INSURANCE CLAIMS –

FRAUD DETECTION USING MACHINE LEARNING

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.

We have a dataset which containing various attributes about the claims, insurer and other circumstansces whoch are included in the data.

Furthermore, we use machine learning to predict which claims are likely to be fraudlent.

Problem Statement:

Business case: 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.

In this project, we are provided 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.

In this example, we will be working with some auto insurance data to demonstrate how we can create a predictive model that predicts if an insurance claim is fraudulent or not.

DATA ANALYSIS:

In this project, we are provided 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 containing 1000 rows and 40 columns . The column name are like policy number, policy bind date, policy annual premium,incident location etc.

- So we firstly checking the nulls of the data by the means of data.isnull().sum()
- Then we will gather the information about the data containg which type of datatype(object or int or float) and how much memory will they used by the help of data.info()

```
      dtype= object )

[7]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 39 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        1000 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
dtypes: float64(1), int64(17), object(21)
memory usage: 304.8+ KB
```

- After checking the nulls we analyse that there is only one column which contain 1000 nulls that is not imp to us i.e- _c39

- By the help of data.describe() we can analyse the column mean, variance and the zeroes if present in the data that need to be changed, if needed.

DATA CLEANING:-

- After checkings the nans we need to clean thembut in this dataset there is only one column which contains the nulls so we drop this and id column which are not imp for this dataset.

- If we are finding the zeroes then it needed to be changed for the better performance of the model wherever it is required do this then only.

EXPLORATORY DATA ANALYSIS:-

- EDA was conducted starting with the dependent variable, fraud_reported. There were approx. 25% of the data were frauds while rest were non – frauds claims.

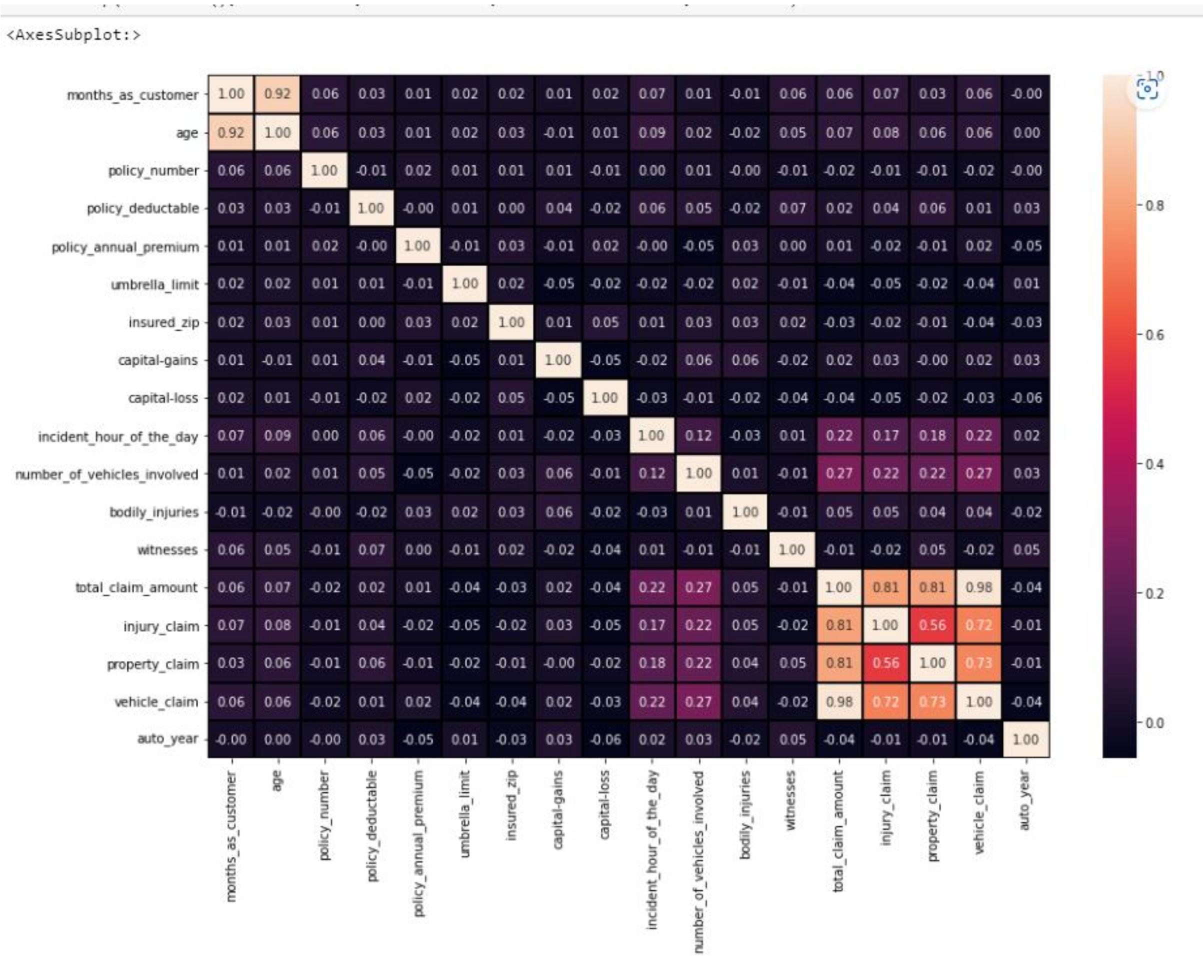
```
[45]: plt.figure(figsize=(10,3))
      ax= sns.countplot(x='fraud_reported',data = data,palette='twilight')
      data['fraud_reported'].value_counts()
```

```
t[45]: N    753
       Y    247
       Name: fraud_reported, dtype: int64
```

