**Insurance Claims – Fraud Detection Using**

**Machine Learning**

**Submited By:**

**Praveena Sampath**

**Problem Definition:**

The goal of Insurance claim – fraud detection project is to build a model that can detect auto insurance fraud. The challenge behind this project using machine learning is that frauds are less common as compared to insurance claim.

Detection of Insurance claim is a challenging problem, the variety of fraud patterns and relatively known frauds in typical samples. While building models, the saving from loss prevention needs to be balanced with the cost of false alert. Machine learning allows for improving predictive accuracy enabling loss control units to achieve higher coverage with low false rates.

**Data Analysis:**

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

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

The positive approach of this data set is the small sample size. However there are still many companies do not have big data sets. The ability to work with available is crucial for any company looing to transition into leveraging data

<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 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

39 \_c39 0 non-null float64

dtypes: float64(2), int64(17), object(21)

memory usage: 312.6+ KB

Compared to a company that waits for the day when it has a huge datasets, Company which started with a small dataset and worked on it. There are some variables that contains null values character ‘?’. The no of null values present is given below.

months\_as\_customer 0

age 0

policy\_number 0

policy\_bind\_date 0

policy\_state 0

policy\_csl 0

policy\_deductable 0

policy\_annual\_premium 0

umbrella\_limit 0

insured\_zip 0

insured\_sex 0

insured\_education\_level 0

insured\_occupation 0

insured\_hobbies 0

insured\_relationship 0

capital-gains 0

capital-loss 0

incident\_date 0

incident\_type 0

collision\_type 0

incident\_severity 0

authorities\_contacted 0

incident\_state 0

incident\_city 0

incident\_location 0

incident\_hour\_of\_the\_day 0

number\_of\_vehicles\_involved 0

property\_damage 0

bodily\_injuries 0

witnesses 0

police\_report\_available 0

total\_claim\_amount 0

injury\_claim 0

property\_claim 0

vehicle\_claim 0

auto\_make 0

auto\_model 0

auto\_year 0

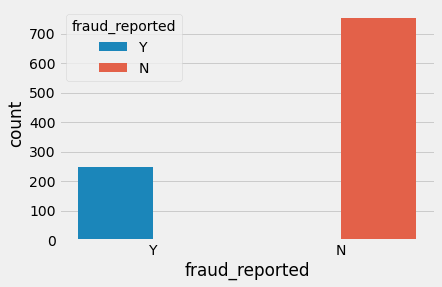
fraud\_reported 0

\_c39 1000

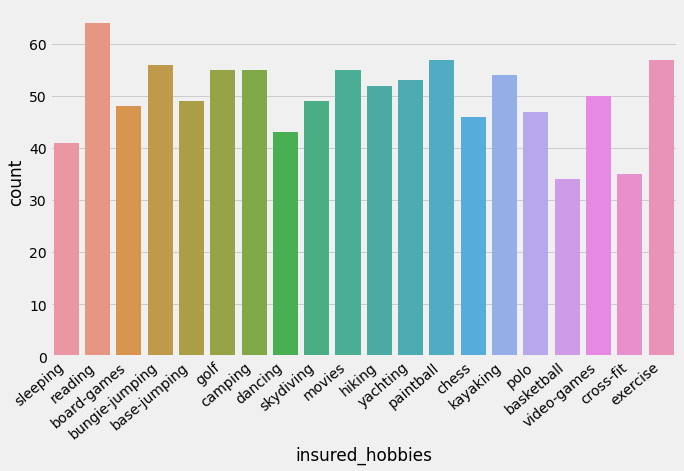
dtype: int64

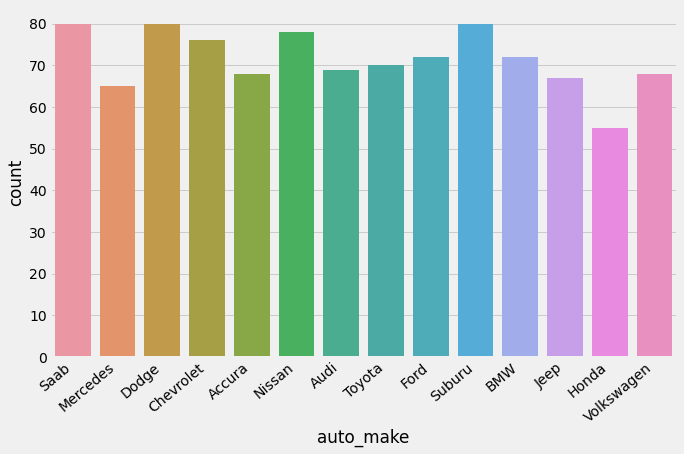
**Exploratory Data Analysis:**

* **Dependent Variable:** Exploratory data analysis was performed from the dependent variable, fraud\_reported. There are 247 frauds and 753 non-frauds. 24.7% of the data were frauds while 75.3% were non-fraudulent claims.

****

* **Correlations among variables:** Heatmap was plotted for variables with atleast 0.3 Pearson correlation coefficient including the DV. Month as customer and age had a correlation of 0.92. Probably because drivers buy auto insurance when they own a car and this time measure only increases with age. Apart from that, there don’t seem to be manu correlations in the data. There was no multicollinearity problems except may be that all the claims are all correlated and somehow total claims have accounted for them. The other claims provide some granularity that will not be captured by total claims. So these variables were kept as it is.
