Insurance claims — Fraud detection using machine learning

Introduction

Insurance fraud has been around since the beginning of insurance organizations. These are varied and complex crimes that often go unnoticed and cost the insurance industry billions a year. Only in the U.S., the loss on fraudulent insurance claims last year reached $34 billion. Different types of insurance are prone to different crimes, however, in most cases, it manifests deliberate damage to the insured item or the purpose to obtain goods without paying. Detecting insurance fraud can be difficult since not every claim can be investigated thoroughly.

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. While building detection models, the savings from loss prevention needs to be balanced with the cost of false alerts. Machine learning techniques allow for improving predictive accuracy, enabling loss control units to achieve higher coverage with low false positive rates.

Insurance frauds cover the range of improper activities which an individual may commit in order to achieve a favourable outcome from the insurance company. This could range from staging the incident, misrepresenting the situation including the relevant actors and the cause of incident and finally the extent of damage caused.

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 state,policy annual premium, incident severity, incident location, auto model, etc.
The obvious con of this data set is the small sample size. However, there are still many companies who do not have big data sets. The ability to work with what is available is crucial for any company looking to transition into leveraging data science.

```
months_as_customer              int64
age                             int64
policy_number                   int64
policy_bind_date               object
policy_state                   object
policy_csl                     object
policy_deductable               int64
policy_annual_premium         float64
umbrella_limit                  int64
insured_zip                     int64
insured_sex                    object
insured_education_level        object
insured_occupation             object
insured_hobbies                object
insured_relationship           object
capital-gains                   int64
capital-loss                    int64
incident_date                  object
incident_type                  object
collision_type                 object
incident_severity              object
authorities_contacted          object
incident_state                 object
incident_city                  object
incident_location              object
incident_hour_of_the_day        int64
number_of_vehicles_involved     int64
property_damage                object
bodily_injuries                 int64
witnesses                       int64
police_report_available        object
total_claim_amount              int64
injury_claim                    int64
property_claim                  int64
vehicle_claim                   int64
auto_make                      object
auto_model                     object
auto_year                       int64
fraud_reported                 object
_c39                          float64
```

