Machine Learning – Final Project

Eddie Tran

I. INTRODUCTION

Over the course of a semester, we were exposed to many different machine learning techniques used for either classification, regression, or both. In this report, I test my knowledge and applied techniques that will help us identity the proper model to best fit our different datasets.

II. CLASSIFICATION DATASET

A. *The Dataset*

The classification dataset, Credit Card Fraud Detection, from Kaggle is a highly unbalanced dataset with a dimension of [284807, 31]. This dataset provides a mixture of non-fraud and fraud transactions. As the dataset is processed, I have come to conclude that the value of non-fraud comes to 284315 transactions while the value of fraud is 492 transactions.

B. *Data Processing*

The dataset determines fraud and non-fraud as 1 or 0, respectively, via the "Class" column, so *X* will become our data that is not "Class" while *y* will become our data that is and only is "Class." Next, I split our *X* and *y* into two subsets: training and testing data. Then, I applied feature scaling which will normalize our training and testing data. With this, we can continue to train our classification techniques.

C. *Logistics Regression*

My first algorithm is Logistic Regression which predicts a binary outcome based on a set of independent variables. I took the sets and trained, fitted, and predicted onto the Logistics Regression model. Next, I computed and displayed the confusion matrix shown as Figure 1.

With this matrix, we can determine if our model best fits this dataset. We can calculate the recall and precision score which is important due to the binary outcome of our fraud and non-fraud transactions.

However, since this dataset is heavily imbalanced, we will not receive a proper value as it is heavily biased towards our non-fraud cases due to its extremely large value. Our values come out to recall: 65% and precision: 87%. This results in false positives being more biased towards our model, which is what we do not want. We can say that doing normal Logistics Regression is unfit for our goal.

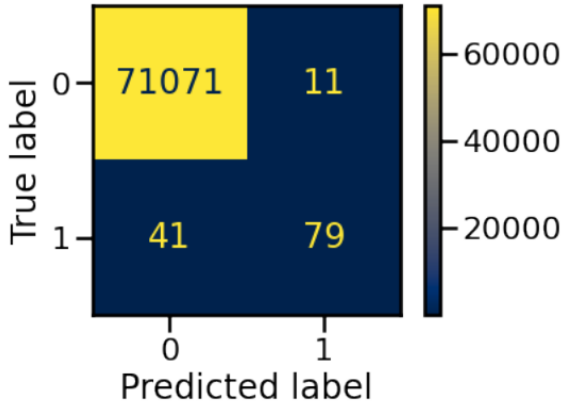

Fig. 1. Logistic Regression Confusion Matrix

D. *Logistics Regression with RandomOverSampler*

To fix the issue of imbalanced dataset, I applied a library, imblearn, to remove any bias towards the minority class. In this case, I used oversampling. Oversampling randomly duplicates examples from the minority class and adds them to the training set. We can see the resulting confusion matrix as shown on Figure 2.

Oversampling has improved in terms of recall score being significantly higher than the precision score. Our resulting values are recall: 90% and precision: 6%. The oversampled model does a better job fitting towards our dataset, but is oversampling on such a large dataset really the correct choice?

It seems that Logistics Regression can do a better job to fitting our data. Next, we will apply under sampling to our Logistics Regression.

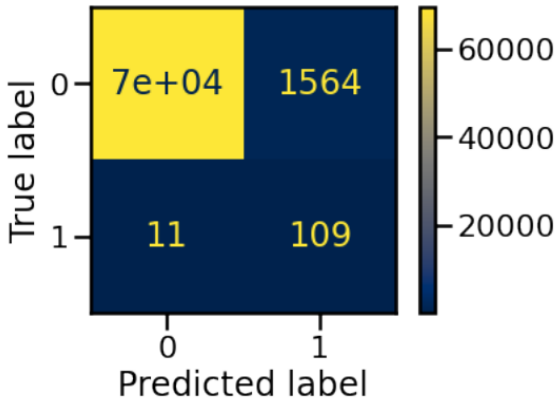

Fig. 2. Over sampled Logistics Regression Confusion Matrix

E. *Logistics Regression with RandomUnderSampler*

Under sampling is technique where it randomly selects examples from the majority class and deletes them from the training set. This technique also comes from the same library as oversampling, imblearn. As I applied under sampled Logistic Regression, you can see the resulting confusion matrix on Figure 3.

