[CMPUT466/566] Course Project Report

Project Introduction

In this project, our objective is to categorize the balance scale dataset into three distinct classes, namely “L” (left), “R” (right) and “B” (balance).The three algorithms we have chosen to classify the balance scale dataset are KNN, SVM, and Logistic Regression with SGD. To optimize the performance of each model, we systematically tuned their hyperparameters. For the KNN classifier, the primary hyperparameter of interest was K(neighbors), we experimented with various values of k and applied K-fold cross-validation to determine the optimist value that resulted in the highest accuracy. In the case of Logistic Regression with the SGD model, we explored different learning rates, regularization terms (L1, L2 and Elastic Net) and a number of epochs. Using cross-validation, we determined the optimal set of hyperparameters that lead to the highest performance. Lastly, for the SVM classifier, we focused on the regularization parameter (C) and maximized the model’s performance using cross-validation accuracy. Ultimately, we compared the performance of each model when configured their hyperparameters. To visualize the performance of each model during the hyperparameter tuning process, we plotted the validation accuracy as a function of the corresponding hyperparameters.

Dataset Introduction

The Balance Scale dataset is a multiclass classification problem, where the goal is to predict the balance scale tip direction (left, balanced, or right) based on the weights and distances of objects placed on the scale. It consists of data related to the balance scale of a seesaw, with the goal of predicting the direction in which the scale will tip based on the weight

Balance Scale Data Set

Download: Data Folder, Data Set Description

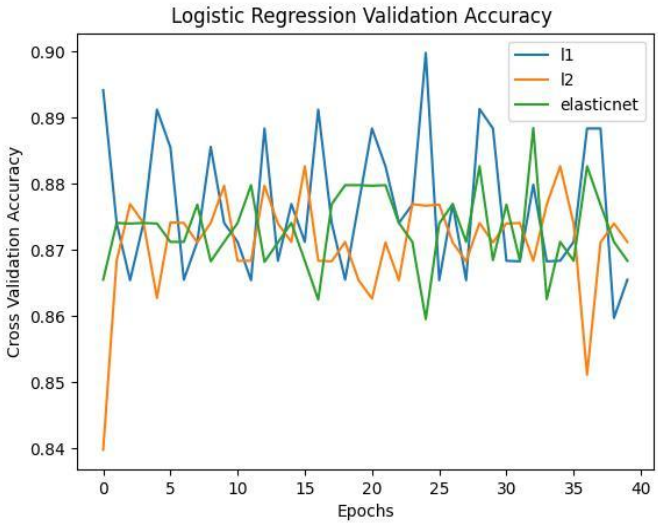
Abstract: Balance scale weight & distance database

Data Set Characteristics:	Multivariate	**Number of Instances:**	625	**Area:**	Social
Attribute Characteristics:	Categorical	**Number of Attributes:**	4	**Date Donated**	1994-04-22
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	336468

and distance of the objects placed on it. The dataset contains three classes which are ‘L’(left), ‘R’(right) and ‘B’(balance).

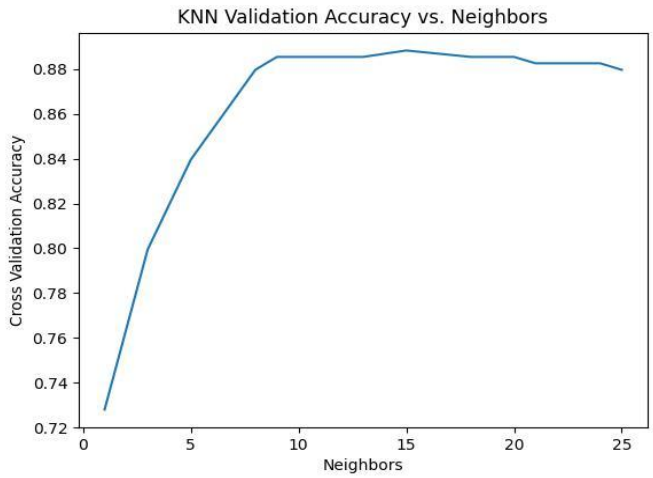
Logistic Regression with SGD

Logistic Regression is a widely used statistical model for binary and multi-class classification (softmax), we employed Logistic Regression with Stochastic Gradient Descent (SGD) as an optimization technique to fit the model. To implement logistic regression with SGD, we utilized the ***`SGDClassifier`*** class from the Scikit-learn library. We specified the ***log*** loss function to the model logistic regression and experimented with different hyperparameters such as learning rate, regularization term (L1, L2, Elastic net), and the number of iterations. We selected the candidate learning rates as alphas = [0.0001, 0.001, 0.01, 0.1] and set up the maximum number of epochs. For each regularization term, we followed the systematic approach to identify the optimal learning rate. First, we fixed the regularization term and then iterated through each learning rate in the alphas list. For every combination of the regularization term and learning rate, we trained the Logistic Regression with ***`SGDClassifier`*** and evaluated its performance on the validation dataset. We compared L1, L2 and Elastic Net with their respective learning rates and found that L1 is the best regularization term with alpha =0.0001. Upon executing the program with the optimal hyperparameters (**L1 regularization** and a **learning rate of 0.0001**), we obtained the highest test accuracy for our Logistic Regression with SGD model, which was **89.36%** (0.8936 when rounded to four decimal places).



