ECE 6254: Statistical Machine Learning Project Final Report

Credit Card Fraud Detection

Allegra Allgeier, Clara Daniels, Derek Wu, and Li Wei (Dave) Yap

1. Project summary

The problem of credit card fraud detection gives us a unique opportunity to analyze a highly skewed real-world dataset. We accomplished an extensive pre-processing of the data including resampling and dimensionality reduction, and applied supervised learning for detecting fraudulent transactions. We performed a comparative analysis of the performance of machine learning methods including Naive Bayes, Logistic Regression, kNN, Extreme Gradient Boosting, Neural Nets, and Support Vector Machines. The “winner” among these algorithms based on accuracy, precision, recall and the f1-score is described in detail in subsequent sections.

2. Introduction

According to the Nilson Report, in 2018 there was a global loss of $28 billion due to card fraud [1]. As credit card fraud is one of the common consumer frauds, detecting it is a relevant problem for today’s economy. The goal of this project is to correctly classify previously unseen transactions as fraudulent or normal. Our dataset is taken from a Kaggle competition [2]. Since our dataset contains many more data points of normal transactions than fraudulent transactions, our first goal is to conduct preprocessing to prevent overfitting.

3. Data Pre-Preprocessing

The original given datasets for the training data are “train_transaction” and “train_identity”. The first dataset provides information on transaction time, amount, address, distance, card type and other such information about the payment transaction. The latter dataset provides identity information on network connection information (IP, ISP, Proxy, etc) and digital signature (UA/browser/os/version, etc) associated with a transaction. Since our dataset is from a competition, the test dataset provided does not contain the true labels. Therefore we use a portion of our training dataset as the test dataset. Since the “train_identity” dataset does not contain the identity information for all samples in the “train_transaction” dataset, we will use samples containing both the transaction and identity information for our project.

Data Reduction:

First, we merge “train_transaction” and “train_idenitity”. We then remove features with NaN ratio higher than 0.5. Next, the data is normalized and we apply a low variance filter. We then perform feature selection by using correlation of the data, reducing the number of features in the dataset [3]. After replacing the remaining NaN entries of a numerical feature with the mean and NaN entries of a categorical feature with a dummy category, we apply Principal Component Analysis (PCA) to numerical data, and apply Multiple Correspondence Analysis (MCA) to categorical data.

Data Resampling:

Before the resampling, we split the reduced data into training dataset and testing dataset. The training dataset is created by selecting 75% of the dataset and the testing dataset is created by selecting the other 25%. Since the classes are not balanced, in order to prevent bias in our models we conduct resampling on the training dataset using SMOTE (synthetic minority 'oversampling' technique) [4].

SMOTE synthesizes data points from the existing pool of the minority class and adds them to the dataset. This technique ensures there's very little data leakage by creating new, unseen data points for the model to train on.

The sizes of the input and final datasets is shown in Table 1. Our original dataset had 394+41 features, which is reduced to 64 in the final training and testing datasets.

Dataset	Number of datapoints	Number of Features
train_transaction	590540	394
train_identity	144233	41
Final training dataset	199270	64
Final testing dataset	36059	64

Table 1. Dimension of original and final training and testing datasets.

4. Performance Metrics

The classification performance is measured by the following metrics. **Accuracy** is defined as the number of correct predictions out of the total number of predictions. **Precision** corresponds to the ratio of correctly predicted positive data points to the total number of predicted positive observations. **Recall** is the ratio of correctly predicted data points of a class out of the total number of actual data points of that class. The **f1-score** refers to the weighted average of precision and recall.

5. Machine Learning Models

Naive Bayes (NB):

In obtaining a Naive Bayes classifier on our dataset, we will model the conditional distribution of our features by a Gaussian distribution. Since the features that were originally categorical were transformed to numerical features through the MCA, they will also be modeled by a Gaussian distribution instead of a multinomial distribution.

Binary Logistic Regression (LG):

We will apply the standard binary logistic regression to our data.

K Nearest Neighbors (kNN):

In our experiment, we calculated the scores for varying k values. As we can see from the figures, accuracy, precision, and f1-score seem to decrease as we increase the number of neighbors. In contrast, the recall has a sharp increase at first and then starts to decrease as k gets larger. In the section for comparison among models, we use the result from k = 5 since all four scores are relatively high. On Google COLAB, the naive

Bayes model and the logistic regression model ran in less than 10 seconds. However, the kNN model took about 40 minutes to train and test.

