Classification-Clustering analysis on Credit Approval and Fraud Detection System Implementation

Konstantinos Lampropoulos ***Konstantinos Mavromatis***

TEAM-P

I. Introduction

The goal of the project is to analyze data offered by a bank, concerning the credit approval of its customers. For a lot of bank customers we will analyze their characteristics and we will focus on making a model that predicts whether they are eligible for credit approval or not. We will also make an effort to detect frauds: For example, when a customer gets a credit approval while he/she should not base on their credentials.

II. Data Exploration

A. Data Format

Data concerns with credit card applications and is collected from http://archive.ics.uci.edu/ml/datasets/Credit+Approval. There are 690 observations (37 of which have missing values) with 15 features (among which 6 are real valued and the rest are categorical) and 1 class attributes. For confidential reasons, all attribute names and (categorical) values have been transformed to meaningless symbols.

	CustomerID	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	Class
0	15776156	1	22.08	11.46	2	4	4	1.585	0	0	0	1	2	100	1213	0
1	15739548	0	22.67	7.00	2	8	4	0.165	0	0	0	0	2	160	1	0
2	15662854	0	29.58	1.75	1	4	4	1.250	0	0	0	1	2	280	1	0
3	15687688	0	21.67	11.50	1	5	3	0.000	1	1	11	1	2	0	1	1
4	15715750	1	20.17	8.17	2	6	4	1.960	1	1	14	0	2	60	159	1

B. Data Imputation

Several methods have been applied for data imputation:

1. Deletion: delete any observations with missing values.
2. Simple imputation: impute missing value of numerical variable with its mean and missing value of categorical variable with any of its categories.

Conclusion: we didn't observe any significant differences between them when building models to get estimated error.

Data Visualization

It would be extremely useful and could bring in more insights about modeling if data could be visualized before applying any further operations. Fig. 1 shows the scatter plot of numerical variables from the data set where no particular patterns could be observed between any pairs which suggests no apparent correlations and hence we need further investigations on the data set.

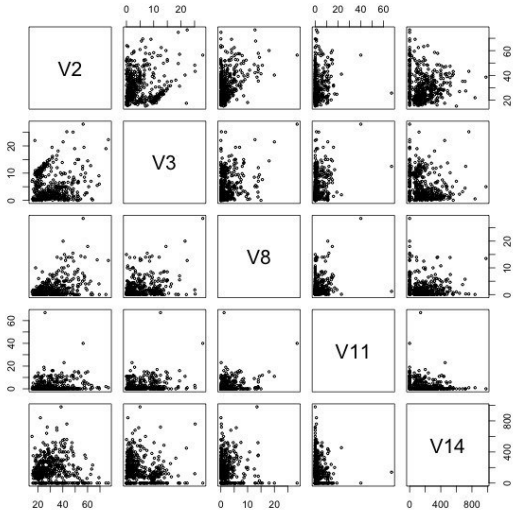


Fig. 1. Scatter plot of numerical variables

Moreover, principal component analysis (PCA) and Kernel-PCA are applied to detect the main directions of data variance. For 2 principal components the cumulative explained variance is around 50%**.** Despite that fact we will continue with 2 components in the sake of the visualization of the results.

III. Features and Models

A. Features Extraction and Selection

Linear dependency and multi-collinearity are detected among original features via variance inflation factors (VIF). For linear models we delete highly correlated variables and combine several related variables into one compounded variable. We have tried to add top principle components to expand the feature space but it turns out to have no improvement on the model accuracy. In support vector machines, features are further expanded via linear kernel, polynomial kernel, radial kernel and sigmoid kernel.

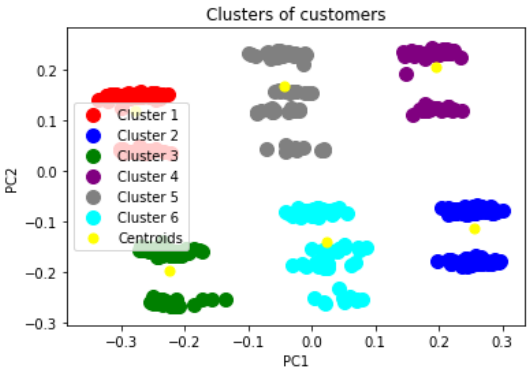
B. Clustering

In order to have a better sense of our data we did some clustering with two different methods:

i) K-means

ii) Hierarchical clustering (ward method)

We obtained in both methods 6 clusters. We tried to continue to model our dataset based on these clusters but we did not get better results than our principal components.



C. Modelling

A good variety of models are tested in this paper and for detailed results please refer to Table I. For modelling our data, we will use the 2 principal components. We divide our data into train(80%) and test(20%) sets. We use (10) k-cross validation (in our train set) and the confusion matrix as a metric (in our test set) in order to test the accuracy of each set.

