|  |
| --- |
| **Machine Learning Report** |

**Habeebuddin Mir**

**NEU ID: 002713929**

College of Engineering

Northeastern University

Toronto, ON

*mir.h@northeastern.edu*

**Abstract**

In this report, I evaluated the performance of supervised learning algorithms on two widely popular datasets. The five learning algorithms include: Decision trees, Neural networks, Boosting, Support Vector Machines and k-nearest neighbors. The two datasets are Credit Card Fraud Detection and Titanic Survival. The learning curve and model complexity of each algorithm on both datasets have been explored and analyzed.

**1 Datasets**

I used the legendary “Credit Card fraud detection” and the “Titanic” datasets hosted on kaggle.com

Table 1: The basic feature of both datasets.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Data Set Characteristics | Attribute Characteristics | Associated Tasks | Number of Instances | Number of Attributes |
| Credit Card Fraud Detection | Univariate | Numerical | Classification | 28407 | 30 |
| Titanic | Multivariate | Mixed | Classification | 1309 | 12 |

**1.1 Data characteristics**

The histograms of classes from both datasets are shown in Figure 1. Class unbalance is evident in both datasets and must be accounted for in calculating the accuracy scoring function. Weighted scoring function is thus used in my analysis.

Chart

Description automatically generated

Figure 1: The class frequency of.

**1.2 Why are these interesting datasets?**

Both datasets are interesting for primarily learning and secondary on the wealth of unlocked knowledge that can be accessed by building machine learning models to understand find meaningful conclusions.

Credit Card fraud detection is a tricky problem to solve, and the anonymity of the data imbalance is what makes it interesting to work upon. On the other hand, the Titanic data, unlike Credit card data has understandable features and multivariance data attributes.

**2 Decision Tree with post-pruning**

I chose post-pruning by traversing the tree and removing all children of the nodes with maximum class-count greater than 50.

Following are the learning curves I got after training my model using DecisionTreeClassifier:

Chart

Description automatically generatedChart, line chart

Description automatically generated

The left side figure is for Titanic Survival and the right one represents the Credit Card fraud detection.

The accuracy score are as follows:

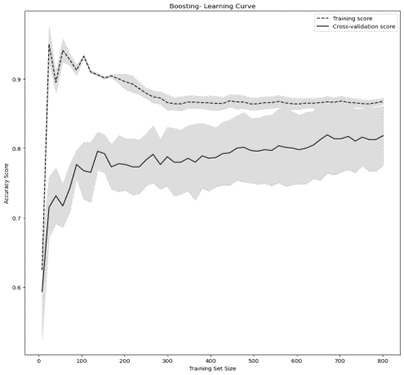
Titanic - 86.76%

Credit Card Fraud Detection - (train): 99.67%, (test): 99.77%, (positive test): 0 of 64 (0.0%)

**3 Boosting**

For Titanic I used XG Boosting and for Credit Card I used AdaBoost as the data is more complex and higher dimensional for Titanic and low noisiness with Credit Card fraud detection.

Following are the learning curves I got after training my model using Boosting:

Chart, line chart

Description automatically generated

The left side figure is for Titanic Survival and the right one represents the Credit Card fraud detection.

The accuracy score are as follows:

Titanic - 86.53%

Credit Card Fraud Detection - (train): 100%, (test): 99.92%, (positive test): 44 of 64 (68.75%)

**4 Neural Networks**

For Titanic, I first shortlisted hyperparameters to test and build using MLPClassifier. The various models I built with different hyperparameters had the following conclusions ranked:

Model with rank: 1

Mean validation score: 0.811 (std: 0.022)

Parameters: {'learning\_rate\_init': 0.003305584892205408, 'hidden\_layer\_sizes': (17, 6, 8, 17), 'alpha': 1e-06, 'activation': 'tanh'}

Model with rank: 2

Mean validation score: 0.673 (std: 0.098)