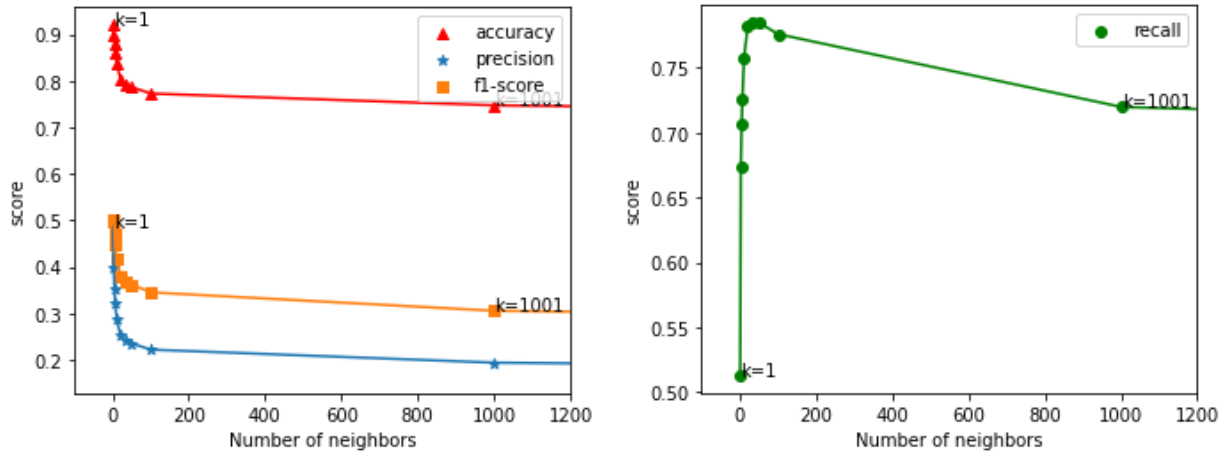

Figure 1: Scores for kNN for k = 1, 3, 5, 7, 11, 21, 35, 51, 101, 1001.

Support Vector Machine (SVM):

Support Vector Machines (SVM)'s find a hyperplane that separates N-dimensional data into two classes, where N is the number of features in the dataset. The support vectors describe the data points that lie closest to the hyperplane, which if removed, would change the position of the hyperplane. The distance between the hyperplane and nearest data point is called the margin. The SVM chooses the hyperplane with the largest margin such that the training data is classified correctly.

We experimented with different kernel functions, which is simply the mapping function, and different values of parameter C, which controls the complexity of the decision boundary.

For a radial basis SVM, we tried varying the value of parameter C, which describes the tradeoff between misclassification of training samples and the simplicity of the decision surface. Some of the performance results for different values of C is given in Figure 1. The default value of C is 1, which is a fairly good value for our dataset, with a decent tradeoff between accuracy and model complexity.

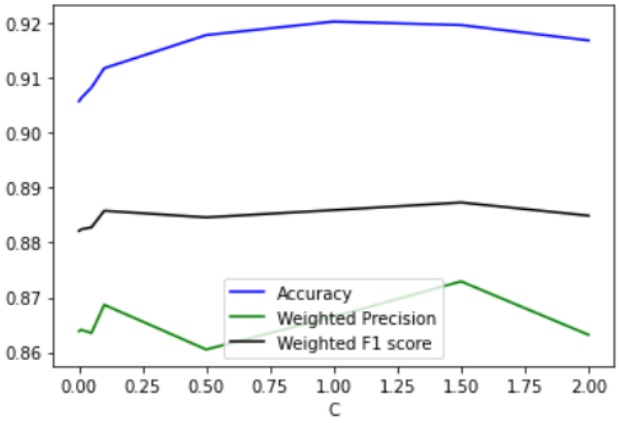

Figure 2. Accuracy, precision, and F1 score for a radial basis SVM by varying C.

We started with the simplest kernel function for the SVM: a linear kernel, then tried a Radial Basis Function kernel, and polynomial kernel. Linear SVM took 4 minutes to train and test, compared to 120 minutes for Radial Basis SVM and 125 minutes for Polynomial SVM.