There is approx 25% incident are those where Fraud can be reported

```
[52]: plt.figure(figsize=(10,7))
```

- **Plotting heatmap:** This is most important steps in a model building because it gives u the correlation between the independent variableas and also tells the relation between the independent and dependent variables i.e- positively or negatively correlated with the output variable.
  It also gives u the instances about the multicollinearity among the independent variables (present or not) .

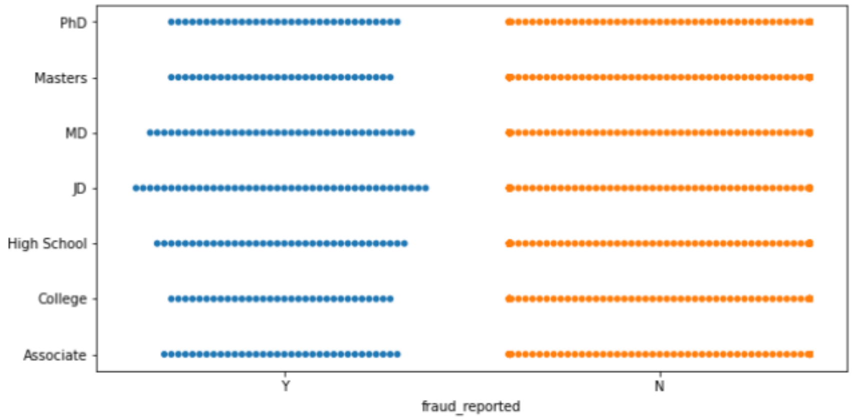


As it shows there is multicollinearity present in some columns as total claim amount and property claim and between months as customer and age etc. so we can check vif values then if its value greter than 5 then we can drop some columns .

Check the highest positively and negatively correlation with the output.

