Data set 1 sign language mnist

introduction

The Sign Language MNIST dataset is a modern reinterpretation of the classic MNIST dataset, focusing on American Sign Language hand gestures. It contains 28x28 grayscale images of hands forming 24 different letters (excluding J and Z) stored in CSV format. The dataset was created by processing original images—cropping to the hand region, converting to grayscale, resizing, and applying various augmentations like filters, pixelation, brightness/contrast adjustments, and slight rotations. This dataset not only serves as a challenging benchmark for computer vision and machine learning methods but also supports the development of practical applications for enhancing communication for the deaf and hard-of-hearing.

Problem statement

The goal is to predict the correct label for each image—that is, to determine which sign the image represents.

The code takes each 28x28 grayscale image and, using a machine learning or deep learning model, outputs a predicted sign category.

The goal is to translate visual information (images of hand signs) into meaningful, discrete labels . This prediction process is key for converting sign language into text or speech.

Table 1. Performance Metrics for Sign Language MNIST

algorithm	train_accuracy	test_accuracy	precision_score	recall_score	f1_score	roc_auc_score
K Nearest Neighbors	1	0.822922	0.840927	0.822922	0.824225	0.905546
Decision Tree Classifier	1	0.479225	0.498604	0.479225	0.484004	0.725104
Support Vector Machine	1	0.848857	0.861457	0.848857	0.851152	0.991216
Neural Network	1	0.822922	0.833786	0.822922	0.824371	0.987339
Scratch Logistic Regression	0.360444	0.391662	0.321133	0.324965	0.321071	0.968618
Scratch Neural Network	0.360663	0.785276	0.769100	0.773277	0.762408	0.983595

Output explanation

1. Support Vector Machine (SVM)

Test Accuracy: ~84.9%

F1 Score: ~85.1% ROC

AUC: ~0.9912 Why It's

Best:

The SVM not only achieved the highest test accuracy but also maintains excellent precision, recall, and an outstanding ROC AUC. These indicators show that it generalizes very well and distinguishes between classes robustly.

2. Neural Network (Built-in Version)

Test Accuracy: ~82.3%

F1 Score: ~82.4%

ROC AUC: ~0.9873 Why

It's Second:

This neural network model performs comparably well on test accuracy to KNN, but with a notably higher ROC AUC. This suggests that it ranks the classes more effectively despite both having perfect training accuracy.

3. K Nearest Neighbors (KNN)

Test Accuracy: ~82.3%

F1 Score: ~82.4% ROC

AUC: ~0.9055

Why It's Third:

KNN matches the test accuracy of the neural network; however, its ROC AUC is notably lower, indicating a less robust ability to separate the classes across various thresholds.

4. Scratch Neural Network

Test Accuracy: ~78.5%

F1 Score: ~76.2%

ROC AUC: ~0.9836 Why

It's Fourth:

While the scratch neural network offers decent performance, its test accuracy and overall metrics fall short compared to the built-in neural network. It does maintain a strong ROC AUC, but the lower accuracy places it behind the top three.

5. Decision Tree Classifier

Test Accuracy: ~47.9%

F1 Score: ~48.4% ROC

AUC: ~0.7251

Why It's Fifth:

Despite achieving a perfect training accuracy (which indicates overfitting), the decision tree dramatically underperforms on test data. The drop in performance indicates poor generalization for this dataset.

6. Scratch Logistic Regression

Train Accuracy: ~36.0%

Precision, Recall, F1: All in the low 0.32–0.39 range

ROC AUC: ~0.3211 Why

It's Worst:

This model shows the lowest performance across almost all metrics. The very low accuracy and associated metrics suggest that the model is underfitting (or may not have been tuned effectively), making it the least effective algorithm for this task.

Data set 2 League of legends classification

introduction

League of legends is a very popular moba game and the dataset is based on that game so This dataset comprises detailed information from ranked League of Legends matches played between January 12, 2023, and May 18, 2023, covering a broad range of skill levels from Iron to Diamond. The data is organized by game time intervals recorded at 20%, 40%, 60%, 80%, and 100% of the match capturing various in-game statistics such as player performance, objective control, and gold distribution. This structure provides valuable insights into how matches evolve over time, making it an excellent resource for analyzing player behavior, game strategy, and trends in competitive play.

