Insurance claims — Fraud detection

Submitted by:

ABHILASHA MEWADA

INTRODUCTION

Fraud is one of the largest and most well-known problems that insurers face. This article focuses on claim data of a car insurance company. Fraudulent claims can be highly expensive for each insurer. Therefore, it is important to know which claims are correct and which are not. It is not doable for insurance companies to check all claims personally since this will cost simply too much time and money.

Problem Definition

Insurance fraud is a huge problem in the industry. It's difficult to identify fraud claims. Machine Learning is in a unique position to help the Auto Insurance industry with this problem.

The goal of this project is to build a model that can detect auto insurance fraud. The challenge behind fraud detection in machine learning is that frauds are far less common as compared to legit insurance claims.

Insurance fraud detection is a challenging problem, given the variety of fraud patterns and relatively small ratio of known frauds in typical samples.

Data Analysis

In this project, we have a dataset which has the details of the insurance policy along with the customer details. It also has the details of the accident on the basis of which the claims have been made.

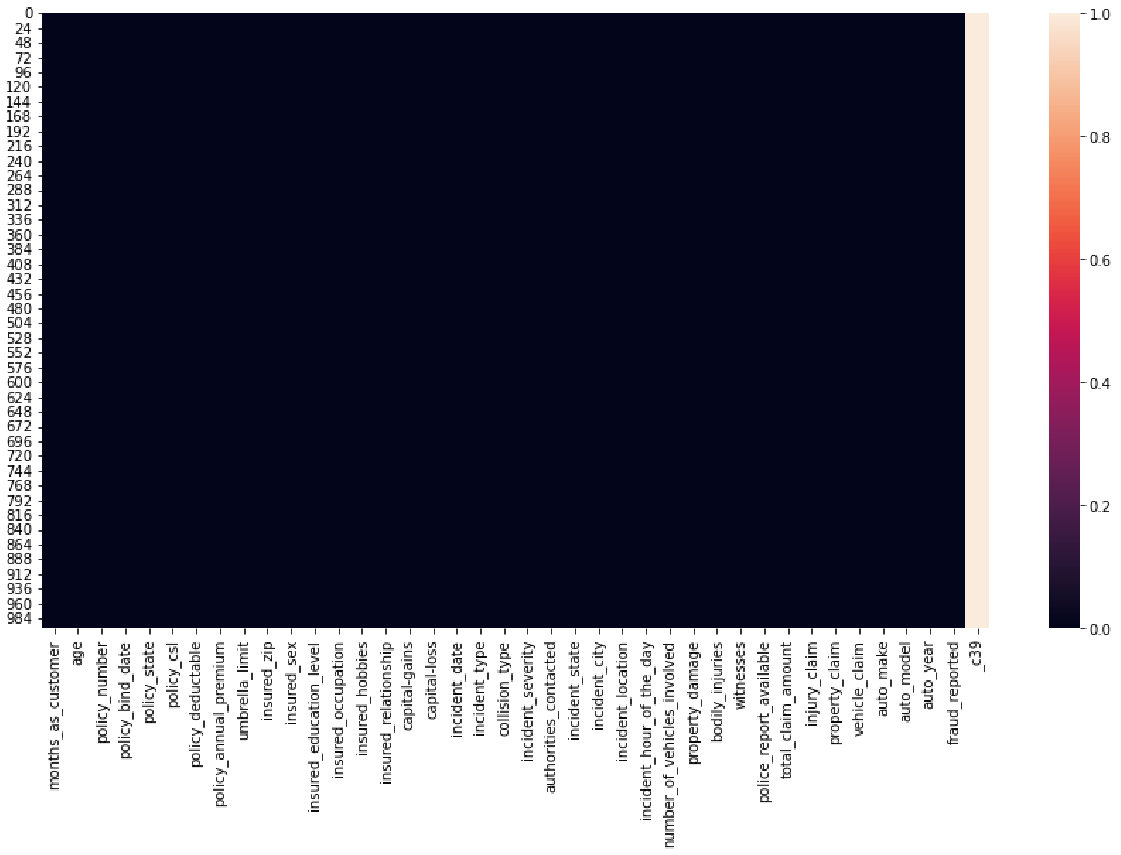
The given dataset contains 1000 rows and 40 columns. The column names like policy number, policy bind date, policy annual premium, incident severity, incident location, auto model, etc.

