Study On Pen-based recognition of handwritten digits

Student Name: RAJ JAISWAL

Enrolment Number: 03619011921

Signature:

Email ID: raj.03619011921@ipu.ac.in

Contact Number: 8750574965

Report:

https://drive.google.com/file/d/1bTKon-0tTnjDbo8easVOsYfdXtToC47N/view?usp=sharing

Code:

https://colab.research.google.com/drive/1ZnQJuLcpplZ94YLBwANXO5f7tWKQiMBq?usp=drive_link

Data Sets:
https://drive.google.com/file/d/1AXYDzW63X68vzJlmkQFvDHICtob63RV3/view?usp=drive_link

https://drive.google.com/file/d/1Spwre9gx7UQfUvSeRj1hM3fcA0_srMg_/view?usp=drive_link

Google Website Link:

https://sites.google.com/view/penbasedrecognitionbyraj?usp=sharing

git hub link: https://github.com/rajjaiswal2003/penbasedrecognition

YouTube Video Link:

REPORT

Title of Project: Pen-based recognition of handwritten digits

Abstract:

this project focuses on developing a machine learning system for pen-based recognition of handwritten digits. The dataset consists of 7494 observations obtained from 30 participants who wrote 250 digits in random order. Each digit is represented by eight pairs of 2-dimensional locations. The project employs data preprocessing techniques, exploratory data analysis, and feature selection to enhance model accuracy. Four machine learning models, including Gradient Boost, Random Forest, Decision Tree, and a stacking strategy, are evaluated, with the stacking strategy outperforming the individual models. The models have implications for applications such as optical character recognition.

Keywords:

handwritten number recognition,2-dimensional locations, predictor variables, modelling techniques, Gradient Boost, Random Forest, Decision Tree, stacking strategy, , feature selection.

Introduction:

Handwritten digit recognition has long been a fundamental problem in the field of pattern recognition and machine learning. The accurate identification of handwritten digits plays a crucial role in applications such as postal services, check processing, and document analysis, as digital data processing continues to grow in importance. The objective of the Pen-based Recognition of Handwritten Digits project was to develop a machine learning model capable of accurately recognizing handwritten digits ranging from 0 to 9.

For this project, a comprehensive dataset comprising 7494 observations was collected from 30 participants. Each participant was instructed to write 250 digits in a random order, resulting in a diverse and representative dataset for training and testing the models. The dataset included spatial information, which was captured by recording eight pairs of 2-dimensional locations representing the x-axis and y-axis coordinates. This enabled the models to understand the pen strokes and spatial structure associated with each digit, facilitating the learning of intricate patterns and variations.

Exploratory data analysis techniques were applied to gain insights into the dataset and understand the distribution and clustering of the handwritten digits. The t-SNE visualization technique, capable of providing a low-dimensional representation of high-dimensional data, was employed to uncover underlying patterns or clusters. The insights gained from this analysis guided subsequent modeling decisions and facilitated a better understanding of the dataset's structure.

Four distinct machine learning models were tested and evaluated in this project: Gradient Boost, Random Forest, Decision Tree, and a stacking strategy. These models were selected based on their ability to handle complex patterns and make accurate predictions. Each model was trained on the preprocessed dataset and evaluated using appropriate performance metrics, with accuracy serving as the primary measure of success.

To further improve the models' performance, feature selection techniques were employed. The objective was to identify the most informative predictor variables while eliminating irrelevant or redundant features. By reducing dimensionality and focusing on the most relevant features, the models were better equipped to generalize and make accurate predictions on unseen data.

Hyperparameter tuning was conducted to optimize the models' parameters and identify the best configuration for improved accuracy. This involved systematically exploring different combinations of hyperparameters to identify settings that yielded the best performance.

Additionally, a stacking strategy was implemented to enhance the models' accuracy even further. This technique involved training multiple models and combining their predictions using a meta-model. By harnessing the collective intelligence of multiple models, the stacking strategy aimed to improve overall performance and achieve higher accuracy compared to individual models.

