Name:Chanchal Sonawane

Class :BE computer

Div:B

**Mini-Project 2**

**Title:** Credit card fraud detection

**Problem Definition:** Apply the machine learning algorithm to detect credit card fraud based on a dataset.

**Objective:** To implement a machine learning algorithm to detect credit card fraud based on a dataset.

**Software Requirement:**

Python language, Ubuntu 16.04

**Hardware Requirement:**

PIV, 2GB RAM, 500 GB HDD, Lenovo A13-4089Model.

**Learning Objectives:**

Learn How to Apply Support vector machine for classification of credit card fraud detection. According to the SVM you have made from previous training data set.

**2.7 Outcomes:**

After completion of this assignment students are able Implement code for Create Cresit card fraud detection for given dataset .

**Theory :**

* Credit card fraud detection

The goal of this project is to implement a machine learning algorithm to detect credit card fraud based on a dataset that contains credit card transactions made by cardholders. This dataset includes transactions that occurred in the course of two days in September 2013, with 492 fradulent transactions out of a total of 284,315 transactions. The dataset is thus highly unbalanced with the positive class (frauds) accounting for just 0.17% of all transactions. This will be addressed later in the discussion of the best model and sampling trategy. The dataset has 30 input features, 28 of which anonymized, and 1 target variable.

* Data description

The loaded dataset below includes 30 input features with only two of which, Time and Amount, being labeled. This doesn't allow us to do EDA on most of the features.Once we've created the feature variable (X) and the target variable (y), we next view the histograms of each of the features below. We see that the unlabeled feature values have been transformed --- they're the result of a PCA transformation. Amount isn't and neither is Time (between transaction), the latter displaying a bimodal distribution (with two modes: one mode around 50K seconds, or 13.89 hours and the other around 150k seconds, or 41.67 hours).

Next let's look at the distribution of class types. It looks like the daraset we have is drastically biased toward non-fradulent transactions (284,315) in comparison with fradulent transactions (492).

* Data preparation

Next we prepare the features for the machine learning algorithm. The machine learning algorithm requires standard normally distributed data. In the histograms above we saw that the anomymized features were all scaled (via PCA transformation) but not the Time and Amount ones. We then scale these two features.

* Supervised machine learning model - Logistic regression

The first model we'll consider will be a Logistic Regression model. We split the dataset into training and test set and train our model. We then predict the target on the test set and produce a classification report, an accuracy score, and confusion matrix.

Our goal is to have the highest success rate possible in detecting fradulent trasactions. This means is that we want to have 100% success rate for TPs (true positives) and the lowest error rate for FNs (false negatives), the latter so we don't miss any fradulent transactions. This means that the most important score in the confusion matrix above is the recall rate (TP/(TP+FN)). The recall score is quite shabby, but this is not surprising given that the training set is skewed toward non-fradulent transactions.

The ROC curve below shows that our model is quite good for detecting true positives and minimizing false positives, but it doesn't say anything about the false negatives, i.e., those fadulent transactions that fly under the radar. We'll move on to the precision-recall scores to get a better idea of how the model fares with respect to false negatives.

The AUPRC (Area Under the Precision-Recall Curve) shows the trade-off between precision and recall: As recall increases, precision plumets to a point that above 0.5 of recall precision is no better than an unskilled model, depicted by the 0.5 line.

* Resampling and tuning the classifier's parameters

In order to address the skewed sample, I've adopted oparga3's code for under-sampling. The idea behind undersampling in this case is creating a 50/50 ratio for class 1 (fadulent) and class 0 (non-fadulent), but randomly selecting a number of observations from the majority class (class 0 in this case) that equals that of the number of observations from the minority class (class 1).

* Logistic regression on undersampled data

Below the logistic model is run on the undersampled yet balanced in terms of the representation of the two classes. Precision for the detection of fradulent transaction is now 0.93 and recall is 0.94, much better results than with the much larger but highly unbalanced set before. Obviously, we want an even better model, one that catches those extra 6% fradulent transaction.

The model's precision-recall curve is what we're looking for: No huge trade off between precision and recall but rather a similarly high rates for both precision and recall.

## Conclusion

In this way we learn that to how to create Credit card fraud detection system by implement a machine learning algorithm to detect credit card fraud based on a dataset.