Ein Bild, das Grafiken, Grafikdesign, Schrift, Silhouette enthält.

Automatisch generierte Beschreibung

Code Description – Machine Learning with Python

LETTER RECOGNITION WITH SCIKIT-LEARN AND EMNIST

Saki Fujita (Saki – 23-603-582)

Neil Keijzer (ZHCODZ – 20-606-596)

Art Mushica (ArtM13 – 20-612-677)

Laura Romero (Metal-gr76 – 22-621-395)

**Table of content**

[**1.** **Introduction** 1](#_Toc154148693)

[**2.** **Import and Load Data** 2](#_Toc154148694)

[i. Import Data 2](#_Toc154148696)

[ii. Loading the Data 2](#_Toc154148696)

[**3.** **Data Analysis** 4](#_Toc154148697)

[**4.** **Prediction model** 6](#_Toc154148698)

[i. Logistic Regression Model 6](#_Toc154148699)

[ii. Random Forest Model 7](#_Toc154148700)

[**5.** **Evaluation of the two Models** 8](#_Toc154148701)

[**6.** **Measuring the Model Performance** 9](#_Toc154148702)

[**7.** **Misclassified and correctly classified images with Predicted and Actual Labels** 11](#_Toc154148703)

1. **Introduction**

In this project, our objective is to develop a machine learning model capable of predicting the content of images from the EMNIST handwritten letters dataset. The primary goal of the model is to analyze input images, each measuring 28x28 pixels, featuring handwritten single letters from A to Z. The aim is to accurately classify each image, assigning it the appropriate letter label.

Notably, we adhere to a specific labeling convention where letters of the alphabet are uniquely identified with integers ranging from 1 to 26. For instance, the letter 'A' is labeled as 1, and the letter 'Z' is labeled as 26. This numerical representation simplifies the classification task, enabling the model to associate each handwritten letter with a corresponding integer label.

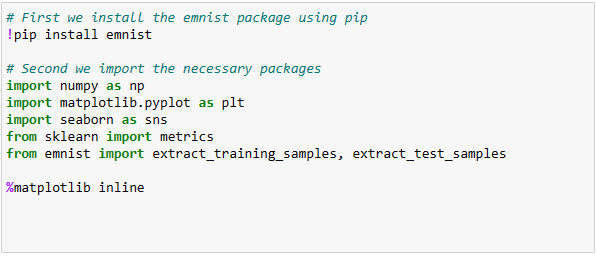
Ein Bild, das Typografie, Schwarzweiß, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

1. **Import and Load Data**

As a first step, we have to make sure, that the needed libraries are loaded, to handle the data.

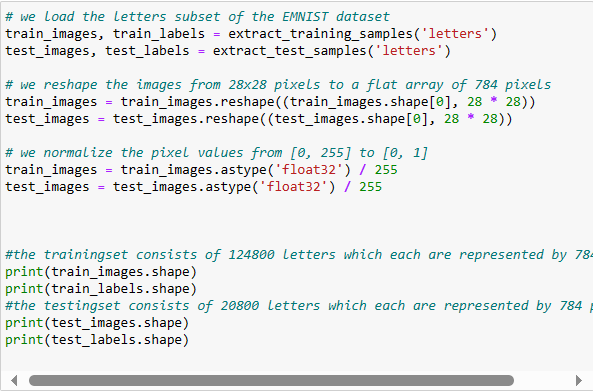
1. Import Data



In the code above we are setting up our Python environment to work with a dataset called EMNIST (Extended Modified National Institute of Standards and Technology). The EMNIST data set was downloaded from the mnist library and is a set of handwritten characters.

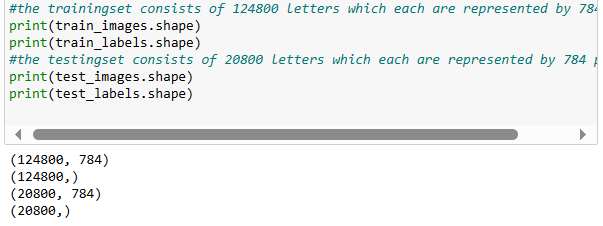
We also import other necessary packages, which will be used in the project.

