Kaggle Santander Customer Transaction Prediction Competition
A Machine Learning Challenge

Mohammadreza Iman
Computer Science Department
University of Georgia
Athens, GA 30602
`mi44512@uga.edu`

Denish Khetan
Institute of A.I.
University of Georgia
Athens, GA 30602
`denish.khetan@uga.edu`

Shannon Quinn
Computer Science Department
University of Georgia
Athens, GA 30602
`squinn@cs.uga.edu`

Abstract

The purpose of this paper is to analyze the performance of various machine learning methods using a dataset from a Kaggle challenge. The dataset is comprised of tabular (numerical) big data with a binary target. This dataset presents many challenges due to its statistical nature, unbalanced instances, and large size. We attempted several prepossessing methods combined with various machine learning techniques. This paper may serve as a useful road-map for anyone working through challenges with tabular, big data.

1 Introduction

Kaggle [1] is a popular platform focused on bringing machine learning-related challenges to the public. One of the ways Kaggle does this is by holding prized challenges for different companies. This paper uses the most recent Kaggle challenge from Santander Bank on customer transaction prediction [2]. The competition is a binary classification based on numerical features.
Tabular data and binary classification are common challenges in machine learning and classification. There are several common approaches for managing this type task such as using regressors, clustering techniques, decision trees, or neural networks. All of the machine learning techniques need some level of data preprocessing, which generally consist of data imputation, removing outliers, feature selection, scaling, and dealing with unbalanced data.
In this paper, we demonstrate our preprocess steps along with the machine learning techniques that we applied on the dataset, ending with the achieved results. The project was a part of the Data Science Practicum graduate-level course at the University of Georgia in Spring 2019 [3].

1.1 The competition

The goal of the challenge is to determine clients who will make a particular transaction in the future based on the client transaction history provided by Santander Bank, regardless of the measure of how much money they transacted. The information is anonymized, meaning it does not include a description of the features.

1.2 Data and Statistics

The dataset provided by Santander Bank for the Kaggle competition is completely anonymized with 200 numeric feature variables plus the binary target column. Both the training and testing datasets consist of 200,000 instances. The target value of the test dataset is not accessible, and the predicted result is evaluated by submitting it to the Kaggle challenge webpage. Importantly, no information is provided about the features, and competitors receive only tabular data. It is also worth mention that the values of each feature follow nearly a perfect normal distribution. See Figure 1 for example distributions [2].

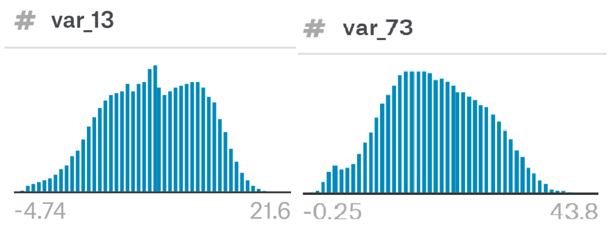

Figure 1: Example of Distributions [2]

As previously noted, this dataset presents several challenges. First, the training dataset is unbalanced with less than 15% of instances falling into class 1. Furthermore, there is no significant correlation between any of features and the target. For example, the highest absolute correlation of features to the target is about 0.08. Table 1 shows the top-ranked features along with their absolute correlation to the target.

Table 1: Features correlation to the target

Features	Correlation	absolute correlation
var_81	-0.08092	0.080917
var_139	-0.07408	0.07408
var_12	-0.06949	0.069489
var_6	0.066731	0.066731
var_110	0.064275	0.064275
var_146	-0.06364	0.063644
var_53	0.063399	0.063399
var_26	0.062422	0.062422
var_76	-0.06192	0.061917
var_174	-0.06167	0.061669
var_22	0.060558	0.060558
var_21	-0.05848	0.058483
var_99	0.058367	0.058367
var_166	-0.05777	0.057773
var_80	-0.05761	0.057609
var_190	0.055973	0.055973

Finally, the features appear unrelated to each other and there is little information that can be gained from any features. Figure 2 demonstrates the complicated nature of these features and the fact that it is impossible to note any pattern within them.

1.3 Document organization

The next section of this document details the machine learning approaches utilized. Although we would typically review related work prior to this, because the challenged recently concluded in March 2019 there is no existing published work to be reviewed. However, in the discussion section, we review the approach of the best-ranked team reported by Kaggle.
Section 2 thus focuses on the machine learning approaches examined. We begin by defining the preprocessing steps applied, followed by a description of the evaluation and feature-selection methods. We then briefly introduce the prediction and classification techniques that we applied under the subsection of methods. We close Section 2 by providing the performance results of the overviewed techniques.