(i) Logistic Regression and PCA selection

We fit logistic regression on our 2-PCA and 2-kernelPCA components separately. The kernel-PCA provided us a better accuracy on our test set so we will continue to other modelling methods with these components.

(ii) K-NN Algorithm

In k-*NN classification*, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors. We got the best result for k=7.

(iii) Tree Models (Decision-Random Forest)

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.

Parameter Tuning for our Decision Trees

In machine learning, hyperparameter optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm.

The traditional way of performing hyperparameter optimization has been grid search, or a parameter sweep, which is simply an exhaustive searching through a manually specified subset of the hyperparameter space of a learning algorithm. A grid search algorithm must be guided by some performance metric, typically measured by cross-validation on the training set or evaluation on a held-out validation set.

A **random forest** is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.

(iv) Support Vector Machine Classification

An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

A margin is a separation of line to the closest class points. A good margin is one where this separation is larger for both the classes.

(v) Extreme Gradient Boosting-XGBoost Classification

Gradient boosting builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function. Gradient boosting is typically used with decision trees of a fixed size as base learners. Gradient boosting is an approach where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models.

(vi) Neural Networks

Deep learning methods usually work well on data with hidden complexities and here we tried neural networks with only two hidden layers. ANNs are composed of multiple nodes. The neurons are connected by links and they interact with each other. The nodes can take input data and perform simple operations on the data. The result of these operations is passed to other neurons.Each link is associated with weight. ANNs are capable of learning, which takes place by altering weight values. In the following models we will use the original data (divided into train-test) and not the principal components. Our first ANN will have (optimizer=adam):
,one input layer (8 nodes , act.function=relu),two hidden layers (8 nodes , act.function=relu),one output layer (1 node , act.function=sigmoid).Our secondANN had the same network architecture but it was also used the **dropout** method in order to reduce overfitting which actually worked since we got a higher prediction in the test set.

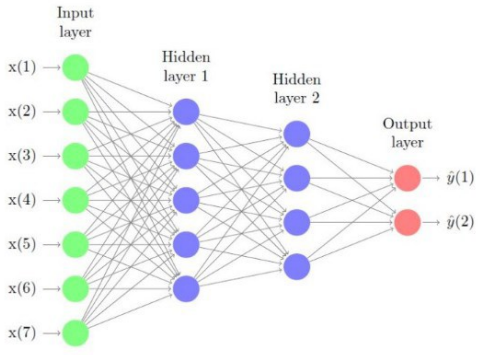


Figure 1: An illustrative example of a feed-forward neural network with two hidden layers, seven features and two output states. Deep learning network classifiers typically have many more layers, use a large number of features and several output states or classes. The goal of learning is to find the weight on every edge that minimizes the out-of-sample error measure.