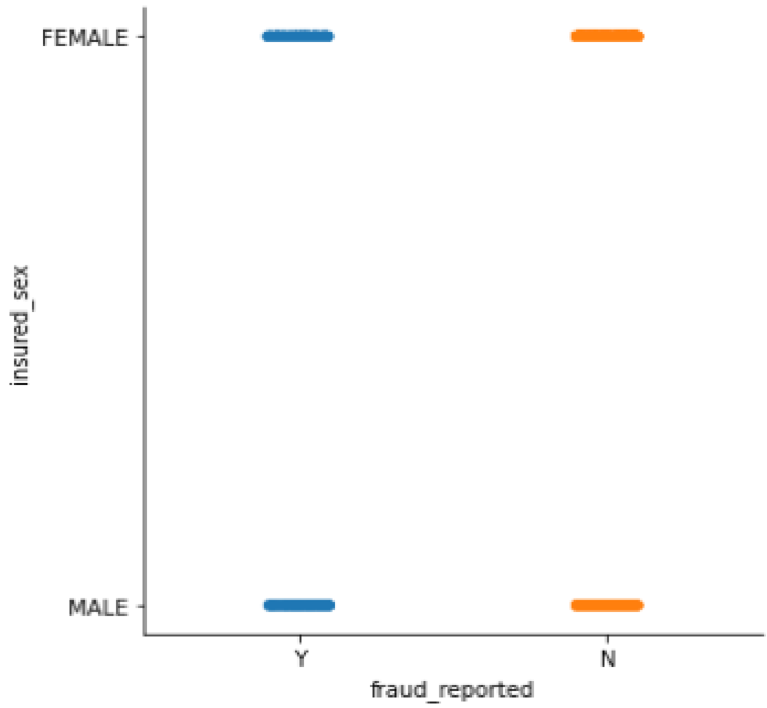
- Visulise the data between the fraud reported and insured sex and the education level

esSubplot:xlabel='fraud_reported', ylabel='insured_education_level'>



It clearly shows every group of education level report Frauds

:Figure size 720x360 with 0 Axes>



It shows the fraud equally made by both males and females

PRE PROCESSING PIPELINE

We all know that preprocessing is very important thing to do in the project .

We almost spent about 70% of the time in the preprocessing of the data.

It consist of various stages that are :-

- **Data cleaning** attempts to impute missing values, removing outliers from the dataset.
- **Data integration** integrates data from a multitude of sources into a single data warehouse.
- **Data transformation** such as normalization, may be applied. For example, normalization may improve the accuracy and efficiency of mining algorithms involving distance measurement.
- **Data reduction** can reduce the data size by dropping out redundant features. Feature selection and feature extraction techniques can be used.

As we treated null values before by droping tht one column which containing all nans in that column

Converting **labels into numeric** and all object datatype which contain object data or categorical data into numeric data by the help of LABEL ENCODER which turns object data type into the numeric data type. So machine can easily readable that data bcz in Machine learning it can not read the object datatype.

By the help of distribution plot we can find that there are some **skewness** find in the dataset so that we removed the skewness by the help of Log transformation/ Power transformation/ sqrt /cbrt etc method

To check the outliers we mostly plot boxplots so we can easily find the outliers . If any present then check the data needs , may or may not it is always possible that all time u have to remove the outliers because some time it contain real values so u cant neglect them.

But for removing the outliers we perform **zscore**

Balancing our imbalanced data

There are different algorithms present to balance the target variable. We use the SMOTE() OR PSA algorithm to make our data balance.

SMOTE algorithm works in 4 simple steps:

1. Choose a minority class as input vector.
2. Find its k-nearest neighbors.
3. Choose one of these neighbors and place a synthetic point anywhere on the line joining the point under consideration and its chosen neighbors.
4. Repeat the step until the data is balanced.

Building machine learning models

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 the model fitting we have to seprate the predictor and target variable, then we pass this variable to the train_test_split method to create the training set and testing set for the model training and prediction.

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 5 models:

1. **LOGISTIC REGRESSION**;- It is one of the simplest ML algorithms that can be used for various classification problems

```
lr.fit(x_train,y_train)
pred_lr=lr.predict(x_test)
print("Accuracy",accuracy_score(y_test,pred_lr)*100)
print('-------------------')
print('===================')
print('-------------------')
print(confusion_matrix(y_test,pred_lr))
print(classification_report(y_test,pred_lr))

Accuracy 78.0
-------------------
===================
-------------------
[[195   0]
 [ 55   0]]
              precision    recall  f1-score   support

           0       0.78      1.00      0.88       195
           1       0.00      0.00      0.00        55

    accuracy                           0.78       250
   macro avg       0.39      0.50      0.44       250
weighted avg       0.61      0.78      0.68       250

In [154]: Test_accuracy = accuracy_score(y_test,pred_lr)

In [155]: from sklearn.model_selection import cross_val_score

In [156]: scr_lr = cross_val_score(lr,x,y,cv=5)
          print("Cross Validation Score of logistic regression model is :- ",scr_lr.mean())
Cross Validation Score of logistic regression model is :-  0.752
```