* **Visualizing variables:** The value of fraud reported differs across hobbies of the customer. It seems like chess players and cross fitters have higher chances to fraud.

****

****

Checking correlation between dependent and independent variables.

**Pre-Processing Pipeline:**

Data preprocessing is a predominant step in machine learning to yield highly accurate and insightful results. Greater the quality of data, the greater is the reliability of the produced results. Incomplete, noisy and inconsistent data are the inherent nature of real-world datasets. Data preprocessing helps in increasing the quality of data by filling in missing incomplete data, smoothing noise and resolving inconsistencies.

* **Incomplete data:** can occur due to many reasons. Appropriate data may not be persisted due to a misunderstanding because of instrument defects malfunctions.
* **Noisy Data:** can occur for a no of reasons. The instruments used for the data collection might be faulty. Data entry may contain human errors. Data transmission errors might occur.

There are many stages involved in data preprocessing.

* **Data Cleaning** attempts to impute missing values, removing outliers from the dataset.
* **Data Integration** integrated data from a multitude of sources into a single data warehouse.
* **Data Transforamtion** normalization applied. For eg, normalization ma improve the accuracy and efficiency of mining algorithms involving distance measurement.
* **Data Redcution** can reduce the data size by dropping out redundant features. Features selection and feature extraction techniques can be used.

**Treating null values**

Sometimes there are certain columns which contain the null value used to indicate missing or unknown values or may be the value doesn’t exist. In our dataset the null values are present in columns collision\_type, property\_damage, police\_report\_available and \_c39 with 178,360,343 and 1000 no of null values.

There are different ways of replacing null values from the dataset but we are using fillna to replace the null values from data.

0

policy\_state 0

policy\_csl 0

insured\_sex 0

insured\_education\_level 0

insured\_occupation 0

insured\_hobbies 0

insured\_relationship 0

incident\_type 0

collision\_type 178

incident\_severity 0

authorities\_contacted 0

incident\_state 0

incident\_city 0

property\_damage 360

police\_report\_available 343

auto\_make 0

auto\_model 0

incident\_period\_of\_day 0

collision\_type, property\_damage,police\_report\_available contain many missing values. so, first isolate these variables, inspect these individually for spread of catagory values.

**Converting labels into numeric**

In machine learning, we usually deal with datasets which contain multiple labels in one or more than one column. These labels can be in the form of words or numbers. To make the data understandable or in human readable form, the training data is often labelled in words.