Data Types of the dataset

There is one variables which contain the null values character . The number 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
```

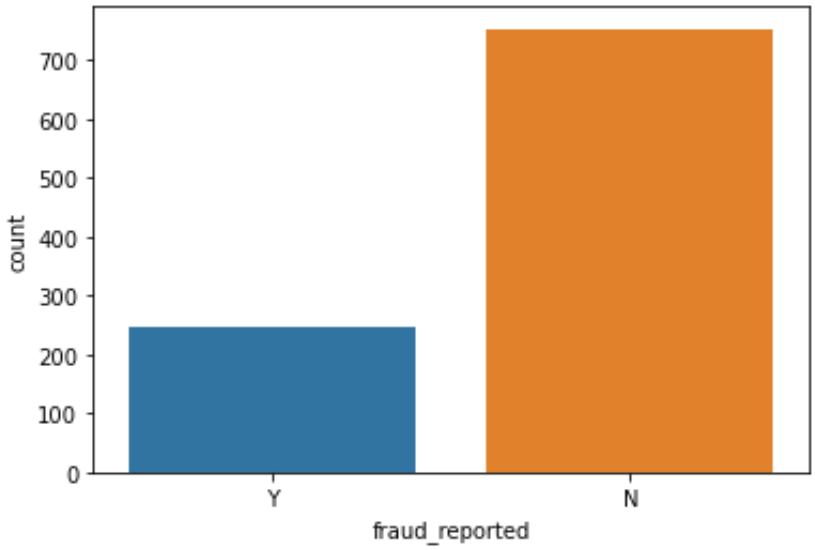
```
insured_sex                      0
insured_education_level          0
insured_occupation               0
insured_hobbies                  0
insured_relationship             0
capital-gains                    0
capital-loss                     0
incident_date                    0
incident_type                    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
```

As_c39 columns has all the null values so we have dropped the column.

Exploratory data analysis

Dependent variable: Exploratory data analysis was conducted starting with the dependent variable, Fraud_reported. There were 247 frauds and 753 non-frauds. 24.7% of the data were frauds while 75.3% were non-fraudulent claims.

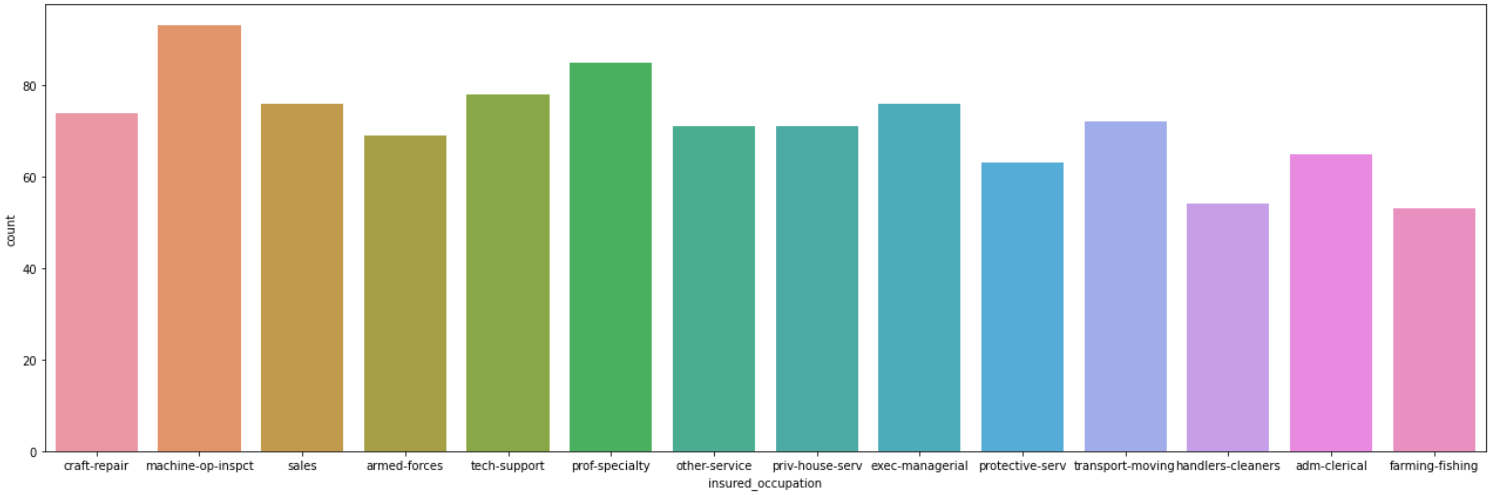
```
N    753
Y    247
Name: fraud_reported, dtype: int64
```



Correlations among variables: Heatmap was plotted for variables with at least 0.3 Pearson's 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 many correlations in the data. There don't seem to be multicollinearity problems except maybe that all the claims are all correlated, and somehow total claims have accounted for them. However, the other claims provide some granularity that will not otherwise be captured by total claims. Thus, these variables were kept.

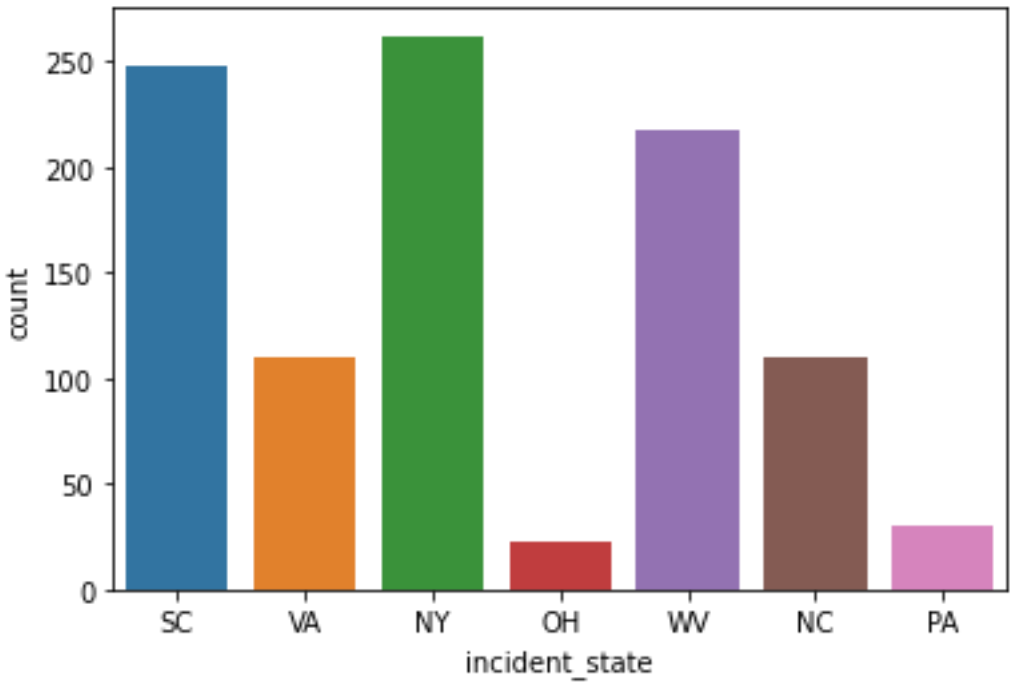
Visualizing variables: The value of fraud reported differs across hobbies of the customer. It see ms like machine-op-inspct and prof-specialty have higher tendencies to fraud.