ACCURACY= 78%

Cross val score = 76%

2. DECISION TREE CLASSIFIER

```
dt = DecisionTreeClassifier()
dt.fit(x_train,y_train)
pred_dt=dt.predict(x_test)
print("Accuracy",accuracy_score(y_test,pred_dt)*100)
print('-------------------------')
print(confusion_matrix(y_test,pred_dt))
print(classification_report(y_test,pred_dt))
```

```
Accuracy 78.8
-------------------------
[[162  33]
 [ 20  35]]
              precision    recall  f1-score   support

           0       0.89      0.83      0.86       195
           1       0.51      0.64      0.57        55

    accuracy                           0.79       250
   macro avg       0.70      0.73      0.71       250
weighted avg       0.81      0.79      0.80       250
```

```
scr_dt = cross_val_score(dt,x,y,cv=5)
print("Cross Validation Score of Decision TREE model is :-",scr_dt.mean())
```

```
Cross Validation Score of Decision TREE model is :- 0.8009999999999999
```

3. RANDOM FOREST CLASSIFIER

```
rf = RandomForestClassifier()
rf.fit(x_train,y_train)
pred_rf=rf.predict(x_test)
print("Accuracy",accuracy_score(y_test,pred_rf)*100)
print('=======================================')
print(confusion_matrix(y_test,pred_rf))
print(classification_report(y_test,pred_rf))
```

```
Accuracy 79.2
=======================================
[[176  19]
 [ 33  22]]
              precision    recall  f1-score   support

           0       0.84      0.90      0.87       195
           1       0.54      0.40      0.46        55

    accuracy                           0.79       250
   macro avg       0.69      0.65      0.66       250
weighted avg       0.77      0.79      0.78       250
```

```
scr_rf = cross_val_score(rf,x,y,cv=5)
print("Cross Validation Score of RANDOM FOREST model is :-",scr_rf.mean()
```

```
Cross Validation Score of RANDOM FOREST model is :- 0.7849999999999999
```

4. SVC

```
sv = SVC()
sv.fit(x_train,y_train)

pred_sv=sv.predict(x_test)
print("Accuracy",accuracy_score(y_test,pred_sv)*100)

print('=========================================')
print(confusion_matrix(y_test,pred_sv))
print(classification_report(y_test,pred_sv))
```

```
Accuracy 78.0
=========================================
[[195   0]
 [ 55   0]]
              precision    recall  f1-score   support

           0       0.78      1.00      0.88       195
           1       0.00      0.00      0.00        55

    accuracy                           0.78       250
   macro avg       0.39      0.50      0.44       250
weighted avg       0.61      0.78      0.68       250
```

```
scr = cross_val_score(sv,x,y,cv=5)
print("Cross Validation Score of SVC model is :-",scr.mean())
```

```
Cross Validation Score of SVC model is :- 0.7529999999999999
```

5. GRADIENT BOOSTING CLASSIFIER

```
gbd = GradientBoostingClassifier()
gbd.fit(x_train,y_train)
pred_gbd=gbd.predict(x_test)
print("Accuracy",accuracy_score(y_test,pred_gbd)*100)
print('=========================================')
print(confusion_matrix(y_test,pred_gbd))
print(classification_report(y_test,pred_gbd))
```

```
Accuracy 83.2
=========================================
[[165  30]
 [ 12  43]]
              precision    recall  f1-score   support

           0       0.93      0.85      0.89       195
           1       0.59      0.78      0.67        55

    accuracy                           0.83       250
   macro avg       0.76      0.81      0.78       250
weighted avg       0.86      0.83      0.84       250
```

```
gbdt = cross_val_score(gbd,x,y,cv=5)
print("Cross Validation Score of GBDT model is :-",gbdt.mean())
```

```
Cross Validation Score of GBDT model is :- 0.825
```

Conclusion from models

We got our best model i.e GRADIENT BOOSTING CLASSIFIER with the accuracy score of 83.2%. Here our model predicts 165 true positive cases out of 195 positive cases and 43 true negative cases out of 65 cases. It predicts 30 false positive cases out of 195 positive cases and 12 false negative cases out of 65 cases. It gives the f1 score of 89%