This model has done a better job in reducing the number of false negatives than oversampled and normal Logistic Regression. Not only this, but the resulting recall score is significantly higher than the precision score. The resulting values is recall: 92% and precision: 4%.

In the end, it seems that this model best fits our data, however, there is plenty of other techniques to look forward to.

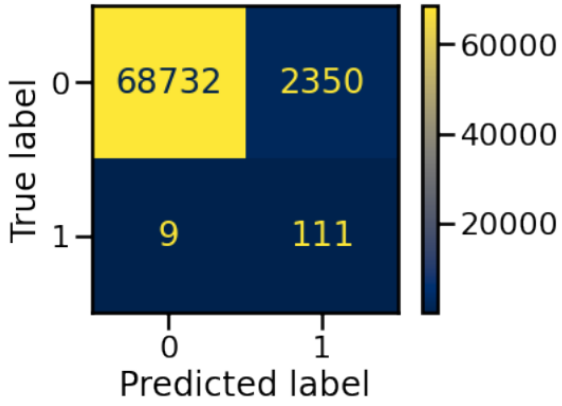

Fig. 3. Under sampled Logistics Regression Confusion Matrix

F. Random Forest

My second algorithm is Random forest which is a supervised learning algorithm that builds multiple decision trees and merges them together to get a more accurate and stable prediction. Following the same steps as Logistic Regression, I trained, fitted, and predicted the training and testing sets with a resulting matrix of Figure 4.

Once again, the imbalanced poses a problem for models that do not resampled. Resampling is when we create a newly transformed training set in which examples have a different class distribution.

Since resampling has not been done here, we are resulting in scores that depict in bias of false positives. The recall score turns out to be lower than the precision score. The calculated values come out to recall: 78% and precision: 94%.

In the next few points, we will talk about the recall and precision value through over sampling and under sampling.

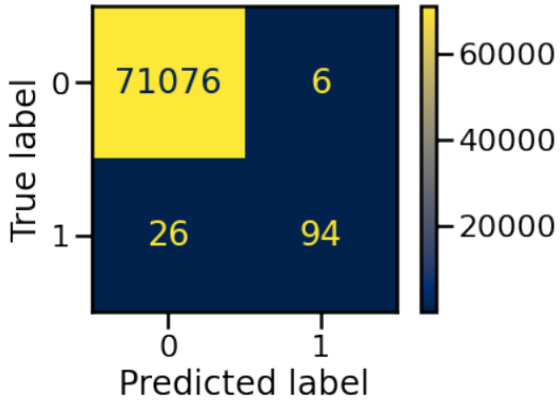

Fig. 4. Random Forest Confusion Matrix

G. Random Forest with RandomOverSampler

The resulting matrix of over sampled Random Forest can be found at Figure 5. Just by deciphering the confusion matrix, we can tell that we have slightly improved our model with just the use of oversampling. With the false negative value being slightly lower means an improvement, however, this is too little to call as efficient.

The resulting recall: 80% is still lower than the resulting precision: 93%. Once again, this has proven that oversampling an extremely imbalanced dataset is not the way to go.

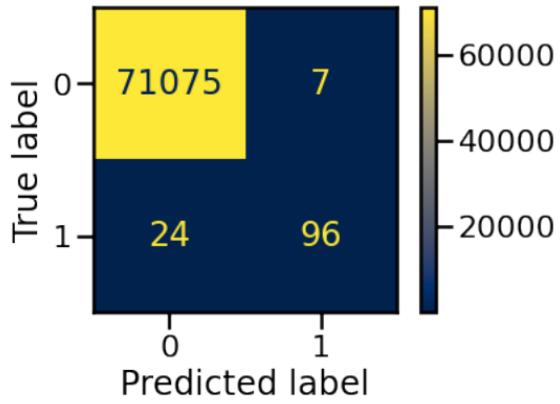

Fig. 5. Over sampled Random Forest Confusion Matrix

H. Random Forest with RandomUnderSampler

The resulting matrix of under sampled Random Forest can be found at Figure 6. By reading the confusion matrix, we can see a much better improvement than was shown with the oversampled Random Forest confusion matrix, Figure 5.

With the work of under sampling, the bias on the dataset has disappeared, and is being evenly distributed. The final recall: 88% and precision: 4% determines that this model is the better choice as it removes the bias on the false positives. Along with the

decrease of false negatives and increase to true negatives. We can say that under sampled Random Forest has done a decent job.