We started the codes with essentials libraries:

- Hardware and Software Requirements and Tools Used

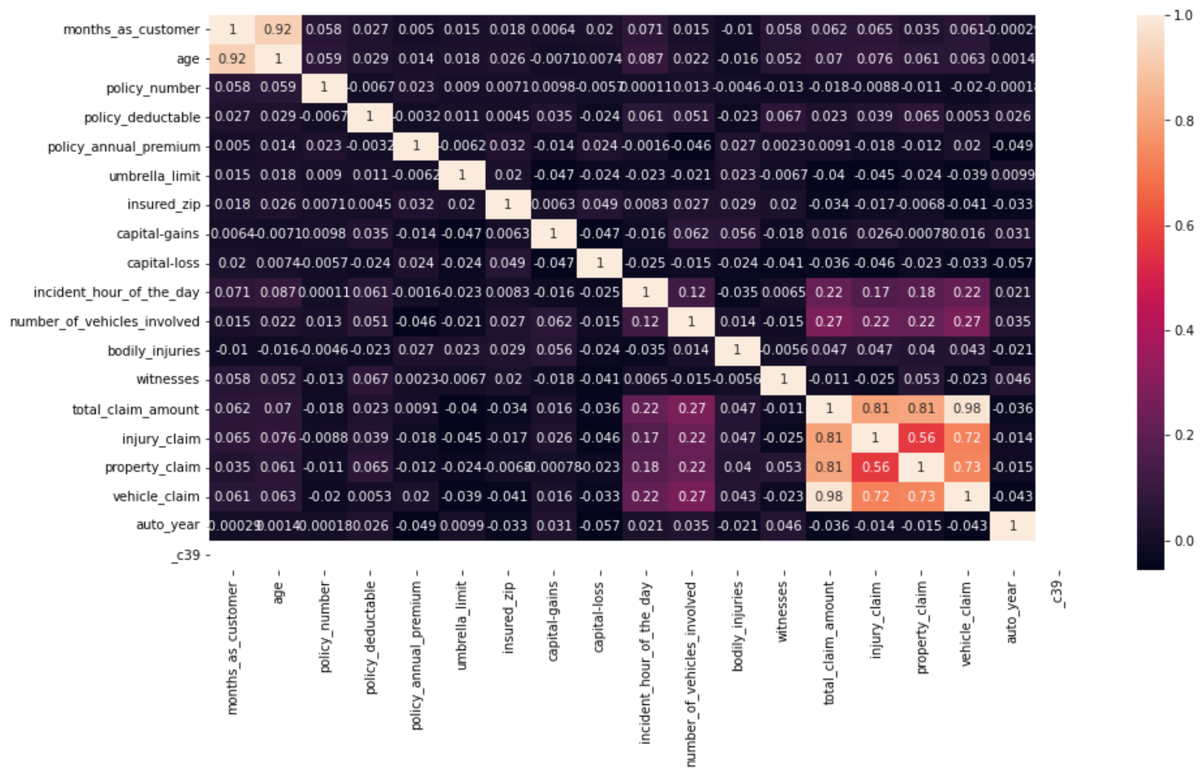
```
#importing essential libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import
train_test_split,GridSearchCV,cross_val_score
from sklearn.preprocessing import
StandardScaler,MinMaxScaler,power_transform
from sklearn.linear_model import
LinearRegression,LogisticRegression
from sklearn.metrics import r2_score,
mean_squared_error,accuracy_score,roc_auc_score,r
oc_curve,confusion_matrix,classification_report
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import
AdaBoostRegressor,GradientBoostingClassifier
import warnings
warnings.filterwarnings('ignore')
from sklearn import preprocessing
import scipy.stats as stats
from scipy.stats import zscore
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import MultinomialNB
import pickle
```

- Data set has no null values and duplicates ,
  We plot heat map for correlation and for checking null values.