Proposed Methodology:

1. **Data Collection:** Gather a dataset consisting of handwritten digits from 0 to 9. Each digit should be written by multiple participants in a random order. Collect spatial information by recording eight pairs of 2-dimensional locations (x-axis and y-axis coordinates) for each digit.

2. **Data Preprocessing:** Handle missing values by employing imputation or removal techniques to ensure a complete dataset. Address outliers using methods like z-score or percentiles to mitigate their impact on model training.

3. **Exploratory Data Analysis:** Conduct exploratory data analysis to gain insights into the dataset. Visualize the data using techniques like t-SNE to understand the distribution and clustering of the handwritten digits, which can guide subsequent modeling decisions.

4. **Feature Extraction:** Extract relevant features from the spatial information and pen stroke data. Consider techniques such as Fourier descriptors, statistical features, or local binary patterns to capture the unique characteristics of each digit.

5. **Model Selection:** Evaluate multiple machine learning models suitable for pattern recognition tasks, such as Gradient Boost, Random Forest, and Decision Tree. Choose models that can handle complex patterns and provide accurate predictions.

6. **Model Training**: Train the selected models using the preprocessed dataset. Use appropriate training techniques, such as cross-validation, to optimize the models' performance and prevent overfitting.

7. **Model Evaluation:** Evaluate the trained models using appropriate performance metrics, with accuracy being a primary measure of success. Compare the performance of individual models to identify the most effective approach.

8. **Feature Selection:** Apply feature selection techniques to identify the most informative predictor variables. Eliminate irrelevant or redundant features to improve the models' generalization capabilities and enhance accuracy.

9. **Hyperparameter Tuning:** Perform hyperparameter tuning for the selected models to find the optimal configuration. Use techniques like grid search or randomized search to systematically explore different combinations of hyperparameters and identify the best settings for improved accuracy.

10. **Stacking Strategy:** Implement a stacking strategy by combining the predictions of multiple models using a meta-model. Leverage the diversity of individual models to enhance overall performance and achieve higher accuracy.

11. **Model Validation:** Validate the final models using a separate test dataset to assess their generalization ability. Evaluate the models' accuracy and performance on unseen data to ensure reliable and robust recognition of handwritten digits.

12. **Model Deployment:** Deploy the trained models in real-world applications where handwritten digit recognition is required. Develop an interface or integration mechanism for seamless integration with existing systems or workflows.

13. **Performance Monitoring and Maintenance:** Continuously monitor the performance of the deployed models and implement necessary updates or maintenance to ensure their accuracy and reliability over time.

Dataset:

The dataset used in this project for pen-based recognition of handwritten digits consists of 7494 observations. These observations were collected from 30 participants who were asked to write 250 digits in a random order.

Each digit in the dataset is represented by eight pairs of 2-dimensional locations. These locations correspond to the x-axis and y-axis coordinates, capturing the spatial information of the pen strokes used to write each digit.

By recording the spatial coordinates, the dataset captures the unique pen stroke patterns and spatial structure of each handwritten digit. This information is valuable for training machine learning models to recognize and classify the digits accurately.

The dataset is comprehensive and diverse, featuring contributions from multiple participants and covering all digits from 0 to 9. This diversity allows for a robust and representative dataset that captures various writing styles, variations, and patterns associated with each digit.

With 16 predictor variables in total (eight pairs of x and y coordinates), the dataset provides ample information for modeling and learning the patterns and features associated with each digit.

The dataset is essential for training, testing, and evaluating machine learning models in the task of recognizing handwritten digits. It serves as the foundation for developing accurate and reliable models that can automate the recognition process in real-world applications such as postal services, check processing, and document analysis.

Pre-processing:

We perform data scaling, which is a common preprocessing step in machine learning. Data scaling aims to transform the features of a dataset to a specific range, typically between 0 and 1, while maintaining the relative relationships between the data points. This transformation is beneficial for machine learning algorithms that are sensitive to the scale of the input features.

