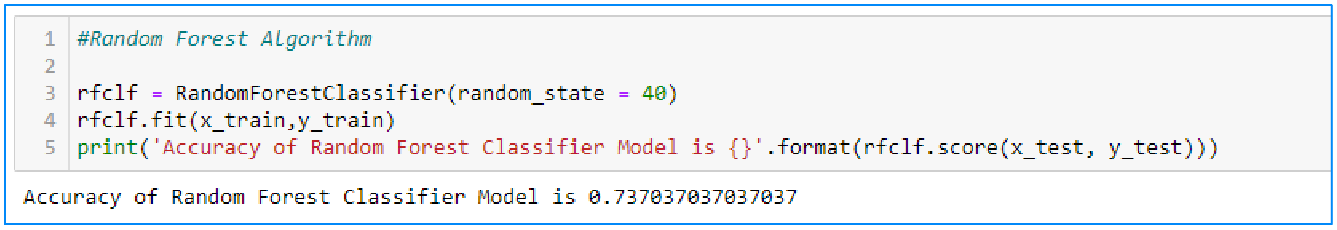
```python
#Naive Bayes Algorithm

nbclf = GaussianNB()
nbclf.fit(x_train, y_train)
print('Accuracy of Naive Bayes Classifier Model is {}'.format(nbclf.score(x_test, y_test)))
```

```
Accuracy of Naive Bayes Classifier Model is 0.7148148148148148
```



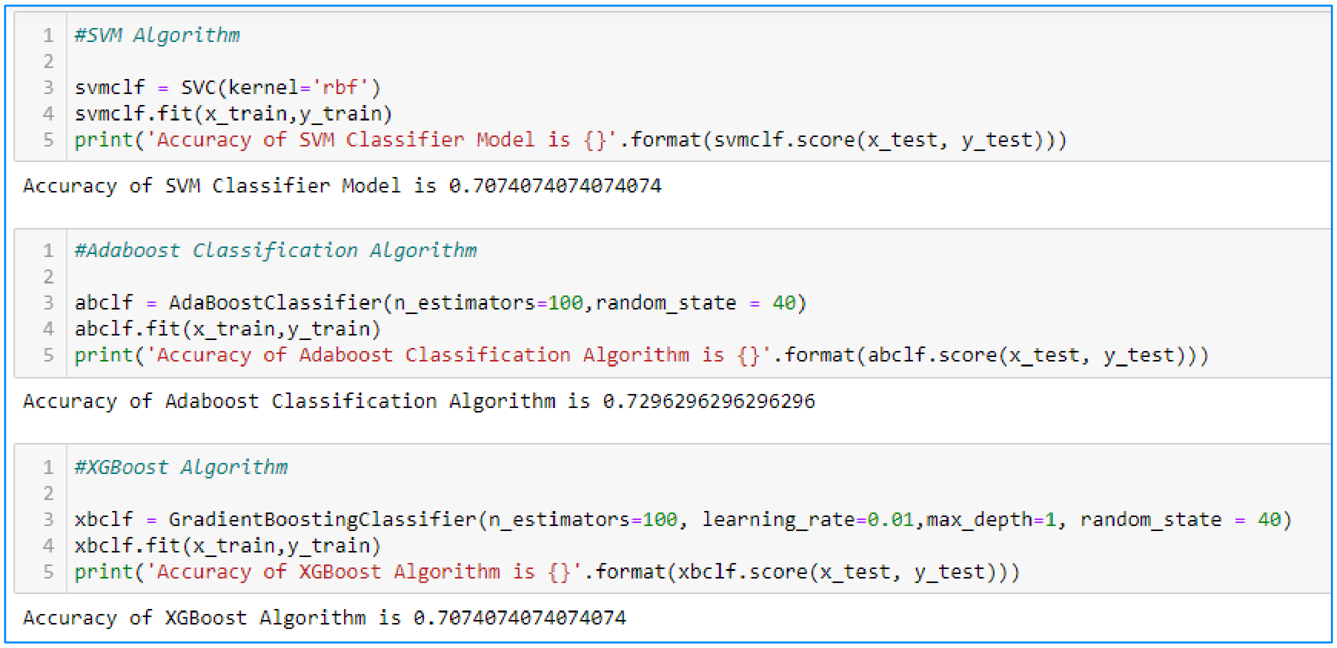
```python
#Decision Tree Algorithm

dtclf = DecisionTreeClassifier(random_state = 40)
dtclf.fit(x_train, y_train)
print('Accuracy of Decision Tree Classifier Model  is {}'.format(dtclf.score(x_test, y_test)))
```

```
Accuracy of Decision Tree Classifier Model  is 0.674074074074074
```



```python
#Random Forest Algorithm

rfclf = RandomForestClassifier(random_state = 40)
rfclf.fit(x_train,y_train)
print('Accuracy of Random Forest Classifier Model is {}'.format(rfclf.score(x_test, y_test)))
```

```
Accuracy of Random Forest Classifier Model is 0.737037037037037
```



```python
#SVM Algorithm

svmclf = SVC(kernel='rbf')
svmclf.fit(x_train,y_train)
print('Accuracy of SVM Classifier Model is {}'.format(svmclf.score(x_test, y_test)))
```

```
Accuracy of SVM Classifier Model is 0.7074074074074074
```

