Supervised Learning for fraud detection

Supervised learning is the scenario in which the model is trained on the labeled data, and trained model will predict the unseen data. Supervised anomaly detection techniques require a data set that has been labeled as "normal" and "abnormal" and involves training a classifier.

Logistic regression is one of the simplest machine learning models. They are easy to understand, interpretable and can give pretty good results. Infact, it is one of the most popular algorithms for binary classification. Given a set of examples with features, the goal of logistic regression is to output values between 0 and 1, which can be interpreted as the probabilities of each example belonging to a particular class.

The **k-nearest neighbors algorithm**, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label (in the case of classification) or averages the labels (in the case of regression).

Although KNN produces good accuracy on the testing set, the classifier remains slower and costlier in terms of time and memory. It requires large memory for storing the entire training dataset for prediction.

Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, the decision tree algorithm can be used for solving regression and classification problems too. A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label. The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred from training data.

Imbalanced data in classifiers

Imbalanced data is a common problem in machine learning, which brings challenges to feature correlation, class separation and evaluation, and results in poor model performance.

Imbalanced classification refers to a classification predictive modeling problem where the number of examples in the training dataset for each class label is not balanced. That is, where the class distribution is not equal or close to equal, and is instead biased or skewed. Classes that make up a large proportion of the data set are called majority classes. Those that make up a smaller proportion are minority classes.

When we are using an imbalanced dataset, we can oversample the minority class using replacement. This technique is called oversampling. Similarly, we can randomly delete rows from the majority class to match them with the minority class which is called under-sampling.

Why autoencoders

Auto-Encoder (AE) is an unsupervised Artificial Neural Network that attempts to encode the data

by compressing it into the lower dimensions (bottleneck layer or code) and then decoding the data to reconstruct the original input. The bottleneck layer (or code) holds the compressed representation of the input data.
Auto-Encoder models are commonly used for anomaly detection, after training, the reconstructed error of normal data is minimized thus anomaly can be detected if its reconstructed error gets higher than the “normal threshold”.

Another application of AEs is representation learning . Specifically, we'll design a neural network architecture such that we impose a bottleneck in the network which forces a compressed knowledge representation of the original input. This compressed representation of data is useful for learning data features and for finding simpler representations of data for analysis. In other words, it is the simplified model of your input data.
Here by using autoencoder, the hidden representation of our data has been extracted into a Sequential model and the learned representation is used for the prediction of non-fraudulent and fraudulent users using this model. Basically, the hidden patterns that the Autoencoder has learned from the data will be utilized to transform it for training models.

Dataset and Data preparation
The dataset contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where it has 492 frauds out of 284,807 transactions. It is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.
Due to confidentiality issues, it only includes the original features and more background information about the data. Features V1, V2, … V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependent cost-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.
Based on basic EDAs that we performed, resulted insights are:

- The data set is highly imbalanced.
- The time does not seem to be a crucial feature in distinguishing normal vs fraud cases. Hence, it has been dropped.
- The numerical amount in fraud and normal cases differ highly, hence we did a scaling.

Metrics
Accuracy is first important metric to measure the performance of classification algorithms. However, in case of imbalanced datasets, this metric does not provide a correct performance evaluation of the specific approach. To measure the performance correctly, researchers measure the performance based on True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN), precision, recall, and F1 scores. We also propose measuring it using the ROC curve. Usually, precision alone is not enough to evaluate a model. This is even more important when faced with imbalanced classes. You will rarely evaluate a precision score alone. There is no

absolute threshold for a "good" precision: it will all depend on the problem to which it is applied, and on other performance metrics (like recall).

Results

Table. Comparing metrics for different algorithms

Metrics:		Precision	Recall	F1-score
Logistic Reg.	with rep. Le.	1.00	0.77	0.87
	without	0.85	0.45	0.59
KNN	with rep. Le.	0.98	0.88	0.93
	without	0.87	0.75	0.81
Decision Tree	with rep. Le.	0.87	0.90	0.88
	without	0.67	0.68	0.67

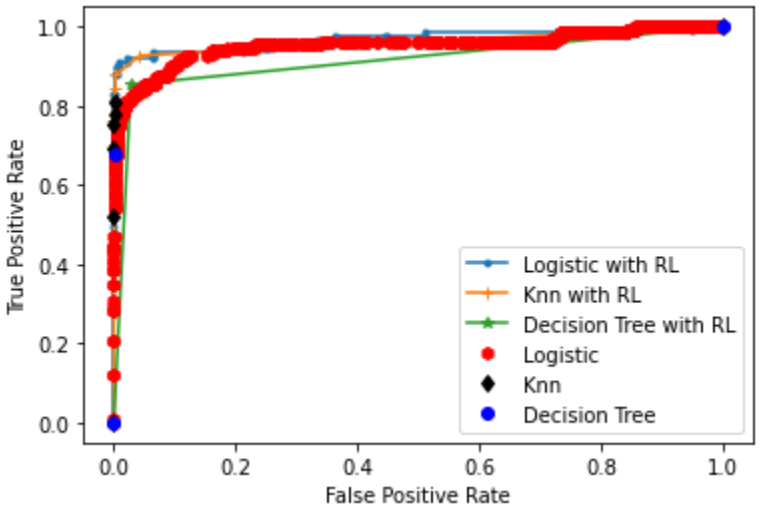

Figure. ROC curve

Is supervised method always good?!

Supervised learning algorithms are trained on labeled data, and the model learns to predict the labels for the unseen data. Labeled data can be expensive to gather. So, the other option is detecting anomalous fraud points through using unsupervised algorithms. They do not require labeled data for training the model. These algorithms identify patterns in the data and try to group the data points based on observed similarities in patterns.