Since we notify that no null values are shown in the map.

Now we can see correlation between the column with the help of heatmap and as we can notify that 4 columns are highly correlated.



From the above plot, we can see that there is high correlation between age and months_as_customer.We will drop the "Age" column. Also there is high correlation between total_clam_amount, injury claim, property claim, vehicle claim as total claim is the sum of all others. So we will drop the total claim column.

Encoding using hot encoder

Selecting a variable and test train split:

One-hot encoding is a technique that is used to convert categorical features to a suitable format to be used as an input in Machine Learning algorithms. It transforms a single variable with n observations and d distinct values to d binary variables, where each observation indicating the presence as 1 or absence as 0. In one-hot encoding, the categories are represented as independent concepts.

Univariate analysis:

```
plt.figure(figsize = (25, 20))
plotnumber = 1


for col in df.columns:
    if plotnumber <= 24:
        ax = plt.subplot(5, 5, plotnumber)
        sns.distplot(df[col])
        plt.xlabel(col, fontsize = 15)


    plotnumber += 1
    plt.tight_layout()
    plt.show()
```

Plotting the distplot to show the distribution of data among the dataset.

Outliers:

Outliers are data points that are distant from other similar points. They may be due to variability in the measurement or may indicate experimental errors. If possible, outliers should be excluded from the data set. However, detecting that anomalous instance might be very difficult, and is not always possible.

```
plt.figure(figsize = (20, 15))
plotnumber = 1
for col in df.columns:
    if plotnumber <=
    ax = plt.subplot(5, 5, plotnumber)
        sns.boxplot(df[col])
        plt.xlabel(col, fontsize = 15)
plotnumber += 1
plt.tight_layout()
plt.show()
```

we can see some columns are having outliers,but the %of outliers is less so it can not affect the data.

Transforming the data:

Data is skewed so we can remove it by transform method.

Model building:

For building machine learning models there are several models present inside the Sklearn module.

Sklearn provides two types of models i.e. regression and classification. Our dataset's target variable is to predict whether fraud is reported or not. So for this kind of problem we use classification models.

But before fitting our dataset to its model first we have to separate the predictor variable and the target variable, then we pass this variable to the train_test_split method to create a random test and train subset.

What is train_test_split, it is a function in sklearn model selection for splitting data arrays into two subsets for training data and testing data. With this function, you don't need to divide the dataset manually. By default, sklearn train_test_split will make random partitions for the two subsets. However, you can also specify a random state for the operation. It gives four outputs x_train, x_test, y_train and y_test. The x_train and x_test contains the training and testing predictor variables while y_train and y_test contains the training and testing target variable.

After performing train_test_split we have to choose the models to pass the training variable.We can build as many models as we want to compare the accuracy given by these models and to select the best model among them.
I have selected 8 models:

Logistic Regression from sklearn.linear_model: Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is binary, which means there would be only two possible classes 1 (stands for success/yes) or 0 (stands for failure/no). Mathematically, a logistic regression model predicts P(Y=1) as a function of X.

```
model = LogisticRegression()

model.fit(x_train, y_train)

LogisticRegression()

from sklearn.metrics import r2_score,
mean_squared_error,accuracy_score

import sklearn.metrics as metrics

accuracy=metrics.r2_score(y_test,Y_pred)

print('R square score',accuracy)
```

```
R square score 1.0
```

Now we are going to try classification matrix:

```
from sklearn.linear_model import RidgeClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier



RD_model= RidgeClassifier()
DT_model= DecisionTreeClassifier()
SV_model= SVC()
KNR_model= KNeighborsClassifier()
ADA_model=AdaBoostClassifier()
GB_model= GradientBoostingClassifier()
for m in model:
   m.fit(x_train,y_train)
   m.score(x_train,y_train)
   pred= m.predict(x_test)
   print('Accuracy_Score of ',m, 'is', accuracy_score(y_test,pred)*100)
   print('Confusion Matrix of ',m,' is \n', confusion_matrix(y_test,pred) )
```

```
    print(classification_report(y_test,pred))
    print('*'*50)
```

we find that except **KNeighborsClassifier()** all models are showing 100% accuracy score that proves the best model.

