Machine Learning 21F Final Project

Tina Buzanis[1]

December 17, 2021

Abstract This project analyzes two datasets: a dataset of credit card transactions, and a dataset of energy efficiency for various building shapes. For the credit card transactions, the goal is to manage the imbalanced nature of the dataset in a way that allows for accurate predictions of whether or not a transaction is fraudulent. The second dataset holds information about many different types of buildings, to be used for the purpose of predicting the heating and cooling load.

Credit Card Fraud

The Credit Card Fraud Detection dataset is a set of transactions made by European cardholders over two days in September 2013. This dataset consists of 284,807 total transactions, 492 of which are tagged as fraud. In total, there are 30 features: Time, Amount, and 28 features labeled V1 though V28—the principal components obtained through Prinicipal Component Analysis on the original dataset.

As imbalanced datasets may adversely affect the outcomes of the models' predictions, it is important to take steps to balance our data set. This was done in two main ways: undersampling and oversampling. Before either method was implemented, the Time and Amount columns were scaled with sklearn's RobustScaler, as it is less sensitive to outliers—of which this dataset has many. In both cases, the performance was evaluated with five methods: Logistic Regression, K-Nearest-Neighbors, Support Vector Machines, Decision Trees, and a simple Neural Network.

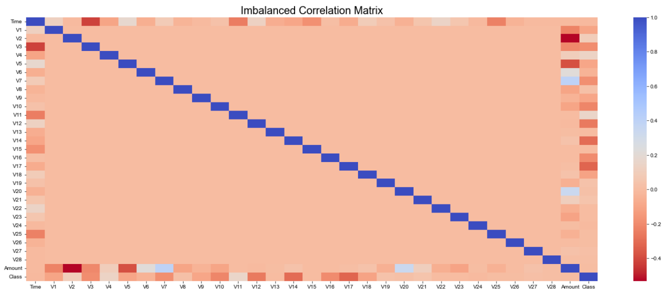


Figure 1: Before Balancing

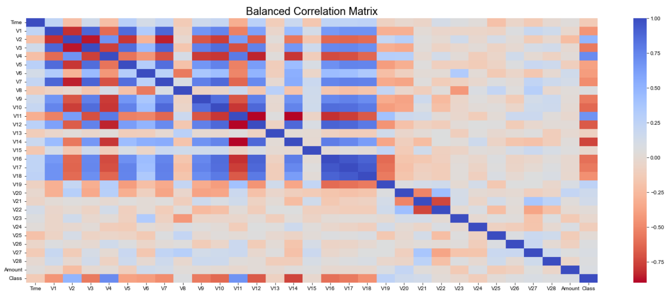


Figure 2: After Balancing

I. Undersampling and EDA

This dataset is highly unbalanced, with the 492 cases of actual fraud constituting only 0.17% of the dataset. In order to proceed, Random Undersampling was chosen as a first approach. This was performed by shuffling the data, isolating the first 492 instances of non-fraud transactions, and concatenating them with the 492 fraud instances.

Next, the outliers were removed based on the Interquartile Range Method—a data point was removed if it existed outside a threshold created by extending the interquartile range by 150%. This optimal results were obtained by performing the removals four times - twice each for fraud and non-fraud cases.

After balancing the dataset and removing outliers, I plotted the results from running the T-SNE, PCA, and TruncatedSVD. The results illustrated that the predictive models should be relatively successful:

II. Results

The accuracy and cross-validation scores, along with plots the learning curves from the training of the models are pictured below.