1. Loading the Data



In the code above we are using the “extract” functions to load the letter images and their corresponding labels. “train\_images” and “train\_labels” are for training our model, while “test\_images” and “test\_labels” are for testing its performance later.

We then reshaped and normalized the dataset. This is usually done in practice to make the data handling simpler. We also found out that this is in our case also the case, due to the prediction models we will be using later on.

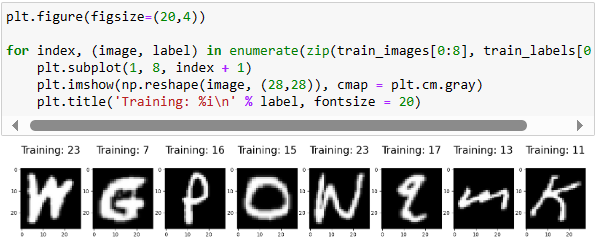


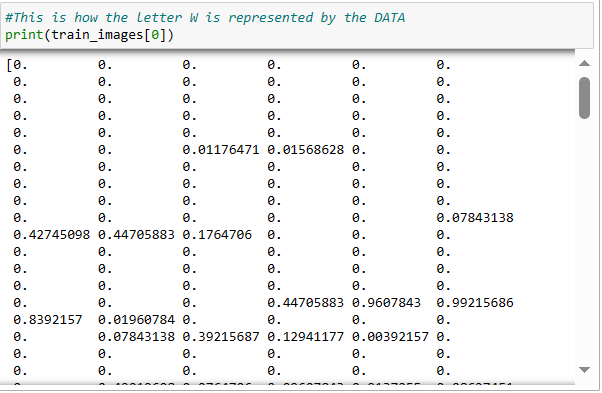
If we print the shapes this can be seen clearly.

1. **Data Analysis**

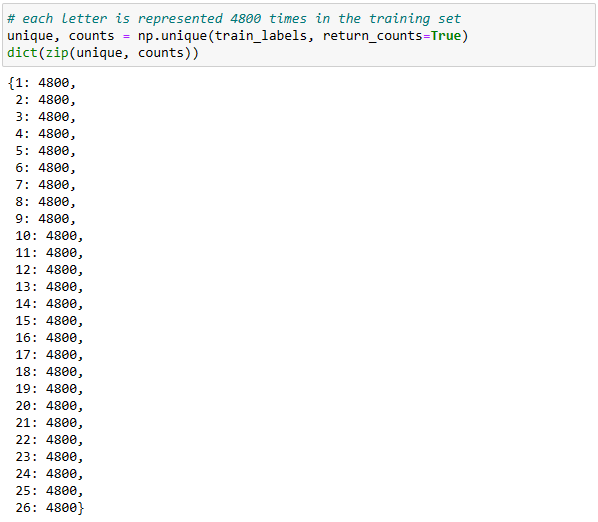
In this Chapter we will showcase the data set before using different data prediction models to the data.

Showing the first 8 Training Letters and their corresponding Labels

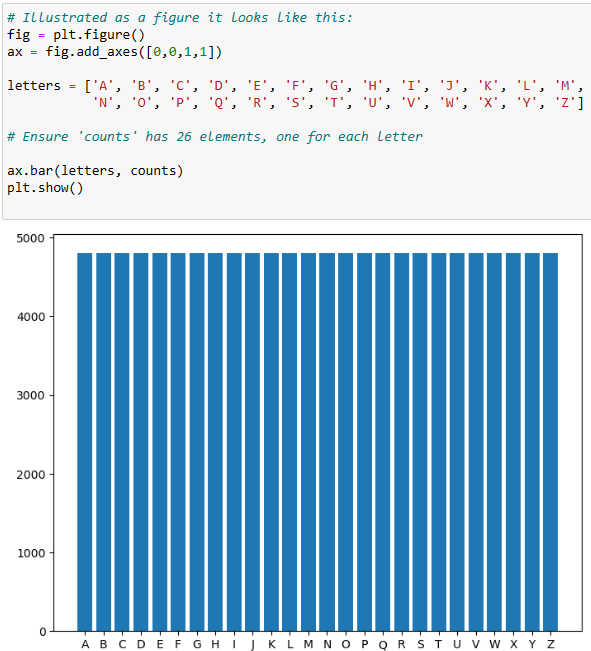


We use a loop to get the first 8 images and their corresponding labels. We than show these with the corresponding title.

