National Taiwan University

Financial Technology

Project 1: Credit Card Fraud Detection

Lecturer:
Che Lin (林澤)

Author: Tadeo Hepperle

October 4, 2022

Contents

1 Introduction

In this project we use a dataset from Kaggle "Credit Card Fraud Detection" to predict fraudulent spendings on credit cards, given a range of features. For this task, we will compare classical machine learning models like logistic regression, support vector machines and random forests with deep neural networks and evaluate their performance.

1.1 Description of the data

The dataset was originally from Kaggle, named "Credit Card Fraud Detection" and contains 31 variables, grouped into 30 features and one class label. Two of these features are the time the transaction took place and the amount spent. The 28 remaining features were obtained by performing a principal component analysis on a larger dataset. The classlabel is either 0 (normal) or 1 (fraudulent) for each record. The entire data consists of 99999 independent transaction records. Of those only 223 are fraudulent, which amounts to only 0.22% and makes the data extremely unbalanced.

1.2 Description of methods

Besides deep neural networks, we also take a brief look at logistic regression, support vector machines and random forest models.

1.2.1 Logistic Regression

In Logistic regression we try to predict a binary response by using an array of numeric features. The model formula looks like this:

$$P(y=1) = \frac{exp(\beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k)}{1 + exp(\beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k)}$$

where $P(y = 1)$ represents the probability of class 1 (fraud in our case) and the $x_1, \ldots, x_k$ represent k predictors. The probability of the other class ($P(y = 0)$) is therefore $1 - P(y = 1)$. Because the derivative of the logistic function is never 0 there is no closed form solution and methods like gradient descent need to be used for optimization. However if the data is linearly separable, these methods will never converge and will result in an "infinitely steep" probability curve.

1.2.2 Support Vector Machines

Support Vector Machines (SVM) aim to find a hyperplane that linearly seperates the data linearly into 2 homogeneous halfspaces. Because this is not always possible, two tricks can be applied. First, it is useful to use a soft margin instead of a hard one which allows some points of the wrong class to lie in the half space of the other class as long as they are still pretty close to the hyperplane. In some cases, like noisy data this can even give better classification results in cases where a hard margin could perfectly seperate the two classes. The second "Trick" that can be used to make Support Vector machines more useful is to use a certain kernel function to transform the input features to a different space domain. This can make even originally not linearly seperable data, seperable if the right transformations are used. In this project, we will not experiment with any kernel functions.

1.2.3 Random Forest Classifier

Random Forest is an ensemble method. That means it combines the power of an array of weak learners that on their own would perform poorly. Typically decision trees are used as the weak classifier, hence the name random "Forest". There are multiple ideas, how to let the individual trees learn and how to combine them. in Bootstrapping for example, we let each tree learn on a bootstrapping sample of the original data and later take a majority vote among all trees as the final decision. The bootstrapping sample is produced by randomly sampling objects from the data n times with put backs (where n is the number of objects in total). The sample is different for every tree and can therefore allow the trees to learn different aspects of the data. The problem with this approach is, that the bootstrapping samples are still highly correlated. Random Forest aims to solve this, by further diversifying and restricting what data can be used by each tree during learning. The idea is simple: At each split to be made in a decision tree, not all features are considered, but only a subset of features is allowed to be used to make the split. Typically for k features, $\sqrt{k}$ randomly selected features are allowed in each split. This increases the variance among weak learners and can lead to classification improvements.

2 Results

With an 80-20 train-test split of the data, 3 models were trained: Logistic Regression, Support Vector Machine and Random Forest Classifier.

2.1 Confusion Matrices and Classification Metrics

As Figure 1 shows, all models are pretty good at classifying the non-fraud cases as non-fraud, which is non-surprising since these comprise the overwhelming majority of the training data. Taking a look at the classification metrics in Table 1 we can see that the SVM is overall performing best, also with highest accuracy. To judge the performance, I would use the F1-score as it provides a good middle ground between precision and recall, in both of which high values are desired. I think the F1 metric is most suitable on this dataset because it is not so much affected by the very imbalanced distribution of fraud vs. non-fraud cases in the data.

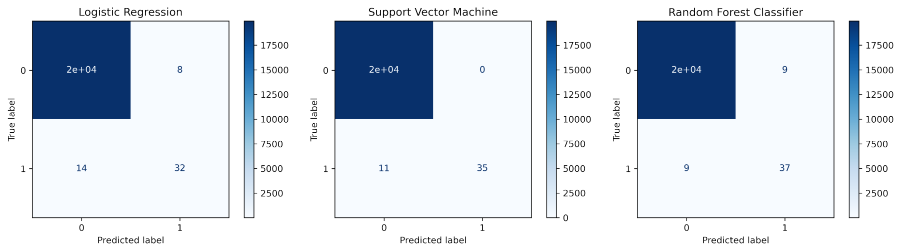

Figure 1: Confusion matrices for logistic regression, SVM and Random Forest on test data

Table 1: Accuracy, Precision, Recall and F1-Score for the 3 models

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.9989	0.8000	0.6957	0.7442
Support Vector Machine	0.9994	1.000	0.7609	0.8642
Random Forest Classifier	0.9991	0.8043	0.8043	0.8043

2.2 Constructing a Deep Neural Network

Secondly we construct a deep neural network and compare it to the methods previously presented. The network has an input layer with 30 nodes, one for each feature. Then there are d fully connected hidden layers, each consisting of k neurons. A ReLU layer is put in between each two neighboring hidden layers. In the end, there is one output node, to which a sigmoid function is applied, to convert scores to probabilities. If the output is greater or equal to 0.5 we classify a case as fraud, otherwise as non-fraud. The model is trained using the ADAM Optimizer, reducing the binary cross entropy loss function. The training uses mini-batch gradient descent. There are some parameters that we need to set in advance:

1. The number of hidden layers d.
2. The number of nodes in each hidden layer k.
3. The learning rate LR.
4. The $BATCHSIZE$, of batches used in the gradient descent.
5. number of $EPOCHS$ (how many times to go over the entire dataset before stopping training)

To perform a full grid search over all of these parameters would be very costly and take too much time, so we will try to figure out good paramters individually.

2.2.1 Learning Rate

For the learning rate the values $LR \in \{0.001, 0.0001, 0.00003, 0.00001, 0.000003\}$are considered while holding other values constant ($d = 4, k = 8, BATCHSIZE =$

$64, EPOCHS = 50$). The following graphs show the results for the resulting confusion matrices on the test data after 50 epochs of training:

2.3 Summary

Bibliography