```
machine-op-inspct     93
prof-specialty        85
tech-support          78
sales                 76
exec-managerial       76
craft-repair          74
transport-moving      72
other-service         71
priv-house-serv       71
armed-forces          69
adm-clerical          65
protective-serv       63
handlers-cleaners     54
farming-fishing       53
```



Most of the incident happened in New York followed by South Carolina

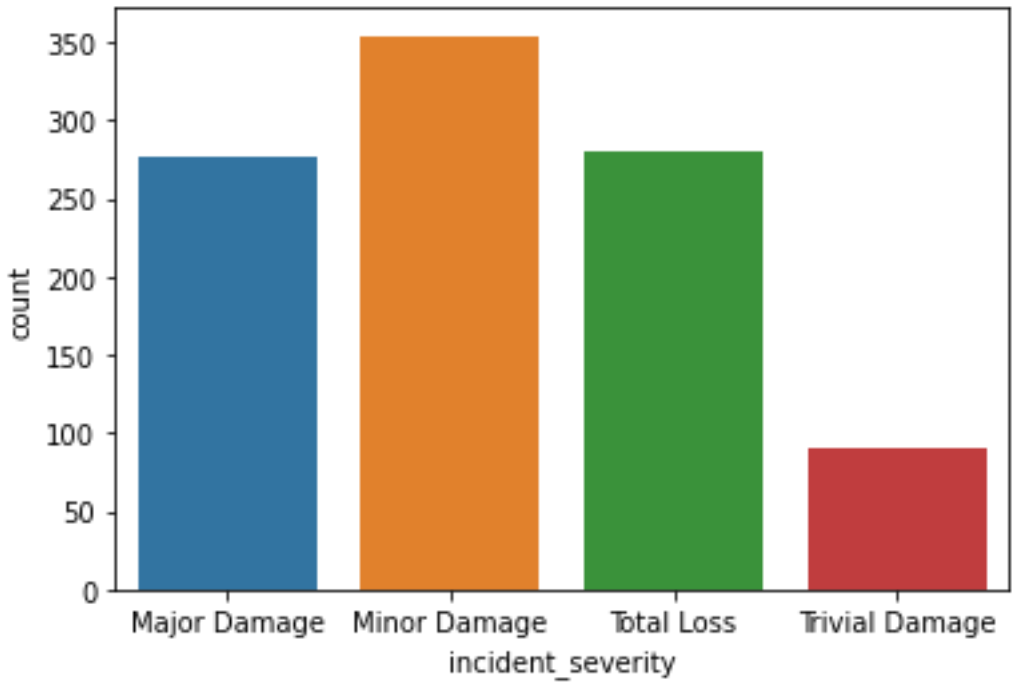
```
NY     262
SC     248
WV     217
VA     110
NC     110
PA      30
OH      23
```



Minor incidents severity seems to have the highest fraud cases that exceeds non fraud cases.

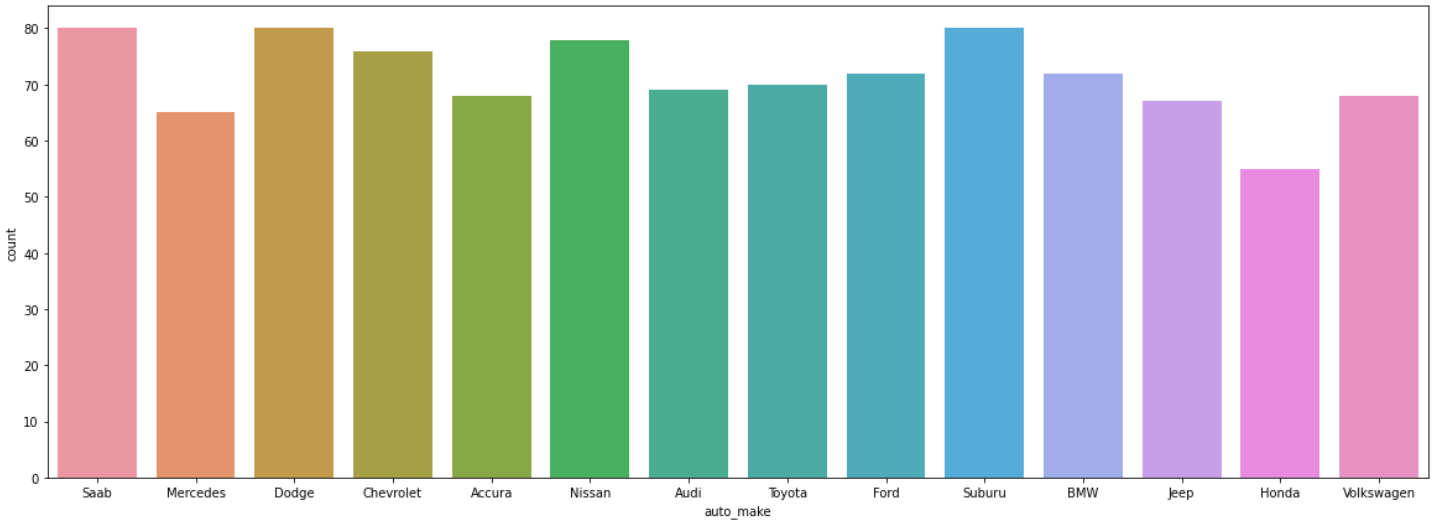
```
Minor Damage       354
Total Loss         280
Major Damage       276
```