Problem statement

Develop a model to predict whether the red team wins a League of Legends match based on game statistics. By using all available features except for the "redWin" column, our goal is to forecast the match outcome and gain insights into the key factors driving team performance.

Table 2 . Performance Metrics for league classification

algorithm	train_accuracy	test_accuracy	precision_score	recall_score	f1_score	roc_auc_score
K Nearest Neighbors	0.794682	0.745151	0.745245	0.745151	0.745150	0.745197
Decision Tree Classifier	0.989490	0.702197	0.702247	0.702197	0.702135	0.702096
Support Vector Machine	0.725277	0.727849	0.728567	0.727849	0.727535	0.727605
Neural Network	0.761463	0.762175	0.769601	0.762175	0.760736	0.762775
Scratch Logistic Regression	0.726180	0.728639	0.729475	0.728391	0.728240	0.821989
Scratch Neural Network	0.732775	0.734959	0.735781	0.734718	0.734585	0.830587
Scratch Gaussian NB	0.007208	0.007235	0.725240	0.723161	0.722763	0.813407

Output explanation

Neural Network (Built-in)

Test Accuracy: 0.762175 (highest of all models)

F1 Score: 0.760736

ROC AUC: 0.762775

This built-in NN achieves the top accuracy and a solid F1 Score, making it the best overall performer if raw accuracy is your main goal.

K Nearest Neighbors (KNN)

Test Accuracy: 0.745151

F1 Score: 0.745150

ROC AUC: 0.745197

KNN is second in accuracy, with metrics that closely match its test accuracy. It generalizes well without major overfitting (train accuracy is higher but not perfect).

Scratch Neural Network

Test Accuracy: 0.734959

F1 Score: 0.734585

ROC AUC: 0.830587 (noticeably high)

While its raw accuracy is slightly below KNN, the very high ROC AUC suggests it discriminates classes quite well across different thresholds. Still, strictly by accuracy, it lands in third place.

Scratch Logistic Regression

Test Accuracy: 0.728639

F1 Score: 0.728240

ROC AUC: 0.821989

Another strong ROC AUC, slightly lower accuracy than the Scratch NN. It's still quite competitive, falling just behind the Scratch NN based on test accuracy and F1 Score.

Support Vector Machine (SVM)

Test Accuracy: 0.727849

F1 Score: 0.727535

ROC AUC: 0.727605

SVM is close behind the Scratch Logistic Regression in all-around metrics but slightly lower in test accuracy and F1, so it ranks fifth.

Decision Tree Classifier

Test Accuracy: 0.702197

F1 Score: ~0.702197

ROC AUC: 0.702096

With very high training accuracy (0.989490) but a steeper drop on the test set, the tree clearly overfits. It occupies the lower end of the performance spectrum here.

Scratch Gaussian NB

Test Accuracy: 0.007235

F1 Score: 0.722763 (likely reflecting extreme class imbalance or a metric-logging bug)

ROC AUC: 0.813407

The near-zero accuracy indicates that the model (or the data/labels) are severely mismatched. Despite the high precision/recall/f1 metrics, the minuscule accuracy makes it the weakest in conventional terms.

Data set 3 telecom regression

introduction

This dataset contains information on 3150 telecom customers collected over 12 months by an Iranian telecom company. It includes various details such as call failures, SMS frequency, complaints, subscription length, age group, charge amounts, service type, seconds of use, customer status, and usage frequency all summarized for the first nine months. The main goal is to use these features to predict the overall value of each customer at the end of the 12-month period.

Problem statement

We aim to predict the overall customer value at the end of a 12-month period using aggregated behavioral data from the first nine months. This prediction helps the telecom company identify which customers are likely to be the most valuable.