Tabular representation of the letter “W” in the data set. Since the images have been flattened and normalized, as described in the previous explanations, what we are seeing is the one-dimensional array of pixel values that correspond to that image.



Furthermore we can see that each letter (1 = A; 26 =Z) is represented 4’800 times. 4’800 \* 26 = 124’800 which is also the whole training dataset. This can be illustrated as follows:



1. **Prediction model**

After going through the data and it’s specifications, we can now use the training data to predict what letter is being presented in the test dataset. Therefore we need to use different prediction models to make sure this happens and to evaluate the accuracy score. In this project we will focus on two prediction model, firstly the logistic regression model and secondly the random forest model. Since it takes a lot of computing power due to the large dataset, we are not able dive deeper into other prediction models.

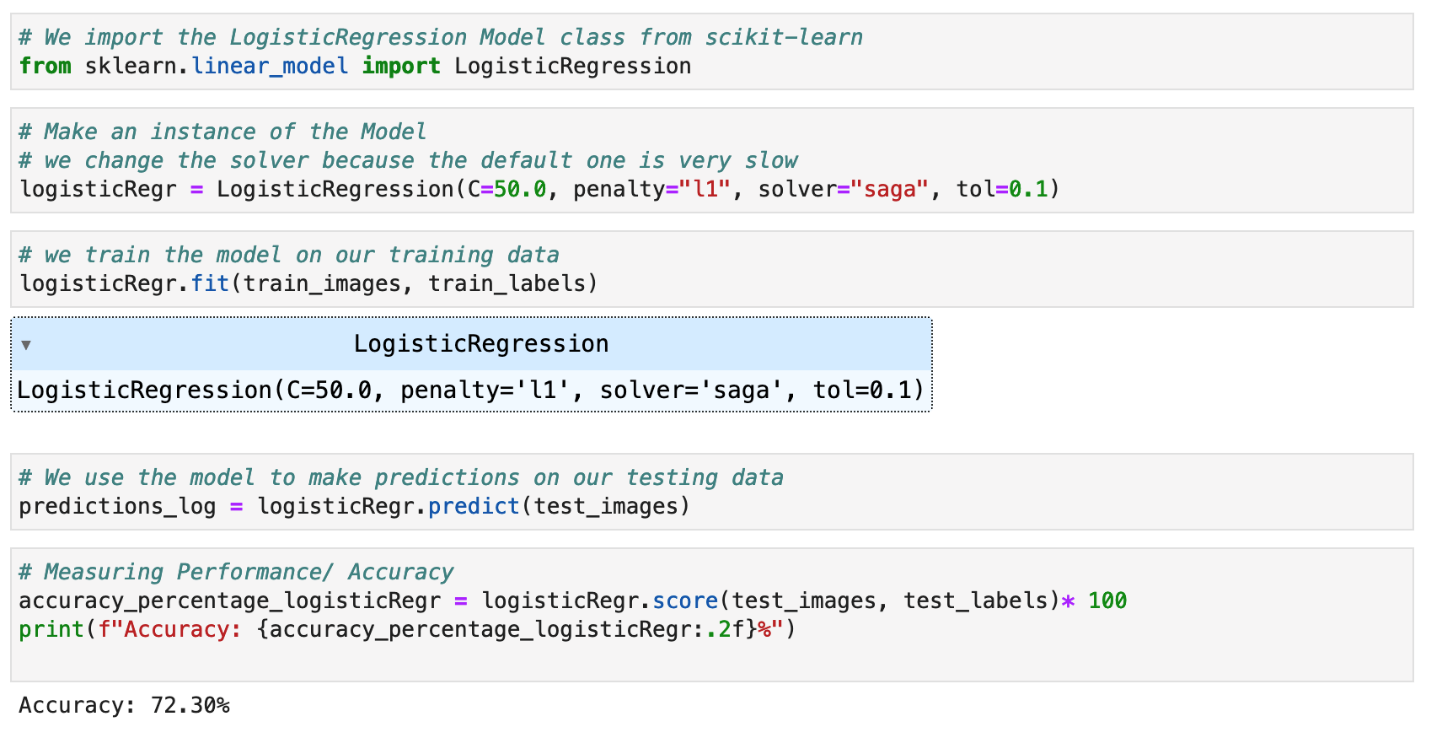
1. Logistic Regression Model

Logistic regression estimates the probability of an event occurring, such as happens or didn’t happen, based on a given dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1. This is particularly useful when we need to predict the probability as an output. This output can then be interpreted as the probability that the given input point belongs to a particular class.

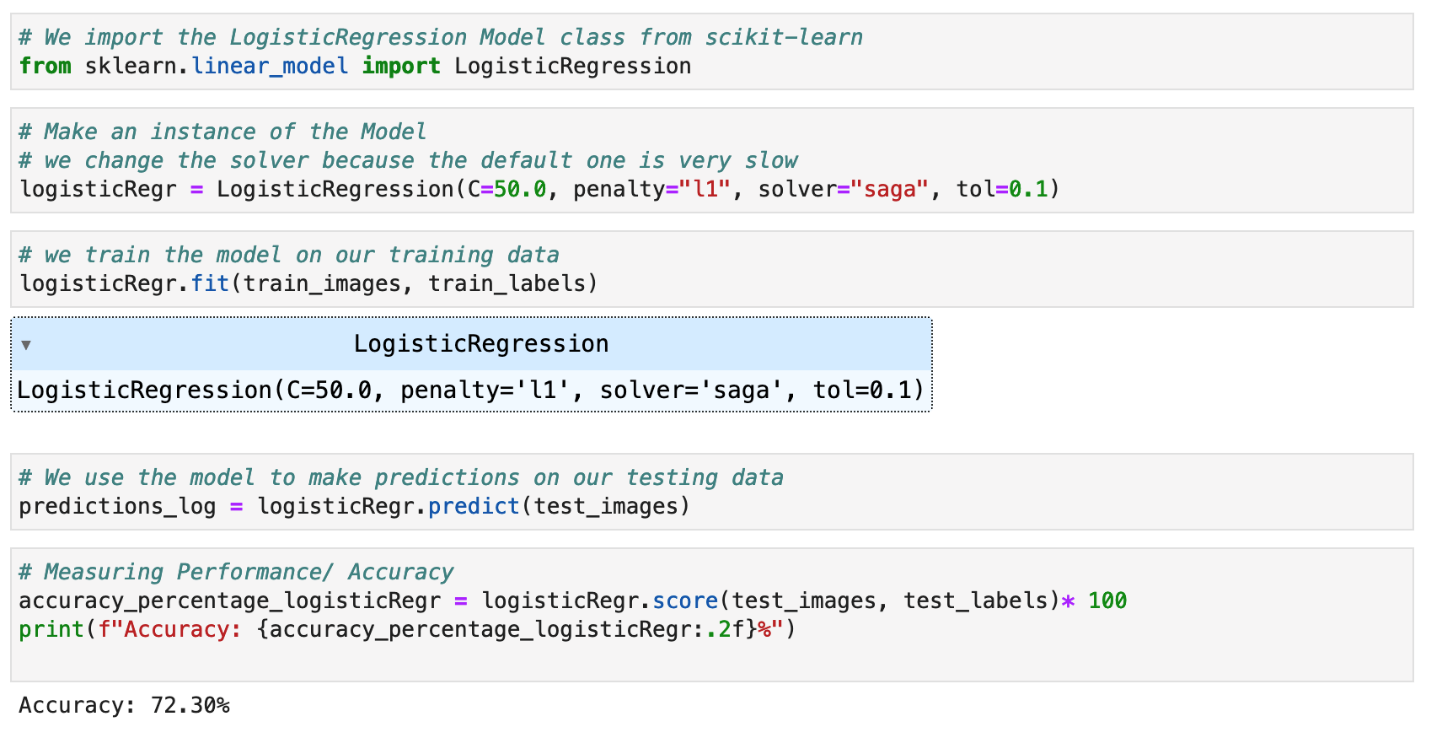
In our case of letter recognition, logistic regression operates by assigning weights to each pixel in an image. These weights reflect the importance of each pixel in determining whether the image represents a particular letter. For instance, if most of the dark pixels are in the shape of an 'E,' the model will calculate a high probability for 'E' and predict that the letter is an 'E.' It learns the right way to count these probabilities by looking at lots of training images of letters and their correct labels.