```
Trivial Damage     90
```

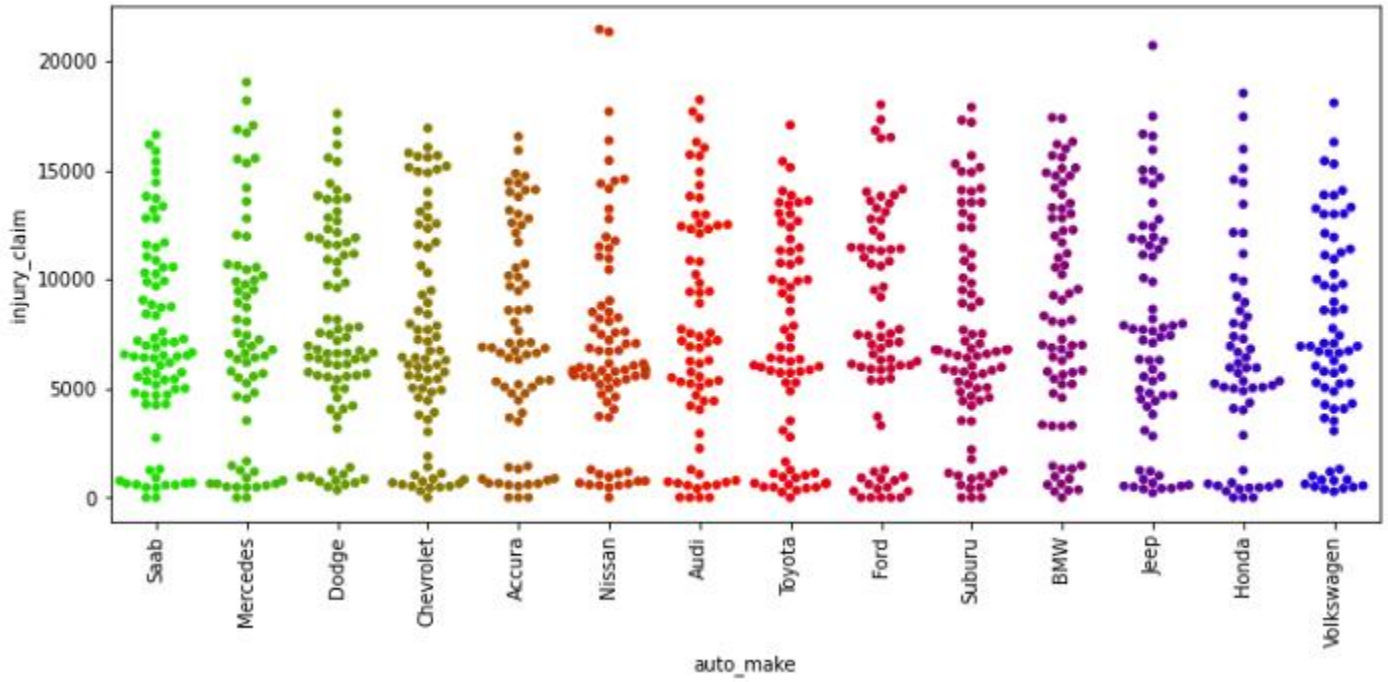


The total_claim_amount is high in Saab,Dodge and Subaru auto_make

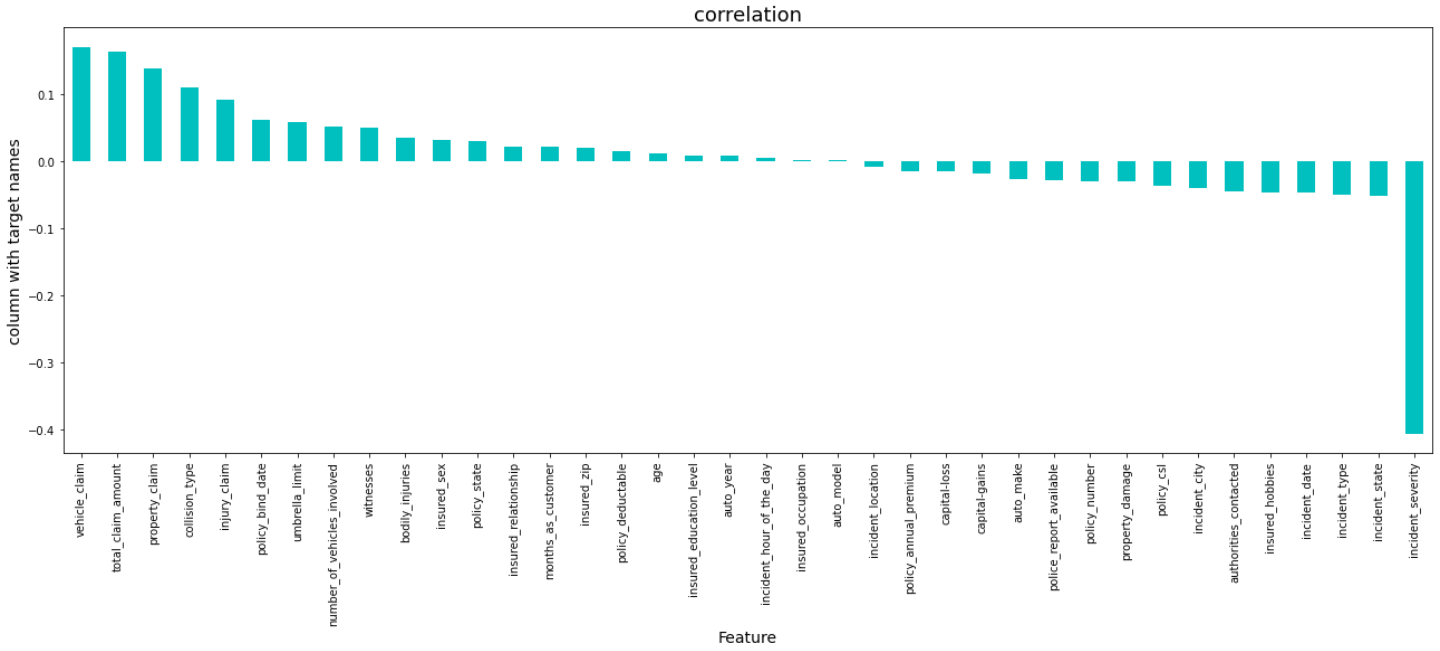
```
Saab           80
Dodge          80
Suburu         80
Nissan         78
Chevrolet      76
Ford           72
BMW            72
Toyota         70
Audi           69
Volkswagen     68
Accura         68
Jeep           67
Mercedes       65
Honda          55
```



The Injury_claim found highest in the Nissan



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, or because of instrument defects and malfunctions**.**

Noisy data can occur for a number of reasons (having incorrect feature values). The instruments used for the data collection might be faulty. Data entry may contain human or instrument errors. Data transmission errors might occur as well.

There are many stages involved in data preprocessing.

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.

Treating null values

Sometimes there are certain columns which contain the null value used to indicate missing or unknown values or maybe the value doesn't exist. n our dataset the null values are present in column_c39

There are different ways of replacing null values from the dataset, but we are dropping the column as this column don't have any 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 one hot encoding or the label encoder. The target variable fraud_reported has to convert by using label encoder only.

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

Outliers handling

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.

Methods to remove outliers:
Z-score — Call scipy.stats.zscore() with the given data-frame as its argument to get a 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 data frame to its absolute value. Use the syntax (array < 3).all(axis=1) with array as the previous result to create a Boolean array.

Balancing our imbalanced data

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

NOTE: SMOTE(Synthetic minority oversampling technique) works by randomly picking a point from the minority class and computing the k-nearest neighbors of this point. The synthetic points are added between the chosen point and its neighbors.

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.