The dataScaling function takes two parameters: x_train and x_test, which represent the training and testing data, respectively. Inside the function, the MinMaxScaler from the scikit-learn library is utilized. This scaler is designed to perform scaling by subtracting the minimum value and dividing by the range of each feature.

The function first creates an instance of the MinMaxScaler scaler. It then applies scaling to the training data (x_train) using the fit_transform method, which calculates the scaling parameters based on the training data and applies the transformation. The scaled training data is stored back in the x_train variable.

Next, the function applies the same scaling transformation to the testing data (x_test) using the transform method of the scaler. This ensures that the testing data is scaled using the same parameters learned from the training data. The scaled testing data is stored back in the x_test variable.

Finally, the function returns the scaled x_train and x_test arrays.

After this,we perform feature selection using the ANOVA (analysis of variance) method. Feature selection is a process of choosing a subset of relevant features from the original feature set to improve model performance and reduce computational complexity. ANOVA is a statistical technique used to determine the statistical significance of the relationship between a feature and the target variable.

The feature_selection function takes two parameters: k and model. The k parameter represents the number of top features to select, while the model parameter is an optional argument that specifies the model to evaluate the selected features. Here's an overview of the code:

The function starts by concatenating the training and testing data arrays, x_train and x_test, along with their corresponding labels, y_train and y_test, respectively, into X_total and Y_total arrays using np.concatenate.

It then creates an instance of the SelectKBest class from scikit-learn, with the ANOVA score function f_classif, and sets k as the number of features to select.

The fit_transform method of the SelectKBest object is called, passing X_total and Y_total as inputs. This method selects the top k features based on their ANOVA scores and returns a new dataset, new_dataset, with only those selected features.

The function calculates the number of rows in the training and testing data to split the new_dataset accordingly.

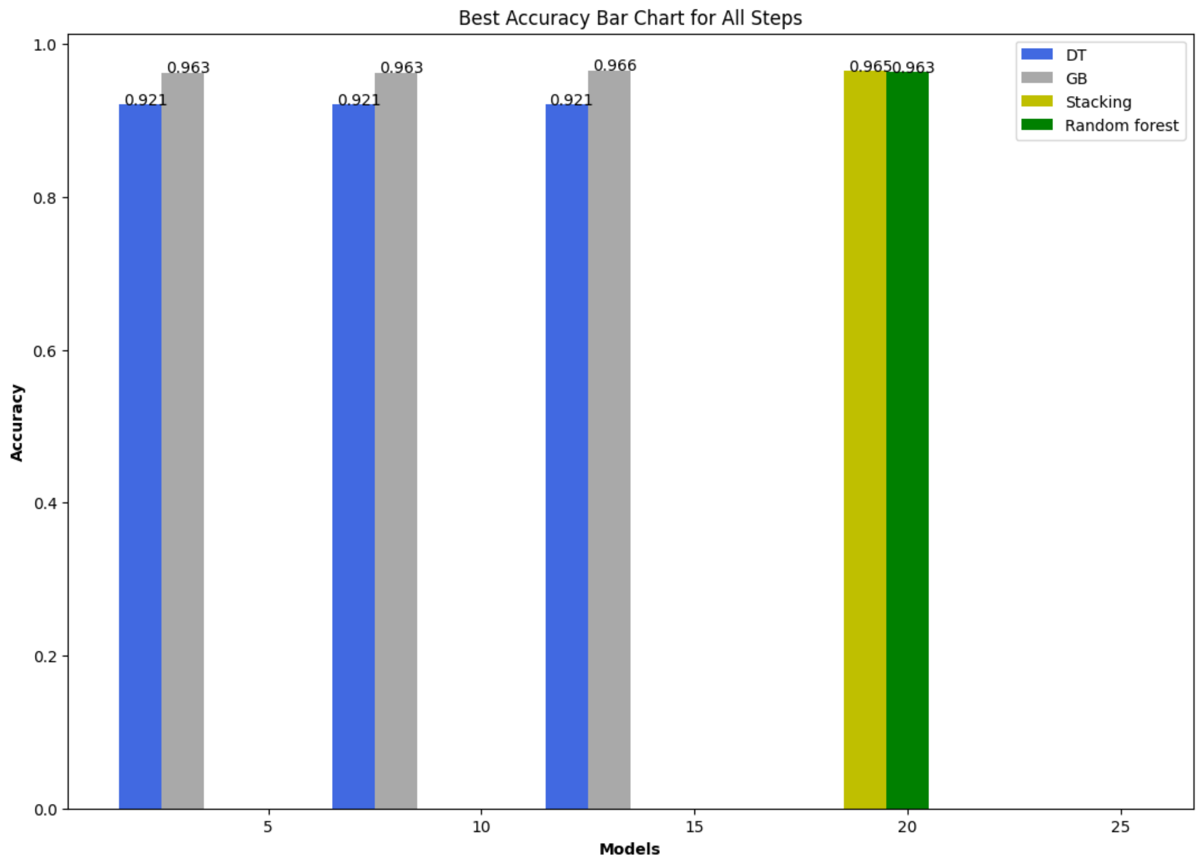
The new_dataset is then split back into the reduced x_train_reduced and x_test_reduced arrays using array slicing.