In our code we firstly import the LogisticRegression class from Scikit-learn, a machine learning library for Python. Then we create a Logistic Regression instance, where we are initializing specific parameters.

* **C=50.0** is the inverse of regularization strength; smaller values specify stronger regularization. (Regularization can help prevent overfitting the model to the training data.)
* **penalty='l1'** specifies the norm used in the penalization. The L1 penalty can lead to sparse models (with few coefficients), which can be useful if we believe many features are not relevant.
* **solver='saga'** is the algorithm to use for optimization. 'saga' is a variant of Stochastic Average Gradient descent which is faster for large datasets like in our case.
* **tol=0.1** is the tolerance for stopping criteria. This means the solver stops when the loss score does not improve by at least 0.1, indicating convergence.

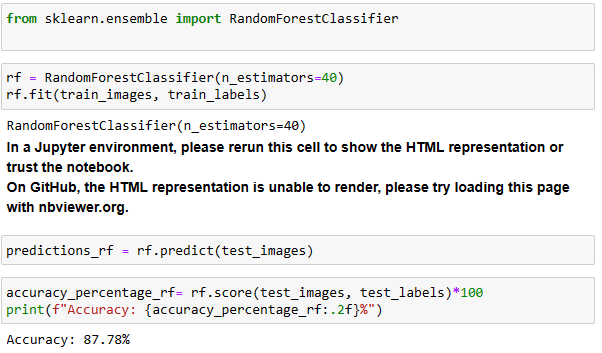


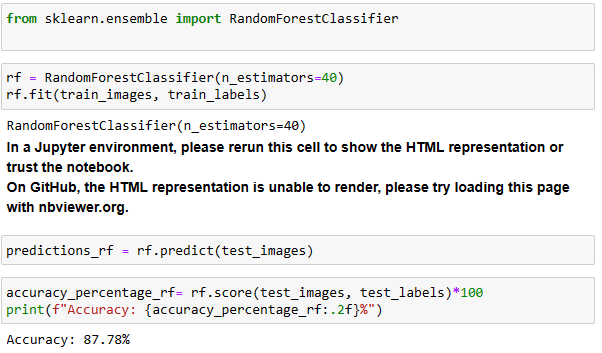
The we are training (fitting) the logistic regression model on our training data.

Last but not least We are using the trained model to predict labels for our test dataset. The predict method uses the learned logistic regression model to predict the labels of the test data. In our case we achieve an accuracy of 72.30% on the test data, which is a measure of how often the model correctly predicted the letter represented by each image.

1. Random Forest Model

A random forest conducts multiple random experiments (trees), each contributing its assessment. The algorithm ensures diversity in the model by randomly selecting subsets of features and data points to build each tree. Therefore the model combines many simple decision-making models (trees). Each tree looks at different parts of the data and makes a guess about the outcome. The forest then makes a final decision by taking the most common guess from all the trees. This method works well because it reduces errors from any single tree through averaging, making the overall prediction more reliable and this randomness helps to avoid the overfitting problem common with single decision trees.

For letter recognition, the random forest model creates a lot of decision trees, each trained with random subsets of the pixels from images of letters. Each tree gets to vote on what letter it thinks an image is based on the patterns it has learned. For example, if most trees see a round shape at the top and a straight line at the bottom, they might vote for 'P'. The model combines votes from all trees to decide which letter most likely represents the image.