cross validation:

```
from sklearn.model_selection import cross_val_score
for i in model:
   print('Accuracy_Score of ',i, 'is', accuracy_score(y_test,i.predict(x_test))*100)
   print("cross Validation accuracy score of ",i ," is ",cross_val_score(i,x,y,cv=5, scoring='accuracy').mean()*100)
   print('*'*50)
```

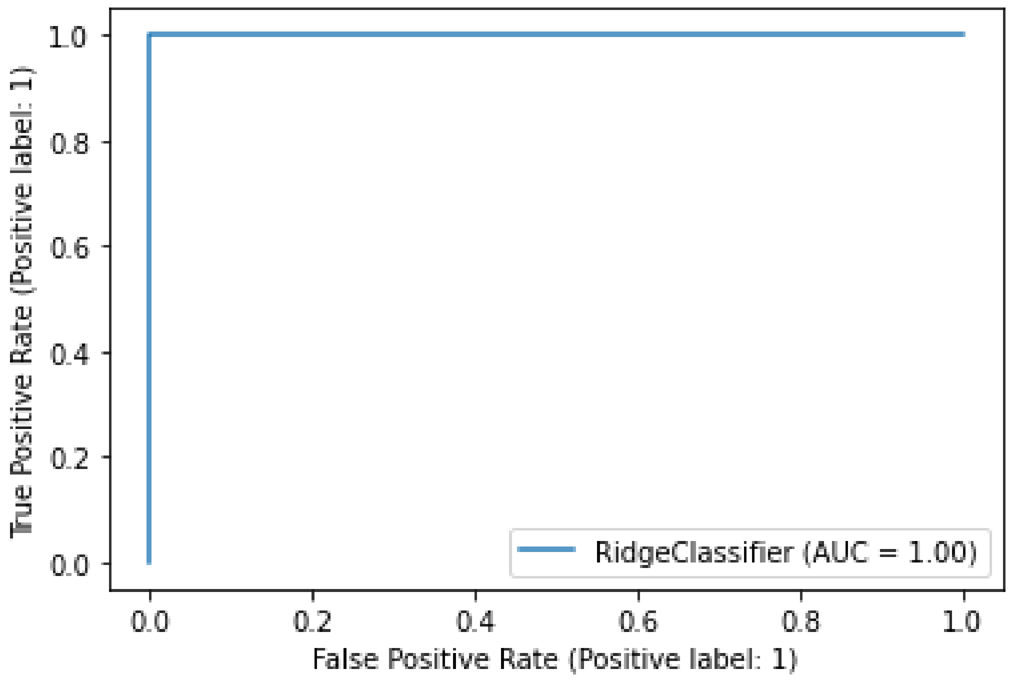
Cross validation of the scores of different models.

hyperparameter using Gridsearch:

Hyper parameter optimisation in machine learning intends to find the hyper parameters of a given machine learning algorithm that deliver the best performance as measured on a validation set. Hyper parameters, in contrast to model parameters, are set by the machine learning engineer before training. The number of trees in a random forest is a hyper parameter while the weights in a neural network are model parameters learned during training. I like to think of hyper parameters as the model settings to be tuned so that the model can optimally solve the machine learning problem.

We will use GridSearchCV for the hyper parameter tuning.

ROC curve:



It is a performance measurement for the classification problems at various threshold settings. ROC is a probability curve and AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting 0s as 0s and 1s as 1s.

Conclusion from models:

We got our best model i.e. RidgeClassifier with the accuracy score of 100%.

Remarks:

This project has built a model that can detect auto insurance fraud. In doing so, the model can reduce losses for insurance companies. The challenge behind fraud detection in machine learning is that frauds are far less common as compared to legit insurance claims.

Saving the model:

Use pickle to save our model so that we can use it later

```
import pickle
# Saving model to disk
pickle.dump(RD_model,open('model.pkl','wb'))
model=pickle.load(open('model.pkl','rb')).
```