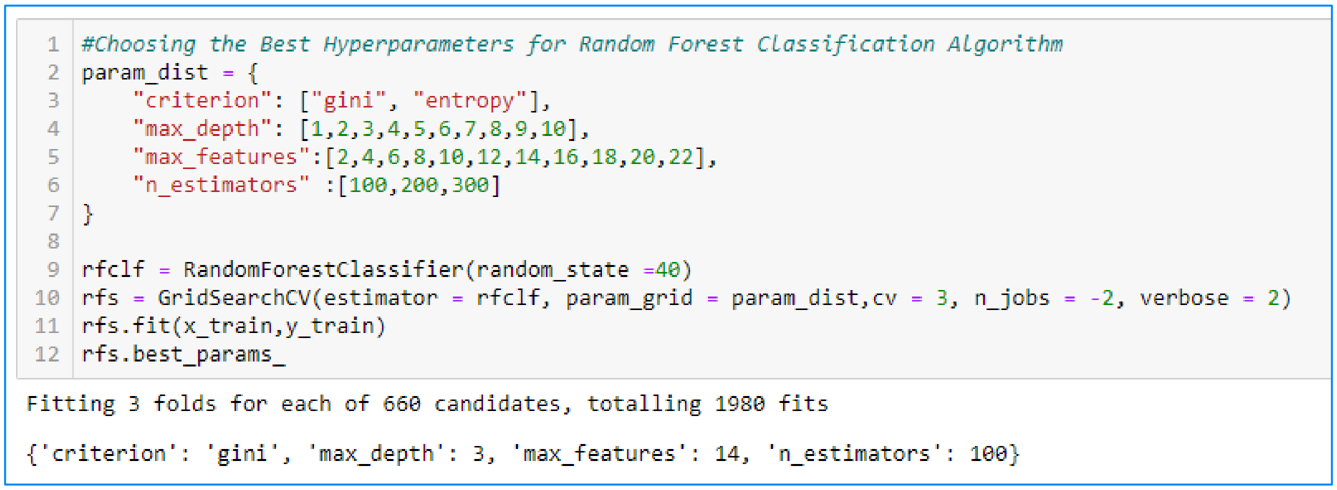
```python
#Adaboost Classification Algorithm

abclf = AdaBoostClassifier(n_estimators=100,random_state = 40)
abclf.fit(x_train,y_train)
print('Accuracy of Adaboost Classification Algorithm is {}'.format(abclf.score(x_test, y_test)))
```

```
Accuracy of Adaboost Classification Algorithm is 0.7296296296296296
```

```python
#XGBoost Algorithm

xbclf = GradientBoostingClassifier(n_estimators=100, learning_rate=0.01,max_depth=1, random_state = 40)
xbclf.fit(x_train,y_train)
print('Accuracy of XGBoost Algorithm is {}'.format(xbclf.score(x_test, y_test)))
```

```
Accuracy of XGBoost Algorithm is 0.7074074074074074
```

We choose **Random Forest Algorithm** as it gives better accuracy among all models. We can find the best hyperparameters by implementing Cross Validation Technique for Random Forest Classifier to improve the accuracy.