In our data there are columns with categorical values. The columns like incident\_severity, incident\_state, incident\_type, insured\_hobbies, authorities\_contacted, incident\_city police\_report\_available, auto\_make, collision\_type, auto\_model, insured\_occupation, insured\_education\_level, property\_damage, insured\_relationship, policy\_state, insured\_sex, fraud\_reported. These columns have to be treated with label encoder. The target variable fraud\_reported has to convert by using label encoder alone.

**Label Encoder**

Resfers to converting the lables into numeric form so as to convert into machine readable form. Machine learning algorithms can then decide in a better way on how labels must be operated. It is important preprocessing step for the structured dataset in supervised learining.

Label encoding in python can be imported from the sklearn library. Sklearn provides a very efficient tool for encoding. Label encoders encode labels with a value between 0 and n\_classes-1.

**Methods to remove outliers:**

* **Z-score** call scipy.stats.zscore() with the given dataframe as its argument to get numpy array containing the z\_score of each value in a dataframe. Call numpy.abs() with the previous result to convert each element in the dataframe to its absolute value. Use the syntax (array<3). All with array as the previous result to create a boolean array.
* **Interquartile range** The interquartile range can be used to detect the outliers in the dataframe.
* Calculate the interquartile range for the data by using scipy.stats,iqr module.
* Multiply the interquartile range by 1.5
* Add 1.5X interquartile range to the third quartile. Any number greater than this is a suspected outliers.
* Subtract 1.5x interquartile range from the first quartile. Any number lesser than this is suspected outliers.

**Balancing Imbalanced Data**

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

SMOTE algorithm works in 4 simple steps:

1. Choose a minority class as input vector.
2. Find its -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.

The original shape of data was 753 for fraud\_reported with NO value and 247 for YES values. The SMOTE algorithm balances our dta with the highest no of values present in it.

**Building Machine Learning Models:**

For building machine learning models it has several models inside the Sklearn module.

Sklearn provides two types of models i.e. regression and classification. Our dataset target variable is to predict whether fraud is reported or not.

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.

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 these below models

* **Logistic Regression from slearn.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 or 0. Mathematically, a logistic regression model predicts P(y=1) as a function of x.
* **DecisionTree Classifier from sklearn.tree:** Decision trees can be constructed by an algorithmic approach that can split the dataset in different ways based on different conditions. The two main entities of a tree are decision nodes where the data is split and leaves, where we get the outcomes.
* **Random Forest Classifier from sklearn.ensemble:** As we now that a forest is made up of trees and more trees means more robust forest. Similarly, a random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting.IT is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.
* **GaussianNB from sklearn.naive\_bayes:** Naïve Bayes algorithms are a classification technique based on applying Bayes theorem with a strong assumption that all the predictors are independent to each other. The assumption is that the presence of a feature in the same class. It is the simplest Naïve Bayes classifier having the assumption that the data from each label is drawn from a simple Gaussian distribution.

**Conclusion:**

We got our best model i.e. RandomForestClassifier with the accuracy score of 85.39%. This model predicts 196 true positive cases out of 218 positive cases and 190 true negative cases out of 234 cases. It predicts 22 false positive cases out of 218 positive cases and 44 false negative cases out of 234 cases. It gives the f1 score of 85.20%.

* **F1 score:** This is the harmonic mean of precision and recall and gives a better measure of the incorrectly classified cases than the accuracy matrix.
* **Precision:** It is implied as the measure of the correctly identified positive cases from all the predicted positive cases.
* **Recall:** It is the measure of the correctly positive cases from all the actual positive cases. It is important when the cost of false Negative is high.
* **Accuracy:** One of the more obvious metrics, it is the measure of all correctly identified cases. It is most used when all the classes are equally important.

The best and final model was a Random Forest that yelled a F1 score of 0.85 and ROC AUG of 0.95. The model performed excellent. The F1 score and ROC AUG scores were the highest among the other models. In conclusion the model was able to correctly distinguish between fraud claims and legit claims with high accuracy.