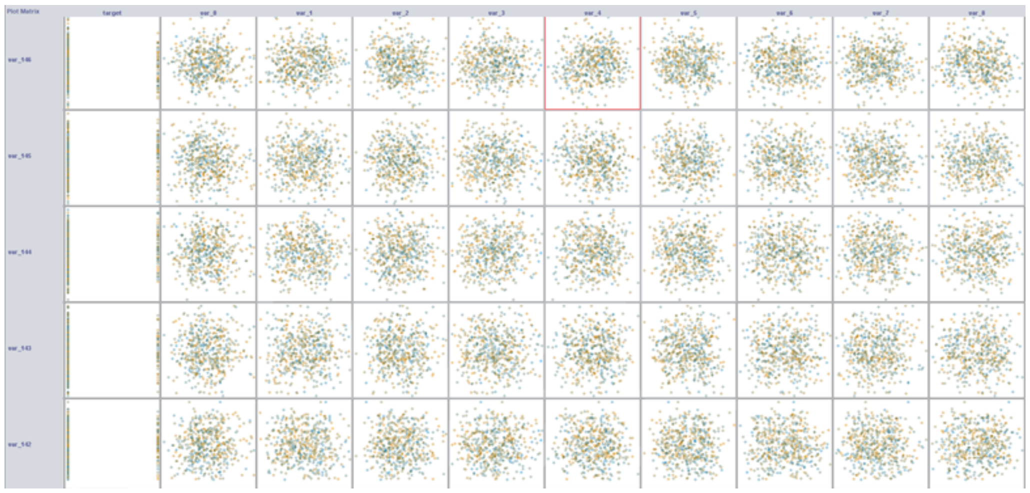

Figure 2: Some Features Visualization

Section 3 includes our analysis of machine learning techniques in addition to a review of the approach utilized by the competition's winning team.
The implementation for the best method can be found in Appendix I, while the all other implementations and details are accessible via the GitHub repository [4].

2 Machine Learning Approaches

In this section, we describe the preprocessing steps, followed by the evaluation methods. We then describe the machine learning techniques utilized before presenting the results of described techniques.

2.1 Data Preprocessing

We used machine learning concepts in our preprocessing to prepare the data in an effort to achieve better results. The training dataset is complete, and there are no missing values on any of the features; thus, no data imputation was needed. To remove outliers, we used the Z-score [5], which is the standard error of the mean. We set a Z-score of 3 and 4 as the threshold to remove outliers. Therefore, if any feature in an instance was more than 3 (or 4) standard deviations above or below the mean, these instances were removed a outliers. This process using Z-score of 3, resulted in the removal of approximately 12,000 instances (out of 200,000) from the training dataset.
To deal with the unbalanced data, we kept track of the weight for each class in the training set and used this weight in the methods. This process considers the weight as an input parameter like the Keras neural network libraries [6].
Finally, we used existing implemented libraries [7] to scale all the values to prepare them for certain machine learning techniques. This is necessary because such techniques – for example, sum-product networks [8], neural networks and deep learning methods – are sensitive to wide-ranging values for features.

2.2 Evaluation methods

The evaluation for the test dataset prediction was done through the Kaggle website using Area Under the Receiver Operating Characteristics (AUROC) [9] as a measurement for accuracy. The Area Under the Receiver Operating Characteristic is a typical summary statistic used for the verification of a predictor in binary classification.
Moreover, we used the Confusion Matrix to achieve a better comparison between methods we applied on the training dataset. To make our comparison more reasonable and also to better examine the performance of the methods to later fine-tune them, we split the training dataset into a training and validation set (20% of whole training dataset).

2.3 Feature Selection

We applied three different methods for selecting features. First, we used the Recursive Feature Elimination (RFE) method of Logistic Regression [10], which resulted in 150 selected features, ranked 1 out of 200 features. Second, 73 features were selected using the Random Forest and Feature Importance Method [11]. Third, we selected 27 features based on their higher ranked correlations to the target.
We applied these methods on all three sets of selected features, as well as on all features to obtain the best result. It is worth highlighting that the best result was achieved using the set of 73 selected features.

2.4 Methods

2.4.1 Trees

We applied two tree methods: Decision Tree and Random Forest, which are described below.

Decision Tree:

The Decision Tree [12] is the most well-known and widely-used method for classification and prediction. Decision trees are constructed using an algorithmic approach that predicts ways to split a dataset based on various conditions. The Decision Tree [12] is also a very common method for supervised learning. Even though decision trees are easy to understand and apply, a primary critique of them is that they can easily overfit the data. This is typically handled with pruning techniques.