Table 3 . Performance Metrics for telecom regression

algorithm	r2_train	r2_test	rmse	mae
K Nearest Neighbors	0.996394	0.993869	39.247698	26.079934
Decision Tree Regressor	1	0.996784	28.425286	14.635127
Support Vector Machine	0.980556	0.974180	80.541856	35.463996

Neural Network	0.989677	0.984951	61.489438	32.346446
Linear Regression (Scratch)	0.758674	0.757642	246.757225	181.125528
Random Forest (Scratch)	0.819511	0.701682	273.766914	197.015700

Output explanation

Decision Tree Regressor

R^2 (Test): 0.996784 — the highest among all models

RMSE: 28.425286 — the smallest, indicating minimal deviation

MAE: 14.635127 — the lowest overall

Observations: Though it achieves a perfect R^2 of 1.0 on training (indicating possible overfitting), it still performs best on the test set, making it the top choice by the numbers provided.

K Nearest Neighbors

R^2 (Test): 0.993869 — very close to the Decision Tree's result

RMSE: 39.247698

MAE: 26.079934

Observations: Delivers excellent performance, though slightly behind the Decision Tree in R^2, RMSE, and MAE.

Neural Network

R^2 (Test): 0.984951

RMSE: 61.489438

MAE: 32.346446

Observations: Solid results with relatively high R^2, but larger errors compared to the top two.

Support Vector Machine

R^2 (Test): 0.974180

RMSE: 80.541856

MAE: 35.463996

Observations: Also a decent model, yet it lags behind the Neural Network based on lower R^2 and higher error metrics.

Linear Regression (Scratch)

R^2 (Test): 0.757642

RMSE: 246.757225 MAE:

181.125528

Observations: Significantly larger errors than the top four, reflecting a lower fit on the test set.

Random Forest (Scratch)

R^2 (Test): 0.701682 — the lowest R^2 among the models listed

RMSE: 273.766914

MAE: 197.015700

Observations: Has the weakest performance in terms of predictive accuracy and error metrics.

Reference list

datacamp (n.d.). *Python Machine Learning: Scikit-Learn Tutorial*. [online] www.datacamp.com. Available at: https://www.datacamp.com/tutorial/machine-learningpython.

egazakharenko (2024). *All popular ML-algorithms from scratch in Python* . [online] Kaggle.com. Available at: https://www.kaggle.com/code/egazakharenko/all-popular-mlalgorithms-from-scratch-in-python

freeCodeCamp.org (2022). *Machine Learning for Everybody – Full Course*. *YouTube*. Available at: https://www.youtube.com/watch?v=i_LwzRVP7bg

freeCodeCamp.org (n.d.). *Machine Learning with Python and Scikit-Learn – Full Course*. [online] www.youtube.com. Available at: https://www.youtube.com/watch?v=hDKCxebp88A

Infinite Codes (2024). *All Machine Learning algorithms explained in 17 min*. [online] YouTube. Available at: https://www.youtube.com/watch?v=E0Hmnixke2g

Müller, A.C. and Guido, S. (2017). *Introduction to machine learning with Python : a guide for data scientists*. Beijing: O'reilly.

Patrick Loeber (2019a). *Logistic Regression in Python - Machine Learning From Scratch 03 - Python Tutorial*. [online] YouTube. Available at: https://www.youtube.com/watch?v=JDU3AzH3WKg&list=PLqnslRFeH2Upcrywfu2etjdxxkL8nl7E&index=3

Patrick Loeber (2019b). *Naive Bayes in Python - Machine Learning From Scratch 05 - Python Tutorial*. [online] YouTube. Available at: https://www.youtube.com/watch?v=BqUmKsfSWho&list=PLqnslRFeH2Upcrywfu2etjdxxkL8nl7E&index=5

Patrick Loeber (2019c). *PCA (Principal Component Analysis) in Python - Machine Learning From Scratch 11 - Python Tutorial*. [online] YouTube. Available at: https://www.youtube.com/watch?v=52d7ha-GdV8&list=PLqnslRFeH2Upcrywfu2etjdxxkL8nl7E&index=11

Preprocessing for Machine Learning in Python. [online] DataCamp. Available at: https://app.datacamp.com/learn/courses/preprocessing-for-machine-learning-in-python