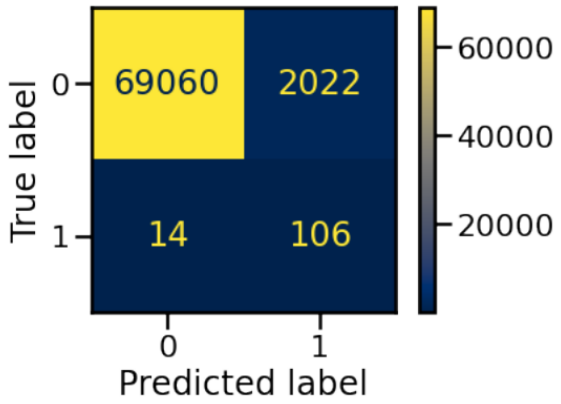

Fig. 6. Under sampled Random Forest Confusion Matrix

I. Neural Network

The third technique used for this classification technique is called Neural Network. This technique uses a series of layers that recognizes relationships in a set of data which mimics that of humans processing data. The layers can differ, where the first layer is known as the input layer. The inner layers are known as hidden layers and the last layer is known as the output layer. These layers interlock, trained, and fitted on.

As I applied Neural Network to the credit card dataset. You can tell that this technique is being trained on imbalanced data as the recall and precision values differ in the way of false positives being biased and false negatives being ignored.

A way to fix this is by using oversampling and under sampling. As shown in Figure 8 and Figure 9. Based on the trend, we know that the under sampled neural network is the best out of neural networks for this dataset. The resulting recall and precision for under sampled is recall: 90% and precision: 5%.

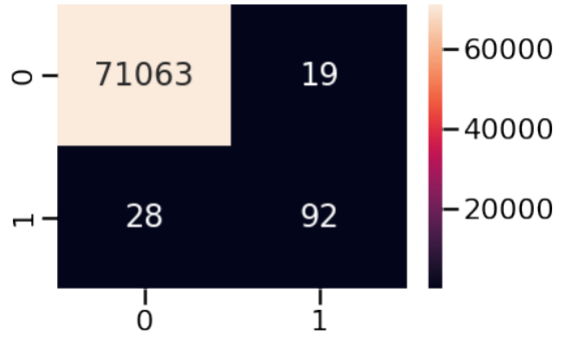

Fig. 7. Neural Network Confusion Matrix

J. Neural Network with RandomOverSampler

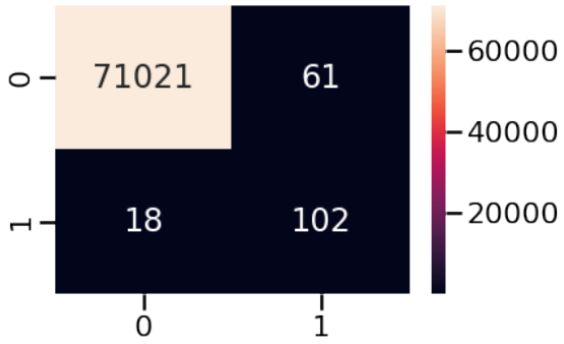

Fig. 8. Over Sampled Neural Network Confusion Matrix

K. Neural Network with RandomUnderSampler

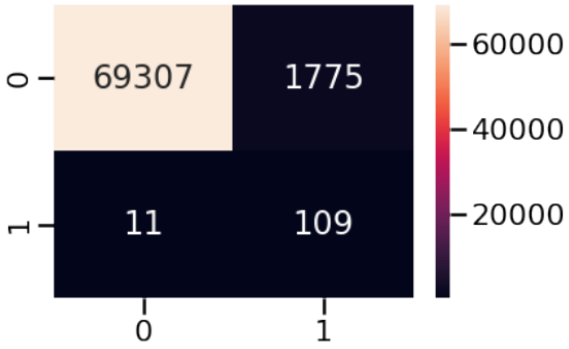

Fig. 9. Under Sampled Neural Network Confusion Matrix

L. Conclusion

Based on the data that was calculated, I can conclude that under sampled Logistics Regression came out the have the best recall score of 92% with the best precision score of 4% for this extremely imbalanced dataset.