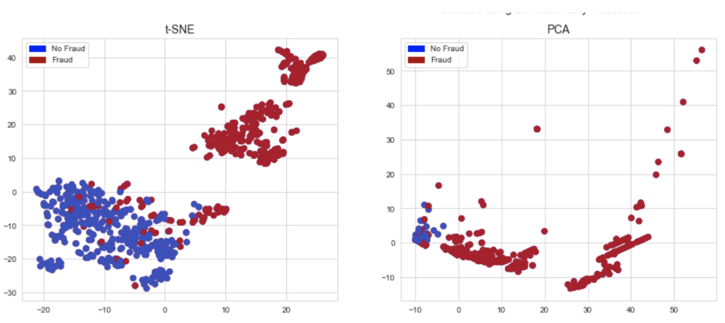


Figure 3: TSNE and PCA

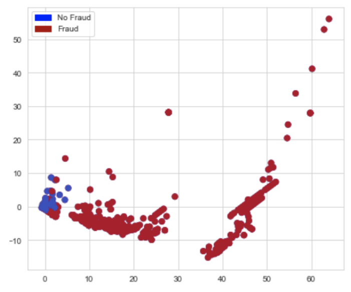


Figure 4: TSVD

	Accuracy Score	Cross-Validation Score
Logistic Regression	94	96.41
KNN	95	95.48
SVC	96	95.9
Decision Tree	91	93.84

Table 1: Initial Scores

The initially high scores, along with the proximity of the Logistic Regression and SVC learning curves may indicate a case of overfitting. As such, the approach was revised, and the fitting / predictions were done in a manner in which the undersampling (this time implemented with the NearMiss algorithm) was performed during cross-validation, not before:

The ROC curves continued to show promising results. Of the classifiers used, Logistic Regression consistently performed the best.

III. Oversampling

Next, oversampling was performed using the Synthetic Minority Oversampling Technique (SMOTE). Instead of removing data points to create balance, SMOTE creates synthetic data points by observing the distances between the closest neighbors in the minority class. The oversampling was performed during cross-validation, resulted in solid scores:

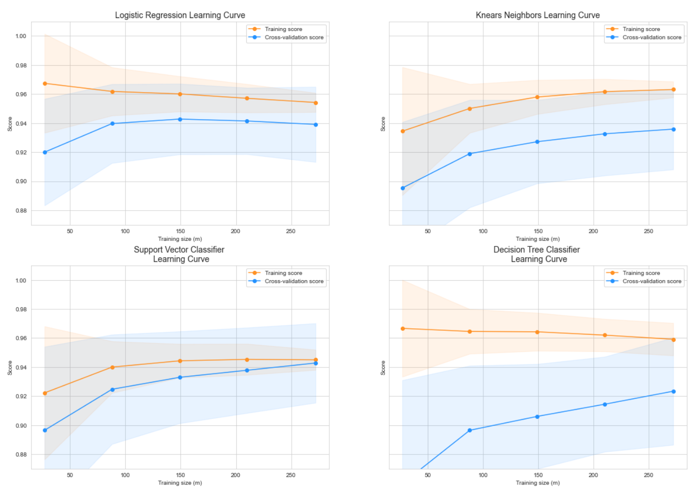


Figure 5: Learning Curves

	Before	During
Recall	0.98	0.92
Precision	0.72	0.01
F1	0.83	0.02
Accuracy	0.80	0.02

Table 2: Score Comparison

Accuracy	0.947
Precision	0.091
Recall	0.894
F1	0.153

Table 3: SMOTE Scores

The overall performance of each classifier using the SMOTE sampling strategy can be seen below:

In the end, the SMOTE strategy outperformed the Random Undersampling Strategy by ten percentage points, resulting in scores of 0.997 amd 0.906 respectively. In both cases, Logistic Regression performed the best.

Finally I implemented a simple Neural Network in order to see which of the logistic regression models (undersampling or SMOTE) presented with better accuracy. This Neural Network has one hidden layer with 32 nodes, and one output node comprised of the two possible results (0 or 1). The learning rate was decided to be 0.001, along with the AdamOptimizer and the Relu activation function. Lastly, for the final outputs, spare categorical cross-entropy was used.