Random Forest:

Random Forest [13] is a tree-based ensemble learning method which can be used for classification and regression. It works by constructing a multitude of decision-trees at training time and outputs the class that is the mode of classification (classes) or regression (mean prediction) of the individual trees. These trees generally work quickly and accurately but, like decision trees, they sometimes suffer from overfitting.

2.4.2 Clustering

Support Vector Machine (SVM) [14] is another very popular machine learning algorithm that can handle both classification and regression including identifying outliers. SVM tries to find an optimal hyperplane that categorizes the new inputs. There are multiple kernels that can be used to find the optimal hyperplane, including Gaussian, sigmoid, and polynomial. Because the data involved are numerical, complex, and have a high degree of dimensionality, the Gaussian method tends to work faster and better than any other kernel type.

2.4.3 Artificial Neural Network (ANN) and Deep Learning (DL)

Artificial Neural Networks (ANN) [15] are a machine learning technique inspired by biological neural networks. Artificial Neural Networks are capable of modeling and processing nonlinear relationships between inputs and outputs. Because they are modeled using layers of artificial neurons, they utilize adaptive weights along paths between neurons. These weights can then be tuned by a learning algorithm that learns from the observed data in order to improve the model. The simplest setup of ANN contains the first layer as the input layer, one hidden layer, and an output layer with one or more neurons in each layer. ANN can become very complex depending upon the number of hidden layers and neurons in each layer. Again this increases the chance of overfitting the data. Deep Learning (DL) [16] is another type of neural network which takes raw input data and processes it through many layers of nonlinear transformations in order to calculate a target output.

2.4.4 Sum-Product Network (SPN)

The Sum-Product Network (SPN) is a new type of machine learning model that uses fast, exact probabilistic inference over many layers. SPN can be pictured "as a rooted directed acyclic graph with univariate distributions as leaves, sums and products as internal nodes, and the edges from a sum node to its children labeled with the corresponding weights" [8]. This type of technique learns in two steps: first, by constructing the network structure based on the features from the training set; and second, by updating the weight on the edges similar to ANN using the training data set. SPN can lead to impressive outcomes on datasets such as image completion, image classification, activity recognition, and collaborative filtering.

2.5 Results

Table 2 demonstrates the results of each applied technique on the validation set from the training set. It also demonstrates the results on the test set that was obtained by submitting the predictions to the Kaggle website. We did not submit the results of the regressions [17] to Kaggle due to their poor performance. The best result was obtained using the random forest technique [7] followed by the neural network technique implemented using Keras [6].

Table 2: Results

Model	Validation set Acc	Test set ACC
Simple regression	0.1799	NA
Ridge regression	0.1799	NA
LASSO regression	0.1799	NA
Decision Tree	0.8396	0.5662
Random Forest	0.7241	0.8050
Support Vector Machine (Kernel:Sigmoid)	0.9005	0.5003
Support Vector Machine (Kernel:Poly)	0.9000	0.5000
Support Vector Machine (Kernel:Gaussian)	0.9101	0.5539
Sum-Product Network	0.6768	0.6753
Neural Network	0.7224	0.7253

3 Discussion

Here we review the steps of the competition's winning team, who managed to achieve a score of around 0.90. This information was accessed through the discussion board on the Kaggle website [2]. The leading team reported that they applied several manual outlier removal processes and feature selection methods. It seems this may have been the only way to achieve such a high score on this challenge. It is worth noting, however, that this general way of applying machine learning techniques is not widely accepted in academic circles. This is due to the fact that it is based on manual processes and therefore is very specific to a dataset rather than a type of dataset. Given this, we sought to find a more general solution for this challenging tabular data without engaging in manual analyzation. However, reproducing the result of the best-ranked team was not our aim in this project. We attempted many different hyperparameters for each method and tried our best over the limited time of this project. We believe future work may achieve better results by allocating more time to this challenge, which would allow for examining other deep learning architectures as well as an additional implementation of sum-product networks.

References

[1] https://www.kaggle.com/

[2] https://www.kaggle.com/c/santander-customer-transaction-prediction

[3] https://dsp-uga.github.io/sp19/index.html

[4] https://github.com/dsp-uga/team-Squadron-final/

[5] Cousineau, D., & Chartier, S. (2010). Outliers detection and treatment: a review. International Journal of Psychological Research, 3(1), 58-67.

[6] https://keras.io/

[7] https://scikit-learn.org/stable/

[8] Poon, H., & Domingos, P. (2011, November). Sum-product networks: A new deep architecture. In 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops) (pp. 689-690). IEEE.

[9] Hajian-Tilaki, K. (2013). Receiver operating characteristic (ROC) curve analysis for medical diagnostic test evaluation. Caspian journal of internal medicine, 4(2), 627.