IV. RESULTS

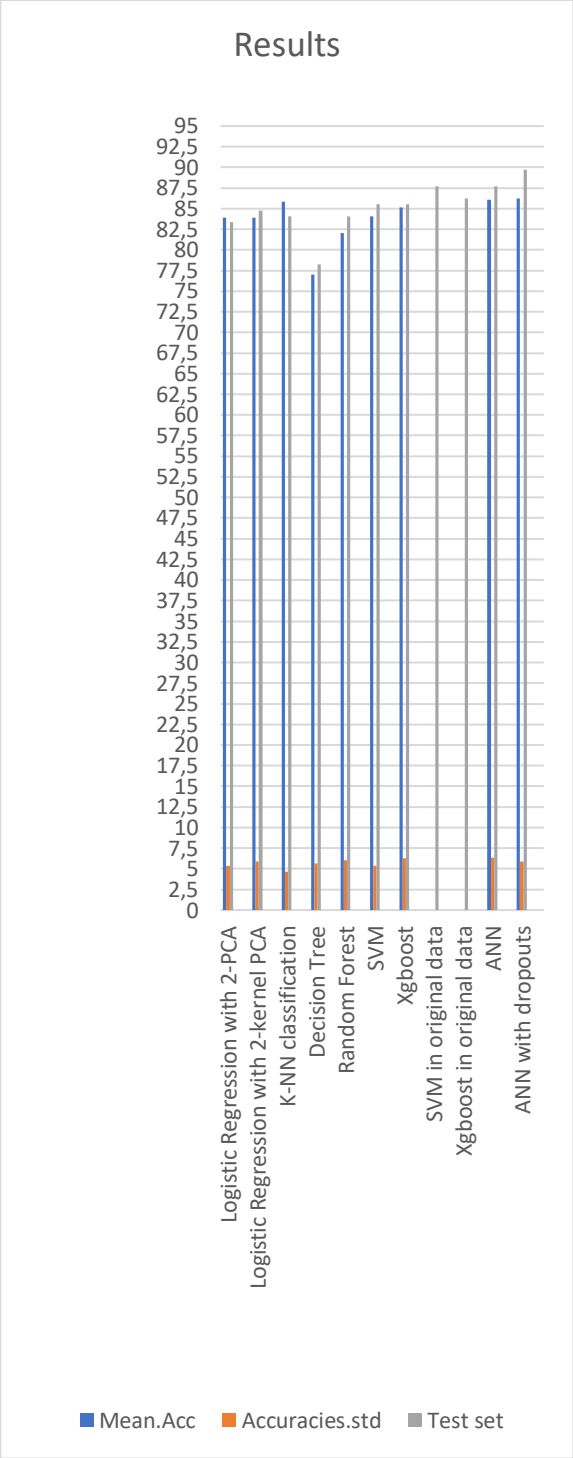
TABLE I
RESULT TABLE

Models	Mean.Acc(%)	Accuracies.std(%)	Test set(%)
Logistic Regression with 2-PCA	83,883116883116882	5,31730960292813 65	83,33333333333334
Logistic Regression with 2-kernel PCA	83,886363636363637	5.88846926186541 79	84,78260869565217
K-NN classification	85,876623376623373	4,62830340880107 94	84,05797101449275
Decision Tree	76,993506493506492	5,68912702452944 17	78,2608695652174
Random Forest	82,081168831168827	6,04776013758265 77	84,05797101449275
SVM	84,064935064935065	5,33681305848199 77	85,5072463768116
Xgboost	85,162337662337662	6,27677264622674 29	85,5072463768116
SVM in original data			87,68115942028986
Xgboost in original data			86,23188405797102
ANN	86,061688084881038	6,31841944318823 07	87,68115942028986
ANN with dropouts	86,243506039117845	5,89151169581245 28	89,70588235294118

We can observe from the results that by applying the **Dropout** method to the ANN that we produced the best result in the test set and that we reduced the standard deviation meaning that we somewhat fixed the problem of overfitting .As for the mainstream algorithms we can observe that xgboost and SVM perform quite well but SVM is actually better and more trustworthy since xgboost is easily overfitted and hence gives false good training error which leads to poor test error.

The moderate performance of neural networks may also suggest that it is very difficult to discover complicated hidden or latent variables associated with the response.

From the result table we notice that the training error is often higher than test error. We believe this might be caused by noise in our training data and apply some other data splittings may give different training error.



[

V. FRAUD DETECTION

We will use the self-organized map (SOM) algorithm. SOM is a type of NN and arranged the data in a two-dimensional grid placing similar members closer together.

Problem Description:

Customers are applying for cards. We need to find out customer who can potentially cheat. When we think about they potentially cheaters they are the outliers in SOM. Question is how can we deduct those outliers. Outliers will be far from its neighbourhood.In the NN we are going to get output neuron that is closest to the given customer (row). This neuron is called winning node. For each customer we would have one winning node. Then we use neighbourhood function to update the weight of the neighbourhood nodes. We do this for all the customers.

SOM Mapping: The outliers(frauds) will have a color close to 1 and must include both classes (red circle and green square).

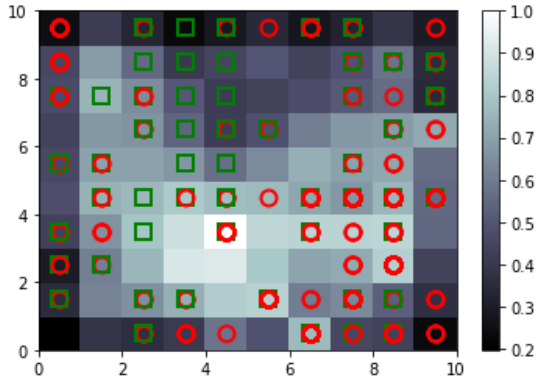


Fig. Self-Organizing Map

We take the “fraud” result for each customer and we add it to our data with a dummy variable . Probably fraud=1 otherwise 0. On the “new” dataset we will aply the ANN algorithm in order to find the probability of fraud for every customer.

```
array([[  1.57997850e+07,   6.63159089e-03],
       [  1.55858550e+07,   8.68631434e-03],
       [  1.56548590e+07,   8.97350442e-03],
       ...,
       [  1.55988020e+07,   3.29723269e-01],
       [  1.57627160e+07,   3.29723269e-01],
       [  1.57901130e+07,   3.29723269e-01]])
```

First attribute is the ID of the customer,second one is probability

We see that the biggest fraud probability is around 33%. That means that we are 33% sure the customers with the corresponding Ids should not have got a credit approval. Our fraud detection system would perform better if we had bigger data and frauds in order to train our model something that would be possible if we were working for a bank.

VII. Reference

References

[1] Classifier Comparisons On Credit Approval Prediction; by Zhoutong Fu; Zhedi Liu , 2014

[2] Quinlan. UCI Machine Learning Repository http://archive.ics.uci.edu/ml/ datasets/Credit+Approval

[3] Giuseppe Vettigli , Minisom implementation

[4] The self-organizing Map;Teuvo Kohonen ,1990