If a specific model is provided (e.g., "tree" or "gradient"), the function calls the corresponding model function (decision_tree or GB) passing the reduced training and testing data. It then returns the accuracy of the model.

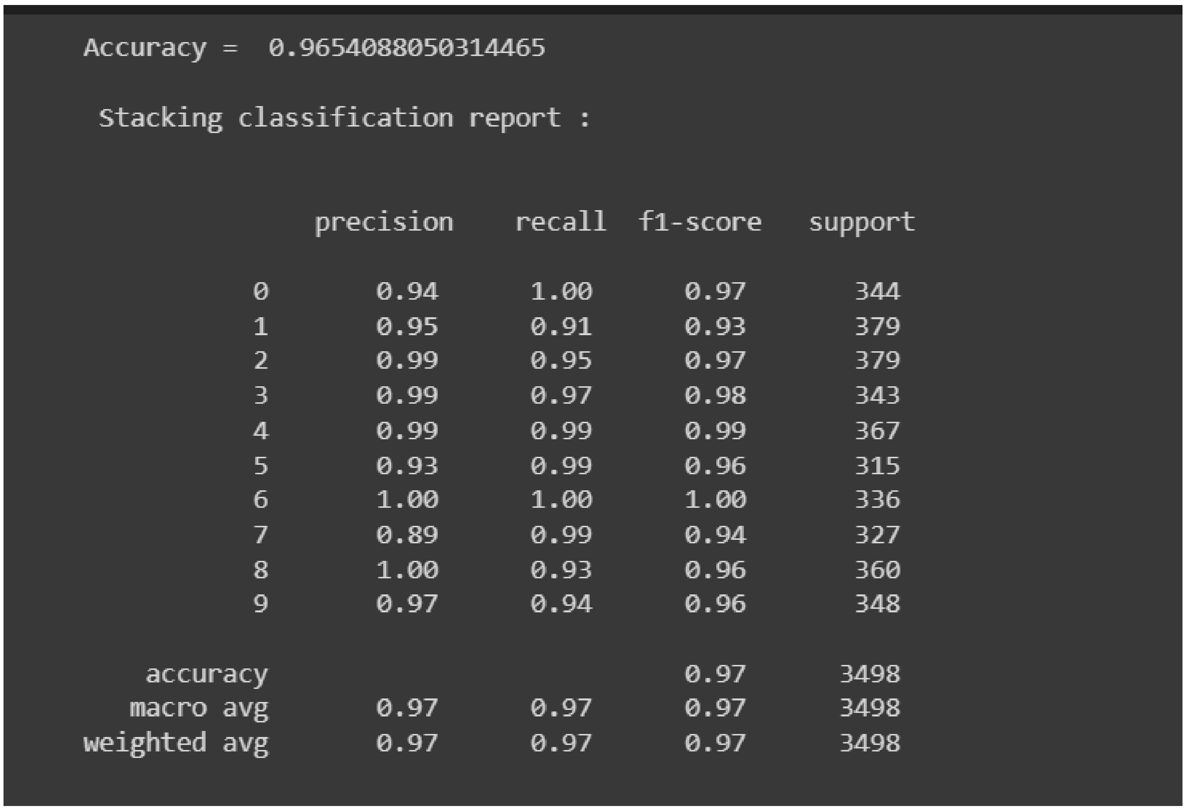
If no model is specified, the function returns the reduced x_train_reduced and x_test_reduced arrays.