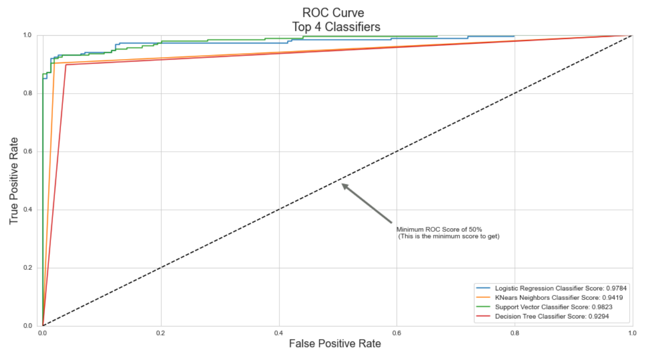


Figure 6: ROC Comparisons

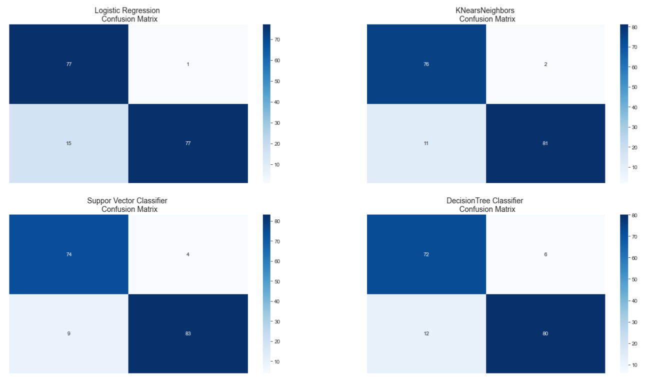


Figure 7: SMOTE Comparisons

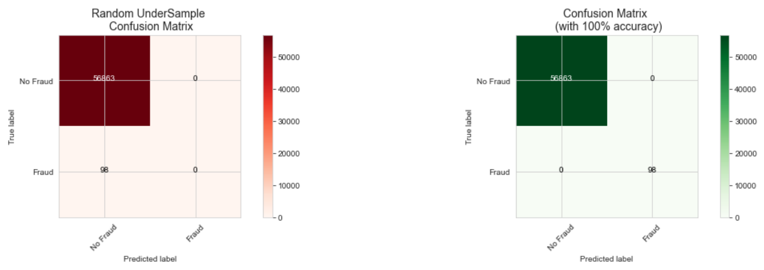


Figure 8: NN with Undersampling

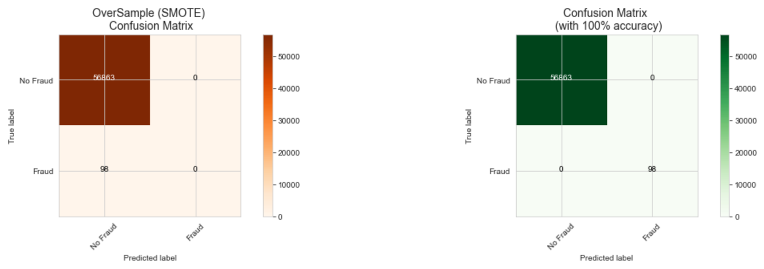


Figure 9: NN with Oversampling

IV. Discussion

Perhaps unsurprisingly, it is evident that even with the Neural Network, the Logistic Regression continued to perform best, along with the SMOTE sampling strategy. This is likely because the Random Undersampling strategy takes into account only a fraction of the data points accessible with the SMOTE method. The resulting data loss likely affects the ability of the algorithms to learn from the data. Additionally, SMOTE may have been helpful in the context of the many outliers.

Energy Efficiency

The Energy Efficiency dataset is a dataset with 10 features, and 768 samples. It details the heating and cooling requirements of various shapes of buildings. The goal with respect to this dataset is to be able to predict the heating and cooling load given building features.

I. EDA

Unlike the previous dataset, the data in this set is relatively evenly distributed, and has a mix of continuous and categorical features .