```
from imblearn.over_sampling import SMOTE

smt=SMOTE()

x,y=smt.fit_resample(x,y)
```

The original shape of our data was 753 for fraud_reported with NO value and 247 for YES. The SMOTE algorithm balances our data with the highest number of values present in it.

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 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 5 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. It is one of the simplest ML algorithms that can be used for various classification problems such as spam detection, Diabetes prediction, cancer detection etc.

```
Acuracy_score 0.7907444668008048
              precision    recall  f1-score   support

           0       0.80      0.78      0.79       255
           1       0.78      0.80      0.79       242

    accuracy                           0.79       497
   macro avg       0.79      0.79      0.79       497
weighted avg       0.79      0.79      0.79       497
```

- **DecisionTreeClassifier 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 outcome.

```
Acuracy_score 0.8048289738430584
              precision    recall  f1-score   support

           0       0.84      0.77      0.80       255
           1       0.78      0.84      0.81       242

    accuracy                           0.80       497
   macro avg       0.81      0.81      0.80       497
weighted avg       0.81      0.80      0.80       497
```

- **RandomForestClassifier from sklearn.ensemble:** As we know 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.

```
Acuracy_score 0.8531187122736419
              precision    recall  f1-score   support

           0       0.84      0.88      0.86       255
           1       0.87      0.83      0.85       242

    accuracy                           0.85       497
   macro avg       0.85      0.85      0.85       497
weighted avg       0.85      0.85      0.85       497
```

- **Support Vector Machine Classifier from sklearn.svm:** SVM or Support Vector Machine is a linear model for classification and regression problems. It can solve linear and non-linear problems and work well for many practical problems. Support Vectors are simply the coordinates of individual observation. The SVM classifier is a frontier that best segregates the two classes (hyper-plane/ line).

```
Acuracy_score 0.8531187122736419
              precision    recall  f1-score   support

           0       0.84      0.88      0.86       255
           1       0.87      0.83      0.85       242

    accuracy                           0.85       497
   macro avg       0.85      0.85      0.85       497
weighted avg       0.85      0.85      0.85       497
```

- **KNeighbors Classifier from sklearn.neighbors:** The K in the name of this classifier represents the k nearest neighbors, where k is an integer value specified by the user. Hence as the name suggests, this classifier implements learning based on the k nearest neighbors. By default, the KNeighborsClassifier looks for the 5 nearest neighbors. We must explicitly tell the classifier to use Euclidean distance for determining the proximity between neighboring points.

```
Acuracy_score 0.6519114688128773
              precision    recall  f1-score   support

           0       0.91      0.36      0.51       255
           1       0.59      0.96      0.73       242