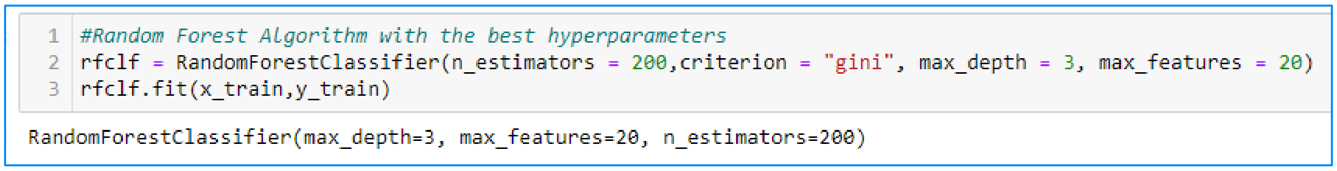


```python
#Choosing the Best Hyperparameters for Random Forest Classification Algorithm
param_dist = {
    "criterion": ["gini", "entropy"],
    "max_depth": [1,2,3,4,5,6,7,8,9,10],
    "max_features":[2,4,6,8,10,12,14,16,18,20,22],
    "n_estimators" :[100,200,300]
}

rfclf = RandomForestClassifier(random_state =40)
rfs = GridSearchCV(estimator = rfclf, param_grid = param_dist,cv = 3, n_jobs = -2, verbose = 2)
rfs.fit(x_train,y_train)
rfs.best_params_
```

```
Fitting 3 folds for each of 660 candidates, totalling 1980 fits

{'criterion': 'gini', 'max_depth': 3, 'max_features': 14, 'n_estimators': 100}
```

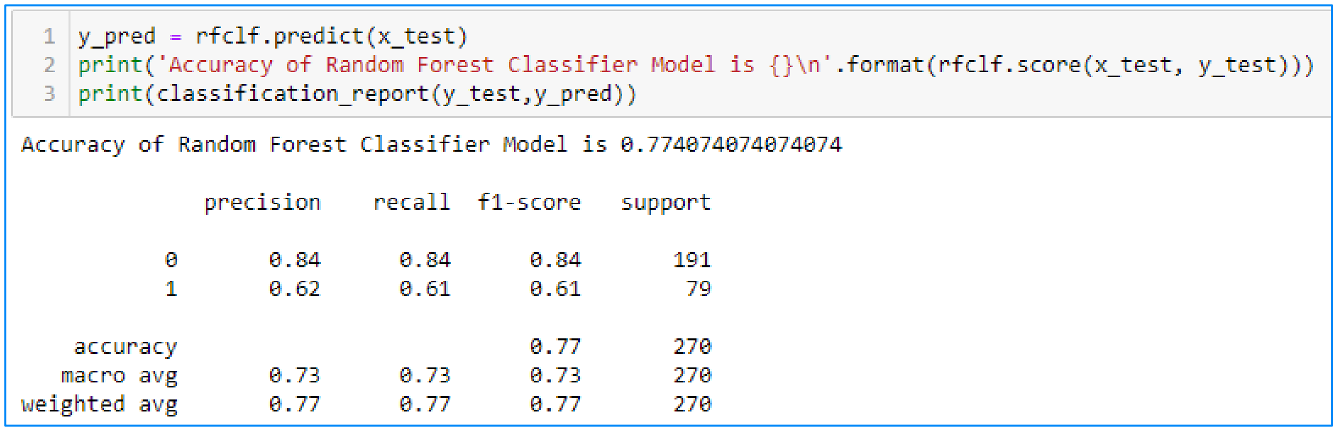
We fill further consider the **Random Forest Model** with **best hyperparameters** for Training & Evaluating the model performance

Executing best training algorithm with testing data



```python
#Random Forest Algorithm with the best hyperparameters
rfclf = RandomForestClassifier(n_estimators = 200,criterion = "gini", max_depth = 3, max_features = 20)
rfclf.fit(x_train,y_train)
```

```
RandomForestClassifier(max_depth=3, max_features=20, n_estimators=200)
```



```python
y_pred = rfclf.predict(x_test)
print('Accuracy of Random Forest Classifier Model is {}\n'.format(rfclf.score(x_test, y_test)))
print(classification_report(y_test,y_pred))
```

```
Accuracy of Random Forest Classifier Model is 0.774074074074074

              precision    recall  f1-score   support

           0       0.84      0.84      0.84       191
           1       0.62      0.61      0.61        79

    accuracy                           0.77       270
   macro avg       0.73      0.73      0.73       270
weighted avg       0.77      0.77      0.77       270
```