Neural Networks:

The fully connected feedforward neural network consists of layers of neurons or *nodes* [5]. It is fully connected because a node is connected to all of the nodes in the following layer. Feedforward means that an output of a layer does not become an input for a previous layer. The first layer of this network takes the dataset as input. In the intermediate layers, certain computations are conducted on the nodes. For our project, we used a transformation involving a matrix multiplication followed by an addition of bias. In the final layer, we classify the outcome as either “fraud” or “not fraud”.

We also experimented with convolutional neural networks (CNN). CNN is known for pattern recognition, and applications include image classification and speech recognition.

The network structure we tried were: [1]d(60) d(1), d(90) d(1), d(30) d(30) d(1), d(60) d(30) d(1), d(60) d(60) d(1), conv1d(60) d(5) d(1). The activation functions in the first and the last layers are ‘relu’ and ‘sigmoid’, respectively. The loss function and the optimizer is the binary cross entropy and Adam. The model was trained on 10 epochs. We utilized keras in the tensorflow package for implementation.

Random Forest (RF) :

A random forest is a bagging method that is utilized to help reduce variance and to avoid overfitting. For the random forest, we have two main types of randomness. First, each tree is built on a bootstrap sample from the original data. Second, at each tree node, a subset of features is randomly selected to generate the best split.
For RF, we tried varying the number of trees while having the maximum depth of a tree set to 9. We use Gini as the quality of a split for parameter tuning.

Extreme Gradient Boosting (XGBoost) :

XGBoost is a decision-tree-based model with a gradient boosting framework. Boosting trains models with each new model being trained to correct the errors made by the previous ones. That is why XGBoost is able to improve upon the base framework through parallelization and tree pruning. We experimented with different learning rates and MSE to evaluate the performance of the Extreme Gradient Boosting.

6. Comparison of Machine Learning Algorithm Performances

We provide a statistical comparison of the performance of each model in Table 2 and Figure 4. Gaussian Naive Bayes performed the worst out of all of the models, with extremely low accuracy and precision. Its perfect recall suggests that it predicts almost all transactions as fraudulent. The binary logistic regression was slightly better and it

[1] ‘d’ represents ‘Dense’ layer. Every first(input) layer has activation function ‘relu’; every last(output) layer has activation function ‘sigmoid.’

predicted more transactions as non-fraud. This can be seen from its higher accuracy, precision, and f-1 score with a slightly lower recall. kNN performed even better.

Among the SVM's, the linear SVM performed the worst. In contrast, radial basis SVM and polynomial SVM performed extremely well. The radial basis SVM seems to be the winner among the three, as it achieved scores higher than 0.9 in all four categories.

The neural nets achieved the highest accuracies among all of our models. However, their precision, recall, and f-1 scores are lower than the radial basis SVM. XG Boost turned out to have an average performance among all models.

	Accuracy	Precision	Recall	f1-score
Gaussian Naive Bayes	0.08	0.08	1	0.14
Binary Logistic Regression	0.67	0.17	0.86	0.28
kNN, k=5	0.88	0.35	0.71	0.47
Linear SVM	0.47	0.12	0.93	0.22
Radial Basis SVM	0.93	0.94	0.19	0.31
Polynomial SVM	0.82	0.26	0.68	0.38
Random forest	0.80	0.28	0.64	0.34
XGBoost	0.86	0.32	0.50	0.35
d(90)d(1)	0.94	0.67	0.38	0.49
d(60)d(30)d(1)	0.94	0.9	0.2	0.32
conv1d(60)d(5)d(1)	0.94	0.77	0.28	0.41

Table 2: Performance of models. The highest score in each column is highlighted.