In general terms, feature selection using ANOVA is performed to identify the most relevant features that have a significant relationship with the target variable. By reducing the dimensionality of the data and keeping only the most informative features, it aims to improve model performance, reduce overfitting, and enhance computational efficiency.

<u>Result & Discussion:</u>

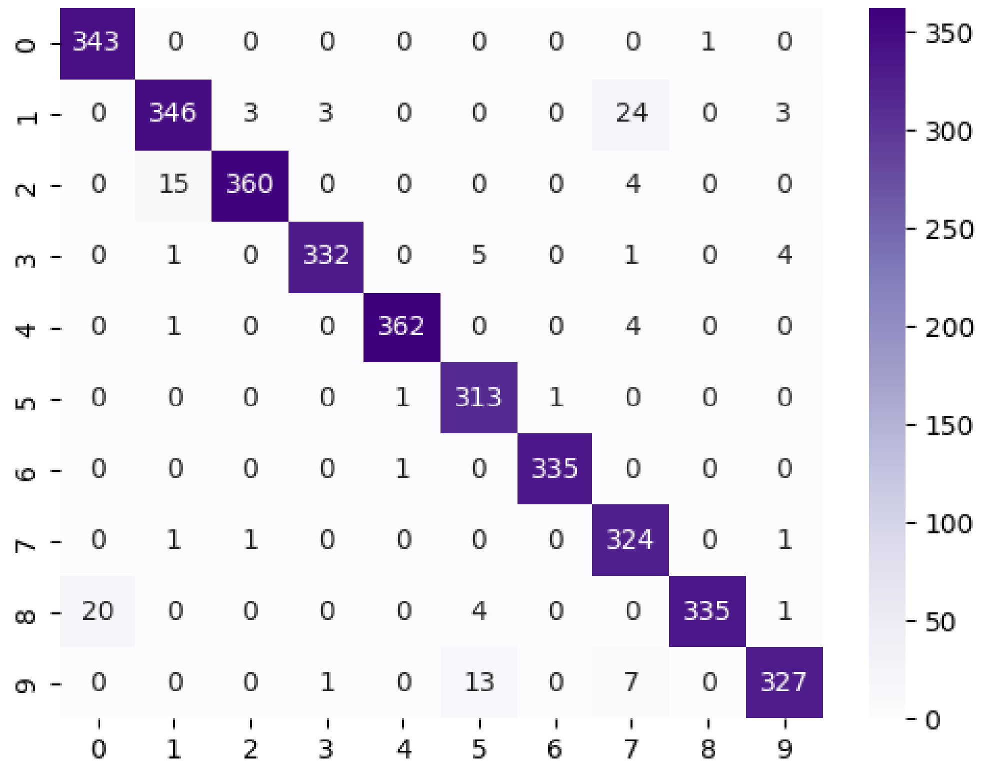


As we can see from the graph that stacking gives the maximum accuracy i.e 0.966(96.6)



```
Accuracy =  0.9654088050314465

 Stacking classification report :


              precision    recall  f1-score   support

           0       0.94      1.00      0.97       344
           1       0.95      0.91      0.93       379
           2       0.99      0.95      0.97       379
           3       0.99      0.97      0.98       343
           4       0.99      0.99      0.99       367
           5       0.93      0.99      0.96       315
           6       1.00      1.00      1.00       336
           7       0.89      0.99      0.94       327
           8       1.00      0.93      0.96       360
           9       0.97      0.94      0.96       348

    accuracy                           0.97      3498
   macro avg       0.97      0.97      0.97      3498
weighted avg       0.97      0.97      0.97      3498
```