Again, we firstly have to import the “RandomForestClassifier” class, which is the tool we used to build a random forest model. In the second block we made sure the model uses 40 decision trees.

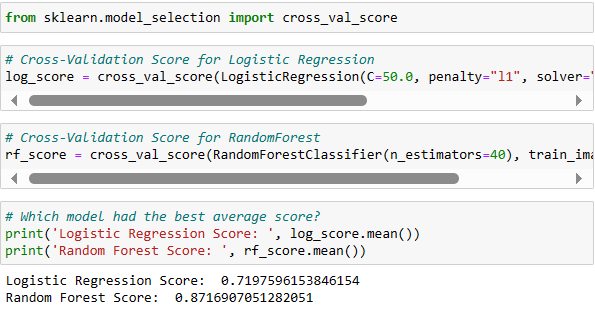
Like in the Logistic Regression model, the dataset has to be trained (fitted) using the Random Forest model. We also predict the accuracy of 87.78% (which is high than the last model: 72.30%).

1. **Evaluation of the two Models**

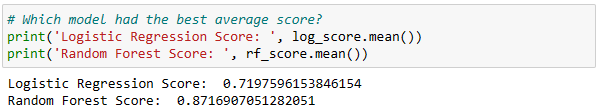
K-fold cross-validation is a known technique for assessing model performance, particularly when compared to the more common validation set approach. This method partitions the data into a predetermined number of subsets, or 'folds', and systematically uses one fold for validation while the others are used for training. This cycle is repeated such that each fold serves as the validation set exactly once.

In our case, where we try to intrpret handwritten letters, K-fold cross-validation offers a robust validation mechanism, ensuring that the model's accuracy is not just a result of the specific subset of data it was trained on, but generalizes well across the entire dataset. This method provides a more reliable measure of the model's ability to handle new, unseen data, which is crucial for tasks like letter recognition where variability in handwriting can be significant.

Again, we need to import the needed package. In this case the “cross\_val\_score” function, which is used to evaluate a model by cross-validation.



Now the class mentioned above is applied to the Logistic Regression and Random Forest Model. With different parameters we already explained. We can add here that the we did not modify the function “cross\_val\_score”, meaning by default, Scikit-learn uses a 5-fold cross-validation.



As can be seen above, using a 5-Fold Cross Validation, the Random Forest Classifier achieves a significantly higher score than the Logistic Regression. Therefore, the Random Forest model is best suited to predict the letters in our data. (Here we can se different score than before, which is due to the K-Fold technique used)

1. **Measuring the Model Performance**

For measuring the performance of the Random Forest Classifier, we use a confusion Matrix.

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte BeschreibungA confusion matrix is a table used in machine learning classification to evaluate the performance of a model. It summarizes the classification results by comparing actual and predicted class labels. In our case the matrix is organized along two axes: the predicted letters and the actual letters, both ranging from 'A' to 'Z'. Each cell in the matrix represents the count of instances corresponding to a specific combination of predicted and actual letters.

We defined a function named plot\_confusion\_matrix and used it to create a visual representation of a confusion matrix for predictions made by our random forest model.

It can be concluded that the model demonstrates overall accuracy, as evidenced by the prominent diagonal line in the confusion matrix. However, it faces challenges in distinguishing between certain pairs, notably I and L, as well as Q and G. Ein Bild, das Text, Screenshot, Diagramm, Reihe enthält.

Automatisch generierte Beschreibung

1. **Misclassified and correctly classified images with Predicted and Actual Labels**

Out of are two models the Random Forest Classifier performed the best for the prediction of the EMNIST data set with an accuracy of 87.78%. To better understand why images could be correctly classified or misclassified we display two sets of plots which both provide visual insights into the model's performance, highlighting instances where it succeeded or struggled in making accurate predictions.

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

Our machine learning model's accuracy in recognizing handwritten characters is likely affected by factors such as diverse writing styles, noise, and distortions in the input data, including unexpected dots or stripes in images that should not be present.