Hyper parameter tuning

After got the best model now we will tunw the best fit model to get the better accuracy on a validation set

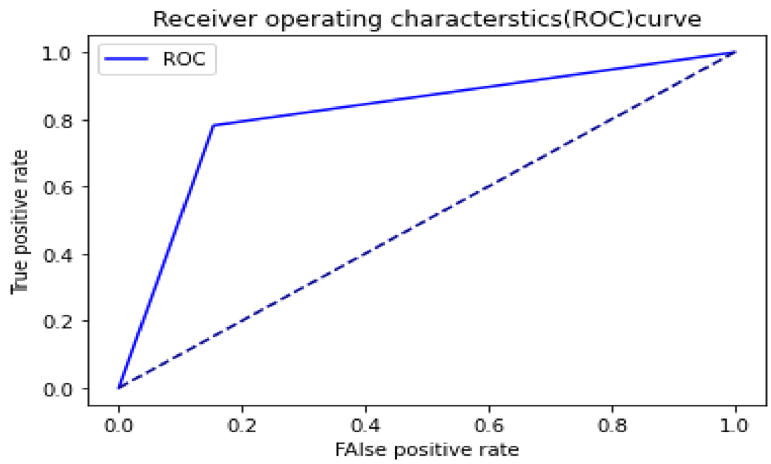
We will use GRIDSEARCHCV approach, the machine learning model is evaluated for a range of hyper parameter values. This approach is called GridSearchCV, because it searches for best set of hyper parameters from a grid of hyper parameters values.

```
from sklearn.model_selection import GridSearchCV

grid_params = {'max_depth':range(4,8),'min_samples_split':range(2,8,2),'learning_rate':np.arange(0.1,0.3)

clf = GridSearchCV(GradientBoostingClassifier(),param_grid=grid_params)
clf.fit(x_train,y_train)

GridSearchCV(estimator=GradientBoostingClassifier(),
             param_grid={'learning_rate': array([0.1]),
                         'max_depth': range(4, 8),
                         'min_samples_split': range(2, 8, 2)})

print(clf.best_params_)

{'learning_rate': 0.1, 'max_depth': 4, 'min_samples_split': 2}

gbd = GradientBoostingClassifier(max_depth = 4, min_samples_split = 4, learning_rate = 0.3)

gbd.fit(x_train,y_train)

GradientBoostingClassifier(learning_rate=0.3, max_depth=4, min_samples_split=4)

y_preds = gbd.predict(x_test)

cfm = confusion_matrix(y_test,y_preds)
cfm

array([[165,  30],
       [ 12,  43]], dtype=int64)

print(classification_report(y_test,y_preds))

              precision    recall  f1-score   support

           0       0.93      0.85      0.89       195
           1       0.59      0.78      0.67        55
```

ROC CURVE

The ROC curve is plotted with TPR against the FPR where TPR is on the y-axis and FPR is on the x-axis.

```
]: plt.plot(fpr,tpr,color = 'blue',label='ROC')
   plt.plot([0,1],[0,1],color = 'darkblue',linestyle='--')
   plt.xlabel('FAlse positive rate')
   plt.ylabel('True positive rate')
   plt.title('Receiver operating characterstics(ROC)curve')
   plt.legend()
   plt.show()
```




```
]: # lets chk area it is covering(AUC)
   auc_score = roc_auc_score(y_test,y_preds)
   print(auc_score)

   0.813986013986014
```

Hence we also got he auc score = 81%

CONCLUDING 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.

SIX different classifiers were used in this project: logistic regression, SVC, Random forest, Decision tree, Knn, Gradient Boosting . Four different ways of handling imbalance classes were tested out with these six classifiers:

We got our best model i.e GRADIENT BOOSTING CLASSIFIER with the accuracy score of 83.2%. Here our model predicts 165 true positive cases out of 195 positive cases and 43 true negative cases out of 65 cases. It predicts 30 false positive cases out of 195 positive cases and 12 false negative cases out of 65 cases. It gives the f1 score of 89%......

AUTHOR

SHIVAM SHARMA