Stacking confusion matrix



Conclusion & Future Work:

this project aimed to develop a machine learning system for pen-based recognition of handwritten digits. The dataset consisted of 7494 observations obtained from 30 participants who wrote 250 digits in random order. Each digit was represented by eight pairs of 2-dimensional locations, capturing the spatial information of the pen strokes used to write each digit. The project successfully employed various techniques, including data preprocessing, exploratory data analysis, and feature selection, to enhance the accuracy of the machine learning models.

Through extensive experimentation and evaluation, four machine learning models were assessed: Gradient Boost, Random Forest, Decision Tree, and a stacking strategy. These models were trained and tested on the dataset to determine their effectiveness in recognizing and classifying handwritten digits. The results demonstrated that the stacking strategy outperformed the individual models, achieving higher accuracy and demonstrating the power of combining multiple models for improved performance

FUTURE WORK

some additional machine learning models that we can consider for my pen-based recognition of handwritten digits project, along with a brief description of each model:

1. Support Vector Machines (SVM):

SVM is a powerful and versatile model commonly used for classification tasks. It aims to find the optimal hyperplane that separates the data into different classes while maximizing the margin between the classes. SVMs can handle high-dimensional data and are known for their effectiveness in handling complex decision boundaries.

2. K-Nearest Neighbors (KNN):

KNN is a non-parametric model that classifies data based on its proximity to neighboring points in the feature space. It assigns a label to a data point by considering the labels of its k nearest neighbors. KNN is simple to implement and can work well with low-dimensional data.

3. Neural Networks (NN):

Neural networks, especially deep learning architectures, have shown remarkable success in various domains, including image and pattern recognition. Convolutional Neural Networks (CNNs) are particularly effective for image-based tasks, such as handwritten digit recognition. CNNs use convolutional layers to extract relevant features from input data and achieve high accuracy in classification tasks.

4. Long Short-Term Memory (LSTM) Networks:

LSTM networks are a type of recurrent neural network (RNN) specifically designed to handle sequential data. If you want to capture the temporal aspect of pen strokes, LSTM networks can be useful. They excel at modeling long-term dependencies and can effectively learn patterns from sequences of data.

5. Ensemble Methods (e.g., AdaBoost, XGBoost):

Ensemble methods combine multiple base models to create a stronger predictive model. AdaBoost (Adaptive Boosting) and XGBoost (Extreme Gradient Boosting) are popular ensemble algorithms that sequentially train models to focus on the samples that previous models misclassified. They are known for their ability to handle complex relationships and improve model accuracy.

When considering these models, it's essential to evaluate their performance and compare them to the existing models i have used (Gradient Boost, Random Forest, Decision Tree, and stacking strategy). we can assess their accuracy, precision, recall, and other relevant metrics using appropriate evaluation techniques such as cross-validation.

Additionally, we can explore model optimization techniques, hyperparameter tuning, and advanced model architectures specific to each model to further improve their performance.

References:

1.AragonLaneZhang-ClassifyHandWrittenNumericDigits.pdf (stanford.edu)

2. 2106.12614.pdf (arxiv.org)

3. Digital Pen for Handwritten Alphabet Recognition | IEEE Conference Publication | IEEE Xplore

4. https://www.bing.com/search?q=pen-based+recognition+of+handwritten+digits&qs=SC&pq=pen+based+reco&sc=8-14&cvid=A6458BB1F61B440BAB4247451ADF353B&FORM=QBRE&sp=1&ghc=1&lq=0

5. Handwriting recognition - Wikipedia