	Model	Recall	Precision	F1 Score
0	LogisticsRegression	0.658333	0.877778	0.752381
1	OverSampledLogisticsRegression	0.908333	0.065152	0.121584
2	UnderSampledLogisticsRegression	0.925000	0.045104	0.086013
3	RandomForest	0.783333	0.940000	0.854545
4	OverSampledRandomForest	0.800000	0.932039	0.860987
5	UnderSampledRandomForest	0.883333	0.049812	0.094306
6	NeuralNetwork	0.766667	0.828829	0.796537
7	OverSampledNeuralNetwork	0.850000	0.625767	0.720848
8	UnderSampledNeuralNetwork	0.908333	0.057856	0.108782

Table 1: Credit Card Classification Report

III. REGRESSION DATASET

A. The Dataset

The regression dataset, Concrete Compressive Strength, is a dataset with a dimension of [1030, 8]. This dataset provides a mixture different features, in this case, material that makes the strength of compression better. In this dataset, we will determine the best model that fits the dataset using RMSE and R2 score.

RMSE is the root mean square error which will determine the quality of our model's predictions. The R2 score will determine how good our model fits the dataset. A low RMSE and high R2 score will typically mean that the model fits well.

B. Data Processing

The dataset determines the overall strength of compression, via the "Concrete compressive strength(MPa, megapascals) " column, so *X* will become our data that is not "Concrete compressive strength(MPa, megapascals) " while *y* will become our data that is and only is "Concrete compressive strength(MPa, megapascals) ." Next, I split our *X* and *y* into two subsets: training and testing data. Then, I applied feature scaling which will normalize our training and testing data. With this, we can continue to train our regression techniques.

C. Linear Regression

In Linear Regression is a technique that models the relationship between two variables by fitting a linear equation to the observed data, however, we will be modelling the relationship between all features.

Once the data processing has been completed, I trained, fitted, and predicted on the Linear Regression model. Next, I created a scatter plot that shows our model's prediction over the true prediction values as shown on Figure 10.

According to the figure, we can see that this model is not well fitted to our dataset along with a bad score underfitting RMSE score of 9.79 and a R2 score of 62%. There is no way to improve this model as there is no hyperparameters available to fine tune.

The reason behind this is because Linear Regression is part of the linear model family which usually just models a straight line. I decided to try using another linear model family member that has a few hyperparameters such as Ridge Regression.

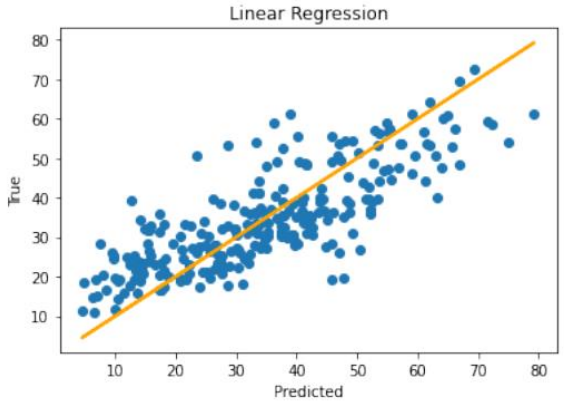

Fig. 10. Linear Regression Scatter Plot

D. Ridge Regression

With Ridge Regression, I tried to apply grid_search_cv for hyperparameter tuning. However, there was only one hyperparameter that was changeable which is alpha. With my attempt to try and get better RMSE and R2 score resulting in having similar results to Linear Regression.

The results were RMSE: 9.79 and R2: 62%. This proves that the linear model family will not be the best fit for the concrete dataset.

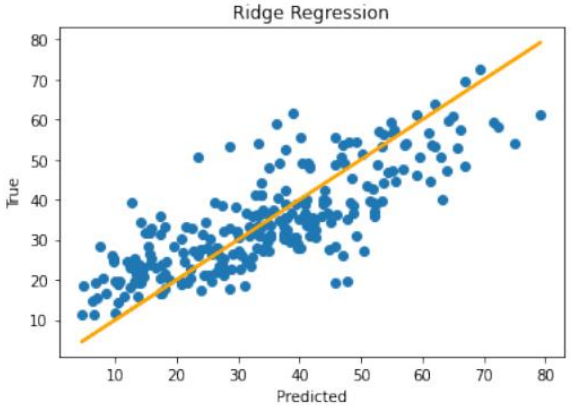

Fig. 11. Ridge Regression Scatter Plot

E. Support Vector Machine

My attempt at SVM turned out to be slightly better than the linear model family. Based on the Figure 12, it may not look like much but there is little improved with no hyperparameter tuning.

The results for a simple SVM came out to be a value of RMSE: 9.47 and an R2 score of 64.7%, however, this slight improvement is not enough to determine to be the better model. Let's try hyperparameter tuning it using grid_search_cv.