[10] Granitto, P. M., Furlanello, C., Biasioli, F., & Gasperi, F. (2006). Recursive feature elimination with random forest for PTR-MS analysis of agroindustrial products. Chemometrics and Intelligent Laboratory Systems, 83(2), 83-90.

[11] Saeys, Y., Abeel, T., & Van de Peer, Y. (2008, September). Robust feature selection using ensemble feature selection techniques. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 313-325). Springer, Berlin, Heidelberg.

[12] Safavian, S. R., & Landgrebe, D. (1991). A survey of decision tree classifier methodology. IEEE transactions on systems, man, and cybernetics, 21(3), 660-674.

[13] Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. R news, 2(3), 18-22.

[14] Scholkopf, B., & Smola, A. J. (2001). Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT press.

[15] Zurada, J. M. (1992). Introduction to artificial neural systems (Vol. 8). St. Paul: West publishing company.

[16] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436.

[17] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Methodological), 58(1), 267-288.

Apendix I

Python 3.5.x code:

```
This program is a part of data prediction for Kaggle Santander Bank
    challenge
This code is for random forest model, it gets train on train set and
    predict on test set and outputs the CSV file for submit to Kaggle.
This code has the best adjusted parameters through several tuning and
    submissions.
This code is implemented using many great directions from https://
    machinelearningmastery.com/feature-selection-machine-learning-python/
'''


import numpy as np
import csv
import numpy
from scipy import stats
from numpy import loadtxt
from sklearn.model_selection import train_test_split
from sklearn.model_selection import StratifiedKFold
from sklearn.ensemble import ExtraTreesClassifier
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.utils import class_weight
from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree
    Classifier
from sklearn.model_selection import train_test_split # Import
    train_test_split function
from sklearn import metrics #Import scikit-learn metrics module for
    accuracy calculation
from sklearn.ensemble import RandomForestRegressor
from sklearn.feature_selection import SelectFromModel

seed = 5
numpy.random.seed(seed)

# Read the CSV into a pandas data frame (df)
```

```
dftr = pd.read_csv('train.csv', delimiter=',')
train_data = np.array(dftr)
dfte = pd.read_csv('test.csv', delimiter=',')
test_data = np.array(dfte)

#Detecting and Removing outliers
train_data = np.array(train_data[:,1:(len(train_data[0]))], dtype=np.
    float)
train_data = train_data[(np.abs(stats.zscore(train_data)) < 4).all(axis
    =1)]

#read csv files to arrays and convert types
XX = train_data[:,1:(len(train_data[0]))]
YY = train_data[:,0]
TT = test_data[:,1:(len(test_data[0]))]
R = test_data[:,[0]]

T = np.array(TT, dtype=np.float)
X = np.array(XX, dtype=np.float)
Y = np.array(YY, dtype=np.int)

#Filter selected features
sel_f= np.array([ True,  True,  True, False, False, False,  True, False,
    False,True, False, False,  True,  True, False, False, False, False,
    True, False, False,  True,  True, False, False, False,  True, False,
    False, False, False, False,  True,  True,  True, False, False, False,
     False, False,  True, False, False,  True,  True, False, False, False
    , False, False, False,  True, False,  True, False, False,  True,
    False, False, False, False, False, False, False, False, False, False,
     False, False, False, False,  True, False, False, False,  True,  True
    , False,  True, False,  True, True, False, False, False, False,  True
    , False, False,  True, False,  True,  True, False,  True,  True,
    False, False, False, True, False, False, False, False, False, False,
     True,  True, True,  True,  True, False, False, False, False,  True,
    False, False, False,  True, False,  True,  True, False, False, False,
     False,  True, False, False,  True, False, False,  True, False, False
    , False, False, False,  True, False,  True, False, False, False,
    True,  True,  True,  True, False, False, False, False, False,  True,
     True, False,  True, False, False, False, False, True,  True,  True,
     True,  True, False, False,  True,  True, False,  True,  True,  True,
     False, False,  True, False,  True, True, False, False, False,  True,
     False, False, False,  True, False,  True,  True, False, False, False
    , False, False,  True, True, False])

T = T[:,sel_f]
X = X[:,sel_f]

#Dividing the train data to train and validation sets for scaled and
    unscaled data
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.15,
     random_state=1)

# Feature Scaling
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
T = sc.transform(T)

#RandomForest model
regressor = RandomForestRegressor(n_estimators = 100, random_state = 55)
# Train the model on training data
regressor.fit(X_train, y_train)

y_pred = regressor.predict(X_test)
```

```
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred)
    )
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(
    y_test, y_pred)))

ynew = regressor.predict(T)
np.savetxt("submission14.csv", ynew,header="target")
```