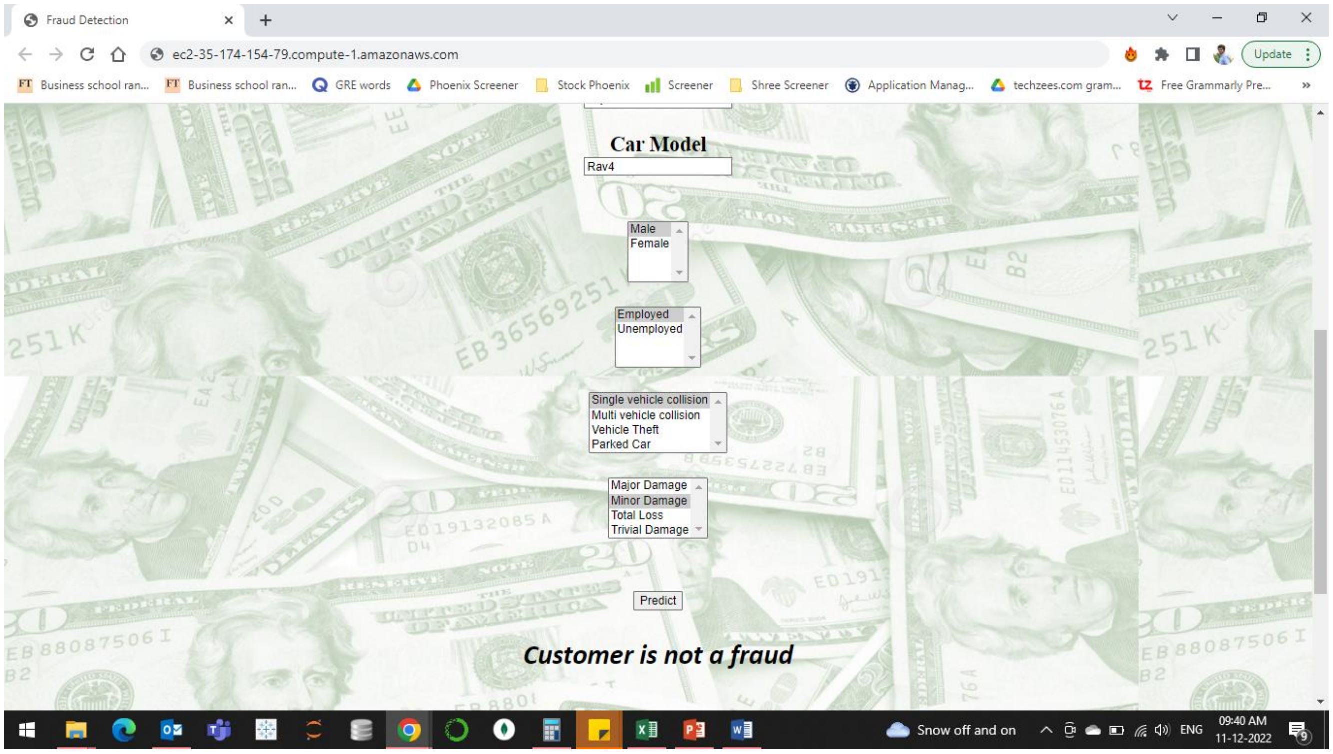
We have measured the following metrics to evaluate the performance of the Random Forest Model. **Accuracy, Precision, Recall, F1-Score.** This trained data is used to predict whether the claim registered is Real or Fraudulent claim. The model is performed well in prediction as well.

We have achieved an accuracy of 77% which is a benefit to the Insurance companies in reducing the fraud loss to great extent. The accuracy can be further improved by capturing additional features and good amount of data representing both legitimate and illegitimate claims for the model to learn. Also, we can use Bagging & Boosting Machine Learning algorithms like Adaboost & XGBoost.

Deployment

Once we get the best model that can be used, we now do prediction using this trained data. So, we are using AWS platform to implement this model.

We firstly write the code using python and then deploy it into AWS using putty. The webpage that would be displayed is shown below. Using this webpage, prediction whether the customer is a fraud or not can be made by entering all the required details to avoid false insurance claims.



Customer is not a fraud

Conclusion

The Model is built successfully to detect the Auto Insurance Fraud claims with around 900 samples. This helps the Insurance companies to save huge amount of Money and Time.

Used Naive Bayes, Decision Tree, Random Forest Classifiers as discussed in Research Paper. Additionally, Implemented SVM, Adaboost and XGBoost Classifiers.

Random Forest Classifier gives most balanced performance with:

- Accuracy Score = 76.4%
- Precision = 77%
- Recall = 77%
- F1 Score = 77%

Having more False Positive rate decrease precision (Impact on customer retention & satisfaction) and having more False Negative rate decrease recall (Impact financial loss) which are risks for Insurance companies. Insurance companies should consider F1 score as their model performance metric as False Positive & False Negative should both are important factors.

Other conclusions that we can draw through our analysis are as follows:

- More Claims registered in NY & SC states.
- Most of the Collisions occurred during Night.
- Collisions that doesn’t involve police reporting were more of fraudulent claims.
- Low level education involved in more claims

Source file

https://www.kaggle.com/datasets/buntyshah/auto-insurance-claims-data

Github Link

https://github.com/group1sdb/Final_Project