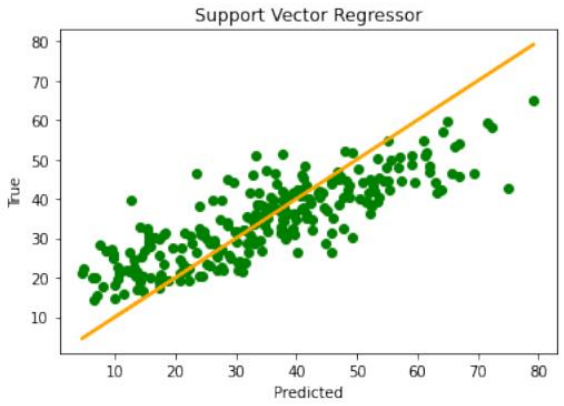

Fig. 12. Support Vector Regressor Scatter Plot

F. Support Vector Machine (Tuned)

In this SVM, I applied grid_search_cv and got the best parameters to best fit this dataset. The specific parameters came out to be an "rbf" SVM with a C value of 9, gamma set to auto, and epsilon value of 0.1.

Based on Figure 13, we can see a definite improvement, a large one at that. The resulting values turned out to be RMSE: 6.96 with an R2 score: 80.9%. This has shown that this model is well fitted to the dataset and may be the winner of our techniques used.

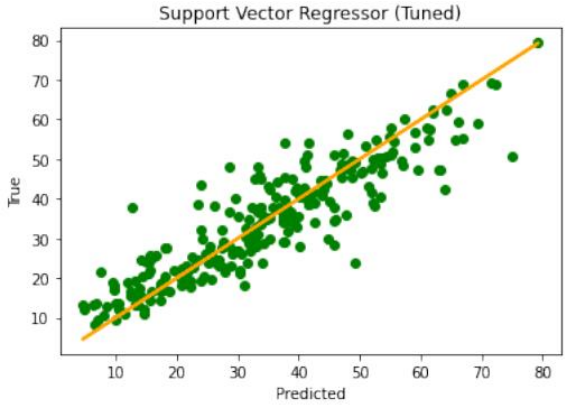

Fig. 13. Support Vector Regressor Scatter Plot (Tuned)

G. Neural Network

In neural network, I have not applied grid_search_cv, but instead applied batch normalization, so that we can get better results each batch run. This has definitely improved the score as I was achieve usually high values of RMSE and R2, without it.

According to figure 14, we can see that this model was able to appropriately fit the model well enough to not be considered underfitted or overfitted.

The resulting RMSE is 5.79 with an R2 score of 86.8%

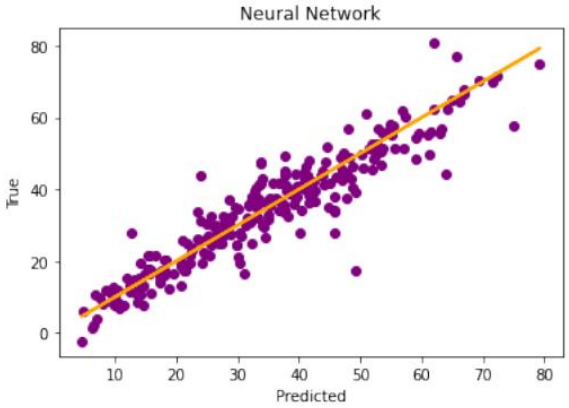

Fig. 14. Neural Network Scatter Plot

H. Conclusion

To conclude this regression problem, I can say that using neural network is our finalist with a RMSE score of 5.79 with an R2 score of 86.8%. These values are not the best, but it is our overall best.

Even though Neural network was our best, I can still say that the hyperparameter tuned SVM is still a good model that fits the dataset well.

Overall, using different techniques can achieve the best model that is not underfitted as well as not overfitted, but good enough.

	Model	RMSE	R2-Score
0	LinearRegression	9.793066	0.623414
1	RidgeRegression	9.788387	0.623774
2	SupportVectorRegression	9.477665	0.647281
3	TunedSupportVectorRegression	6.966604	0.809424
4	NeuralNetwork	5.882659	0.864114

Table 2: Credit Card Classification Report

IV. CONCLUSION

I would like to say that there is a lot of techniques for all kinds of different problems. These techniques can be used in different way to improve our scores. There are endless choices in machine learning!