Parameters: {'learning\_rate\_init': 0.043637230410751945, 'hidden\_layer\_sizes': (12, 8, 15, 10, 6, 6, 16, 17, 10, 10, 9, 17, 6), 'alpha': 1e-05, 'activation': 'relu'}

Model with rank: 3

Mean validation score: 0.616 (std: 0.002)

Parameters: {'learning\_rate\_init': 0.09248902358138807, 'hidden\_layer\_sizes': (15, 18, 6, 16, 17, 9, 14, 9, 5, 8, 10, 9, 9), 'alpha': 6e-06, 'activation': 'relu'}

For Credit Card Dataset I used MLPClassifier with the hyperparameter solver value set to “lbgfs” which is an optimizer in the family of quasi-Newton methods. It works well with smaller datasets by converging faster and performing better.

Following are the learning curves I got after training my model using MLPClassifier:

Chart

Description automatically generated Chart, line chart

Description automatically generated

The left side figure is for Titanic Survival and the right one represents the Credit Card fraud detection.

The accuracy score are as follows:

Titanic - 84.06%

Credit Card Fraud Detection - (train): 99.72%, (test): 99.88%, (positive test): 32 of 64 (50.0%)

**5 k- Nearest Neighbors**

As knn classifies based on the distance between points, the scale of the variables matters and any variables that are on a large scale will have a much larger effect on the distance between observations. Hence, I used scikit-learn’s StandardScaler to standardise feature columns by centering and scaling. Now the data is ready for the algorithm. First, we split the data into training and testing sets using sklearn’s train\_test\_split. Next, we create a knn model using sklearn’s KNeighborsClassifier with n-neighbor hyperparameter set to 1 initially, which is the simplest form of knn considering 1 nearest neighbor. As the data consists of stanndarised real-valued numerical points, the default eucledian distance can be used. Finally, we fit the model into our training data and then predict the testing data points. To evaluate how well our model generalises and to compare actual and predicted target values, we generate the confusion matrix and classification reports

I now use the “elbow method” to find the best value for the number of neighbors parameter that gives us the most precision. To do that, we create a for loop that trains various knn models with different neighbor values, then keep track of the error\_rate for each of these models with a list.

Following are the learning curves I got after training my model using MLPClassifier:

A picture containing diagram

Description automatically generated

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

The top figure is for Titanic Survival and the bottom two represents the Credit Card fraud detection.

The accuracy score are as follows:

Titanic – 83.84%

Credit Card Fraud Detection –

KNN(3) (train): 99.94030898876404%

KNN(3) (test): 99.8700888311506%

KNN(3) (positive test): 32 of 64 (50.0%)

KNN(5) (train): 99.9122191011236%

KNN(5) (test): 99.8771110564938%

KNN(5) (positive test): 34 of 64 (53.125%)

**6 Support Vector Machines**

Support vector machines are highly sensitive to scale of features; hence I started off by using StandardScaler to standardize the dataset. Next I explore the SVC model by changing the kernel values from thr rbf default to sigmoid to understand the impact and changes observed.

Following are the learning curves I got after training my model using KNeighborsClassifier:  
A picture containing chart

Description automatically generated

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

The top figure is for Titanic Survival and the bottom two represents the Credit Card fraud detection.

The accuracy score are as follows:

Titanic - 78.23%

Credit Card Fraud Detection –

SVM (train): 99.91924157303372%

SVM (test): 99.8735999438222%

SVM (positive test): 32 of 64 (50.0%)

SVM (rbf) (train): 99.91924157303372%

SVM (rbf) (test): 99.8735999438222%

SVM (rbf) (positive test): 32 of 64 (50.0%)

**7 Conclusions**

In this report, I have analyzed the performance of 5 supervised learning algorithms on two highly popular Kaggle datasets and explored how different algorithms act with different types of datasets and how choosing the optimized hyperparameters can fir the model better.