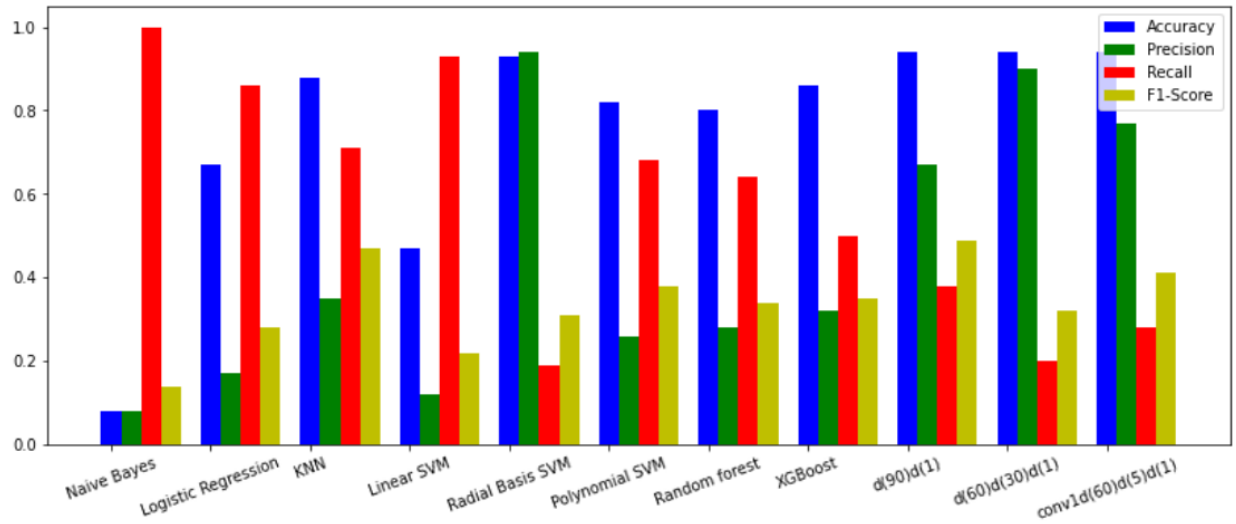

Figure 4. Comparing the testing performance of models.

7. Conclusions

In the time of big data, preprocessing plays an important role. As our data was highly skewed and contained many features, resampling and dimensionality reduction proved to be extremely important for applying our models.

Basic models (Naive Bayes, Logistic Regression, and KNN) tend to have higher recall than precision, suggesting that they correctly classify most of the fraudulent transactions, but they also classify many non-fraudulent transactions as fraudulent.

Linear SVM likely had low accuracy because the hyperplane could not separate the linearly mapped data very well, ie, the data was not linearly separable. More complex kernels resulted in mappings that better separated the different classes.

The neural networks had the highest accuracy, slightly higher than the radial basis SVM, and intermediate precision, but the recall was much lower. This suggests that the majority of what neural networks classified as “fraudulent” was indeed fraudulent, but they did not return many of the fraudulent transaction data points as fraudulent, instead classifying several fraudulent transactions as normal.

In fraud detection, what we want to avoid is overlooking a fraudulent transaction. Therefore the score we care about in our case is the recall. However, as we can see from the Naive Bayes result, simply having a high recall does not mean that the model is good. Hence we also compared the f-1 scores. In the end, we conclude that kNN performed well on our dataset, as it has a high recall and f-1 score. By these standards, Polynomial SVM and Random Forest also have among the best performance and are worth considering for future implementation on such datasets such as this one.

Division of Labor

Allegra and Clara read papers related to background methods in this field, Dave and Allegra applied themselves to Exploratory Data Analysis (EDA), Dave resample the data, and Clara and Derek worked on dimensionality reduction through PCA and MCA.

In the mid stage of the project, we each focussed on a different machine learning algorithm to analyze the data. Allegra applied KNN, Naive Bayes, and logistic regression, Derek used Neural Networks, Dave applied a random forest with boosting, and Clara used Support Vector Machines (SVM)’s. In the final stage of the project, Derek and Dave used the performance measurements of all our machine learning methods to compare the performance on our dataset, Clara and Allegra focussed on writing the proposal and final report, and Allegra and Derek worked on the final video.

References

[1] https://nilsonreport.com/mention/407/1link/

[2] https://www.kaggle.com/c/ieee-fraud-detection/discussion/101203

[3] Feature Selection by Correlation https://en.wikipedia.org/wiki/Feature_selection#Correlation_feature_selection

[4] Chawla, N. V.; Herrera, F.; Garcia, S.; Fernandez, A. (2018-04-20). "SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-year Anniversary". *Journal of Artificial Intelligence Research*. **61**: 863–905.

[5] Schmidhuber, Jürgen (2015-01-01). "Deep learning in neural networks: An overview". *Neural Networks*. **61**: 85–117.

[6] "Boosting algorithm: XGBoost". *Towards Data Science*. 2017-05-14.