    accuracy                           0.65       497
   macro avg       0.75      0.66      0.62       497
weighted avg       0.75      0.65      0.62       497
```

Conclusion from models

We got our best model i.e. RandomForestClassifier with the accuracy score of 85.31%.

Understand what does precision recall and f1 score and accuracy do

- **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.

$$\text{F1-score} = \left(\frac{\text{Recall}^{-1} + \text{Precision}^{-1}}{2}\right)^{-1} = 2 * \frac{(\text{Precision} * \text{Recall})}{(\text{Precision} + \text{Recall})}$$

- **Precision:** It is implied as the measure of the correctly identified positive cases from all the predicted positive cases. Thus, it is useful when the costs of False Positives are high.

$$\text{Precision} = \frac{\text{True Positive}}{(\text{True Positive} + \text{False Positive})}$$

- **Recall:** It is the measure of the correctly identified positive cases from all the actual positive cases. It is important when the cost of False Negatives is high.

$$\text{Recall} = \frac{\text{True Positive}}{(\text{True Positive} + \text{False Negative})}$$

- **Accuracy:** One of the more obvious metrics, it is the measure of all the correctly identified cases. It is most used when all the classes are equally important.

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{(\text{True Positive} + \text{False Positive} + \text{True Negative} + \text{False Negative})}$$

Cross Validation

Cross-validation is **a resampling procedure used to evaluate machine learning models on a limited data sample**. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. We need provide the value of K to divide the group in same number.

In cross Validation we are also getting Random Forest classifier accuracy score is high which is 86.19%

Hyper parameter tuning

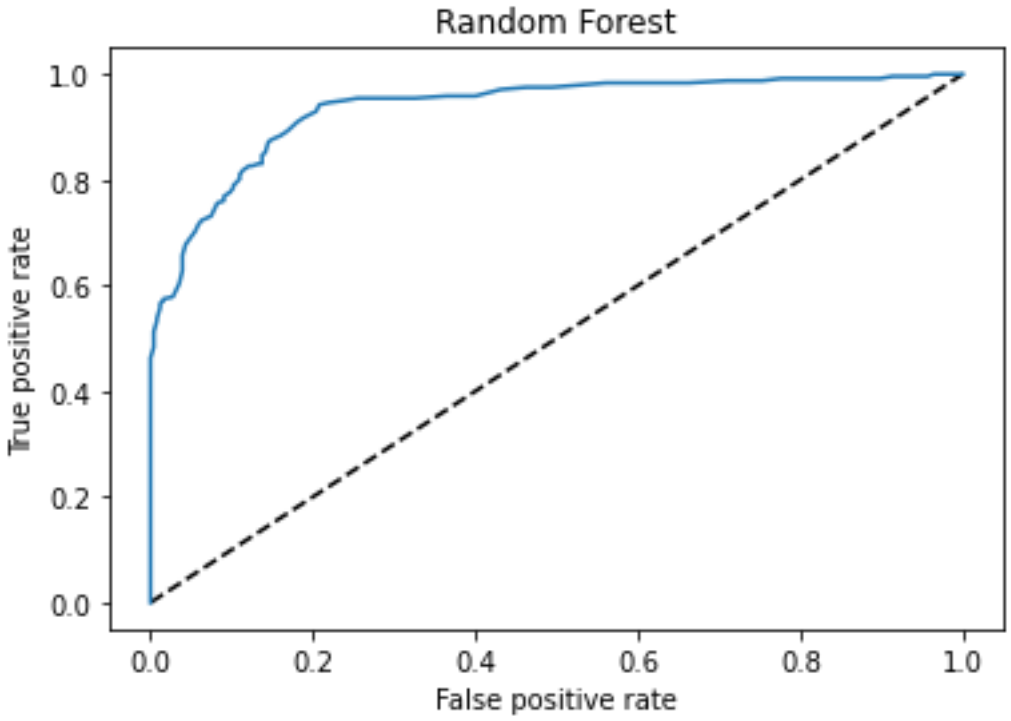
Hyper parameter optimization in machine learning intends to find the hyper parameters of a given machine learning algorithm that deliver the best performance as measured on a validation set. Hyper parameters, in contrast to model parameters, are set by the machine learning engineer before training. The 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.

GridSearchCV

In the 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.

We got Random Forest Classifier with the accuracy score of 83.50% after 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. By analogy, the Higher the AUC, the better the model is at distinguishing between patients with the disease and no disease.
The ROC curve is plotted with TPR against the FPR where TPR is on the y-axis and FPR is on the x-axis.



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.

Five different classifiers were used in this project: logistic regression, K-nearest neighbors, Random forest, Decision tree, Support Vector Machine. Five different ways of handling imbalance classes were tested out with these five classifiers: model with class weighting, oversampling with SMOTE, Cross validation, hyper parameter tuning, and plotting roc curve of the models.

The best and final fitted model was a weighted **Random Forest** that yelled a F1 score of *0.85* and a ROC AUC of *0.95*. In conclusion, the model was able to correctly distinguish between fraud claims and legit claims with high accuracy.

The study is not without limitations. Firstly, this study is restricted by its small sample size. Statistical models are more stable when data sets are larger. It also generalizes better as it takes a bigger proportion of the actual population. Furthermore, the data only capture incident claims of 7 states.