CS7641 Assignment 1  
Supervised Learning  
Manar Saiyad

Fall 2024

Code reference: <https://github.com/saiyadma/Supervised-Learning-.git>

# Classification Problems

Given the endless data pool available on the internet, It was very difficult to pick classification problems that are simple yet nontrivial. To keep the project interesting, I decided to pick two classification problems that are both interesting to me personally and will give me great insight into how machine learning works in different contexts. The first Classification problem I picked was the MBA admissions classification. This Classification problem fascinates me as it aligns very well with my current occupation as a master's student. I believe it will give me good insights on what entails the admissions process of a master's student, and how these decision processes can be further automated. The other classification problem I picked was the Loan Qualification Classification. This classification problem is particularly interesting to me as it is very similar to my current occupation. As an underwriter, I work very closely with loans and am very interested in fintech. Through this project, I am excited to learn about the classifiers and performance of a loan approval model. These two classification problems and their inherently different natures will allow me to understand classification models from different points of view, as well as they will lay a great base for comparison as one of them is a binary classification problem and the other is not.

# Hypothesis

The Hypothesis I have for the loan classification function is that the loan classification function will perform very well under RBF or polynomial kernels because they are high dimensional nonlinear features.

For the MBA classification Problem, I believe KNN will have the best performance because the data does not have many outliers and extras that will hinder the process of classification. This hypothesis is based on the fact the correlation matrix below which shows that closely corelated dataset suggesting little to outliers.

A screenshot of a graph

Description automatically generated

# Training performance and Learning Curve

## Neural Network

To test the Learning curve for the classification problems, I analyzed the datasets in and trained and tested them using sklearn’s train test split. I split the data from the range 0 to 100 and tested the data set over each accuracies. Bellow is the plot to show the performances. For neural Networks, the learning performance was just as expected.

A graph of a train size

Description automatically generatedA graph of a train size

Description automatically generated

## *Support Vector Machines*

A graph with blue lines

Description automatically generatedA graph showing the growth of a train

Description automatically generated

The SVM learning performance was very well for the MBA data set suggesting that SVM was well-fitted for this type of classification problem.

## *KNN*

A graph showing a train size

Description automatically generatedA graph of a train size

Description automatically generated

As for KNN the learning did not fluctuate for either of the datasets.

# Hyperparameter Tuning and model complexity graph

Neural Network :

In order to perform, hyperparameter tuning I tested various epoch values**.** This hyperparameter represents the number of times the entire training dataset is passed through the model during training. Increasing the number of epochs can improve the model’s performance but may lead to overfitting if not done carefully. In scikit-learn, you can control the number of epochs (iterations over the entire dataset) for neural networks using the max\_iter parameter. Below is the trend performance for both classification problems over different epochs, They provided very similarly. They both suggest that the ideal max\_iter value is around 70 or 80 and this value is the max accuracy and the accuracy settles after that.

A graph with a line

Description automatically generatedA graph with a line

Description automatically generated

Another Hyperparameter I tested was trying different activation Functions. I tested the tanh and relu activation functions. The accuracy for tanh activation function on loan classification was 0.70, and relu was 0.69. For MBA classification function the tanh activation function accuracy was 0.85, and the relu function accuracy was 0.86. Both of these have very similar accuracies. Suggesting that the data performed verry similarly under either function.

Support Vector Machines :

The three kernel functions I tested in SVMs are Relu, Sigmoid, and Poly. Both the classification functions had very similar performances again. One noteworthy observation made was tat Poly kernels took more than average time comparatively. For loan classification problem, the polly kernel had rough performance. I believe a polynomial kerner is not the ideal fit for these caslsification problems.

|  |  |  |
| --- | --- | --- |
|  | Loan | MBA |
| Relu | A graph with blue lines  Description automatically generated |  |
| Sigmoid | A graph with blue lines  Description automatically generated |  |
| poly | A graph with blue lines  Description automatically generated |  |
| Gama values | A graph with a line  Description automatically generated |  |
| C values | A graph with a line  Description automatically generated |  |

I also tested various C and Gama values, However there was little to no affect of this hyperparameters tuning on either classification problems.

KNN:

The K values I tested ranged from 0 to 1, as the plots below show that both of these classification functions had a positive correlation on performance from increasing k values.

A graph with blue lines

Description automatically generatedA graph of a train size

Description automatically generated

Another hyperparameter I tuned was the weights metric, I used distance and uniform. AS the plots below suggest, the distance matrix did not work very well for the loan data set while it worked very well for MBA.

A graph of a train size

Description automatically generatedA graph with blue lines

Description automatically generated

The next matrix I tried for weights was uniform and as the plots suggest the performances were similar for both of them. Both classifications settled down verry well for these weight matrix.

A graph of a train size

Description automatically generatedA graph of a train size

Description automatically generated

## Results and Analysis

From the previous section of the report, it can be pulled that KNN was a classification function that did the best with 86 percent overall accuracy and relu SVM was not far behind with 84 % accuracy. As for the loan classification, SVM poly kernel had the best performance with 78% accuracy. A Simple neural network was not a great fit for the classification function. However, it is important to note that tuning the hyperparameters changed the accuracy drastically. To improve the accuracies further, a combination of tuned hyperparameters can and should be used to improve performance. All the models were fairly qui kinterms of the runtime except the poly kernel for SVM. Which was notably longer. This might suggest the loan dataset does not have polynomial trends and characteristics. The improved performance with increasing iterations or tuning the data set suggests that there is some complexity in the data sets and the data performance was a direct result of that. Additionally, the similarity in the performance of the two classification problems is also a result of similarity in the classification problems itself.

## Citation:

<https://www.kaggle.com/datasets/taweilo/mba-admission-dataset>

<https://pandas.pydata.org/docs/reference/api/pandas.to_numeric.html>

<https://stackoverflow.com/questions/29432629/plot-correlation-matrix-using-pandas>

<https://www.interviewquery.com/p/classification-projects>

<https://www.kaggle.com/code/rishikachaudhary/loan-prediction>

<https://rubikscode.net/2021/07/19/top-23-best-public-datasets-for-practicing-machine-learning/#tsa>