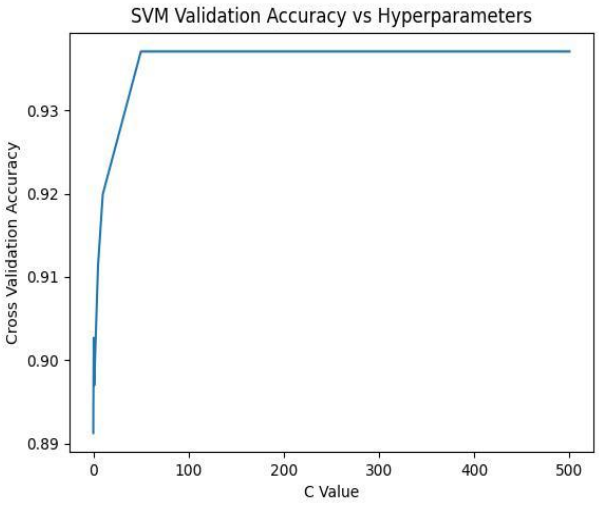
KNN

KNN is a popular instance-based learning method that relies on the concept of distance metrics to identify the most similar training instances for a given test instance, ultimately determining its class label based on a majority vote among its k nearest neighbors.To ensure the best performance of our KNN classifier, we carried out a systematic hyperparameter tuning process, focusing on the key hyperparameter k (the number of neighbors). To determine the optimal value for k, we employed k-fold cross-validation, which is a robust technique for model evaluation that involves dividing the dataset into k equally-sized folds. This method allows for the training and validation of the model on different subsets of the data, ultimately providing a more reliable estimate of the model's performance. When selecting candidate values for the number of neighbors in KNN, it is generally advisable to choose odd numbers in order to minimize the likelihood of ties during the majority voting process. As observed in the plot on the right, the performance experiences a slight decline when an even number of neighbors is used. However, the impact on the overall performance is not substantial, likely due to the fact that the balance scale dataset has an odd number of classes (3). This suggests that the choice of neighbors has a limited influence on the classification outcome in this specific case. Upon executing the program with the optimal hyperparameter setting, we determined that the best number of neighbors for our KNN classifier was **15**. With this configuration, the classifier achieved the highest test accuracy of **88.83%** (0.8883 when rounded to four decimal places) on the balance scale dataset.



SVM

SVM is a powerful and widely used machine learning method that seeks to find the maximum margin hyperplane separating different classes in the feature space. This approach results in a robust classifier that can generalize well to unseen data, and it can handle both linearly separable and non-linearly separable problems with the use of kernel functions. To optimize the performance of our SVM classifier, we carried out a systematic hyperparameter tuning process, focusing on the key hyperparameter C (the regularization parameter). The C parameter controls the trade-off between maximizing the margin and minimizing the classification error. A smaller value of C results in a larger margin but allows some misclassifications, while a larger value of C aims for a smaller margin with fewer or no misclassifications. During the hyperparameter tuning process for the SVM classifier, we utilized k-fold validation to estimate the validation accuracy for each candidate value of the regularization parameter C. After determining the optimal value for C, we retrained the SVM classifier (using the SVC function) with the best C value on the entire training dataset. We then evaluated the performance of the classifier on the test dataset by calculating the test accuracy using the accuracy_score function. Upon running the program with the optimal hyperparameter setting, we determined that the best value for the regularization parameter C in our SVM classifier was 50. With this configuration, the classifier achieved a high test accuracy of **95.21%** (0.9521 when rounded to four decimal places) on the balance scale dataset.



Conclusion

In this project, we employed three different classification algorithms to categorize the balance scale dataset: Logistic Regression with Stochastic Gradient Descent (SGD), Support Vector Machine (SVM), and K-Nearest Neighbors (KNN). Our findings demonstrate that all three algorithms were capable of achieving a high level of accuracy at around 90%. We systematically tuned hyperparameters for each classifier and assessed their performance on the test dataset using accuracy as the primary evaluation metric. Our findings indicate that the SVM classifier outperformed the other two algorithms, achieving the highest test accuracy of 95.21% (0.9521). In comparison, Logistic Regression with SGD achieved a test accuracy of 89.36% (0.8936), while KNN reached an accuracy of 88.83% (0.8883). The superior performance of the SVM classifier can be attributed to its ability to find the maximum margin hyperplane that best separates the classes in the feature space. Additionally, SVM is robust against overfitting, particularly when using appropriate kernel functions and tuning hyperparameters effectively.