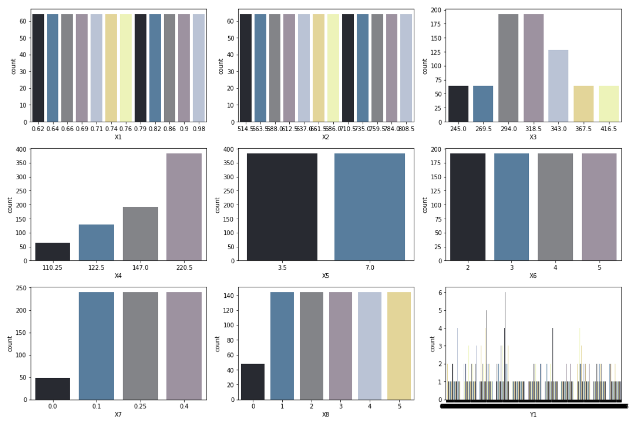


Figure 10: Energy Feature Distributions

The challenge presented with this dataset was with respect to its multicollinearity—there are many instances of features being nearly entirely negatively correlated or being entirely positively correlated. This is an issue because it has the potential to undermine the statistical significance of a given independent variable.

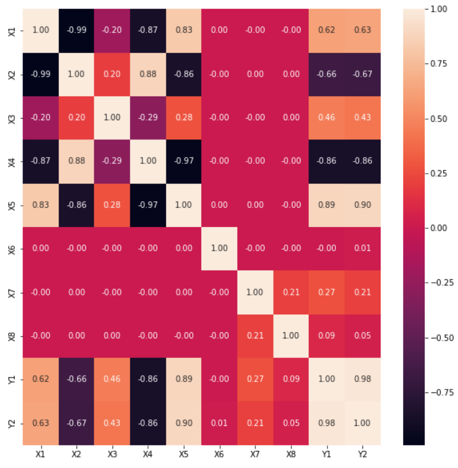
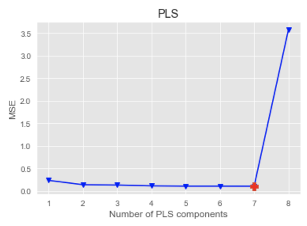


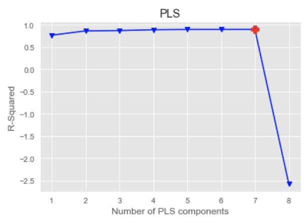
Figure 11: Energy Feature Correlationss

II. PLSR

The first approach was with Partial Least Squares Regression (PLSR). As PLSR seeks to maximize the covariance between X and y, it appeared a good fit for the problem of multicollinearity. Initially, a PLSR model was created with a number of components equal to the number of features (8). The number of components was then optimized by creating a model with a range of 1...8 components, and tracking the number of components that correlated with the best mean squared error (MSE) and R^2 values. The optimal PLSR model was with 7 components, and has the following scores: R^2: 0.892, MSE: 0.108. The training set resulted in a 0.90 accuracy score, with a std of 0.01. The model performed marginally worse on the test set, though still well, with an accuracy score of 0.89, and a std of 0.02.



(a) MSE



(b) R^2

Figure 12: No. Components for PLS

III. Ridge Regression

For Ridge Regression, RandomizedSearchCV was used in order to pick the optimal value for the alpha parameter, which was approximately 0.234. Both the training and testing scores were extremely similar to those from the previous section:

	Training	Test
R^2	0.899	0.91
MSE	0.101	0.094
Accuracy	0.89	0.90
std	0.01	0.01

Table 4: Ridge Regression Scores

Though the R2 is slightly lower, and the MSE is slightly higher, leading to the conclusion that this model performed slightly better than PLSR.

IV. Polynomial Regression

Similarly to the optimization process for the PLS Regresion, the parameters for the Polynomial Regression wre tuned manually by tracking the scores of models of degrees varying between 1 and 6. It was determined that the optimal degree was 4. The training set resulted in a score of 0.89, while the test set resulted in a score of 0.90. i

V Discussion

The challenges presented with this dataset were unique challenges that I had not yet worked with, though it was an interesting journey, from which I learned a lot. Despite the initial issues of multicollinearity, correct model selection and hyperparameter tuning appeared to give solid results. Had I more time, I would have visualized the outputs of the models, and spent more time analyzing their performances, then iteratively improving.