Name: Kodiak Ortiz, Matthew Roxas
Github Usernames: ortiz162, mroxas04
Purdue Usernames: ortiz162, mroxas04
Path: 3
Github Repository: https://github.com/mroxas04/ECE-20875-Final-Project.git
Instructor: Ding

Dataset Description

The original dataset consists of 10 classes, where each class contains roughly 180 samples to make for a total samples count of 1797. Each class is a number from the range of 0-9 and each sample is a handwritten drawing of a class that can be seen as a 8x8 digital image. In order to fully analyze this dataset we utilized the load_digits function in the sklearn.datasets library.

Method

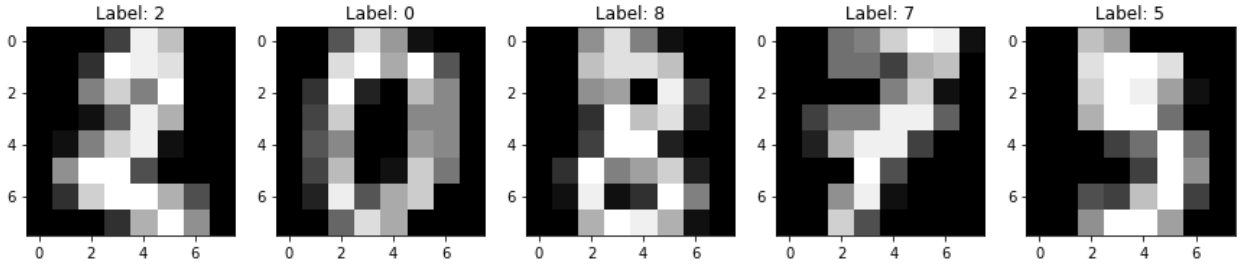
The objective of this project was to compare different types of machine learning models and compare their accuracies when trained on the same dataset. The models used were the Gaussian Naive Bayes model, which is a probabilistic model commonly used in text classification, the K-Nearest Neighbors Classifier, which uses similarities between instances to make predictions, and the Multilayer Perceptron Classifier, which is a neural network that is specifically designed for classification tasks. The accuracy of each model was then defined as the percentage of correctly predicted images within the testing dataset.

In order to train the models, the dataset was split into 40% training data and 60% testing data, so that the models can be trained without overfitting to the training data, and so that the effects of different types of training data can be clearly visualized. After training on the original dataset, each model was then fitted to the training set with added noise, while testing on the original testing set, and these results were recorded. In order to add noise to the training data, each image of the training dataset is being added to a noisy background instead of just a plain, black background. Finally, the noisy dataset was denoised using the KernelPCA method, with which different parameters were chosen to improve accuracy, and each of the three models were trained on the denoised dataset, and tested in the same manner. This method identifies the main components of each class, and filters out components of the image that don't contribute to the main patterns. This differs from the poisoned data because it ideally removes all of the noise created. To standardize the visualization, the same numbers were used, and the predicted labels were shown.

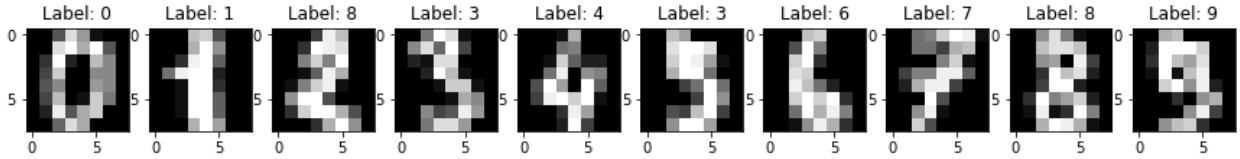
Results

Here are the results of training different models (Gaussian Naive Bayes, K-NeighborsClassifier, and MLPClassifier) based on different datasets. The results are the accuracy of the models, which is represented by the percentage of images correctly labeled. Examples of how each model labeled images of different numbers is also provided.

Testing dataset visualization:

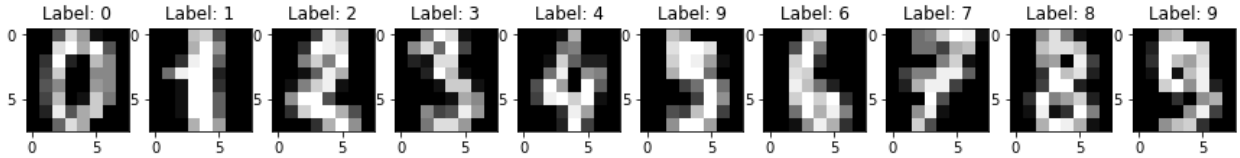


The Gaussian model:



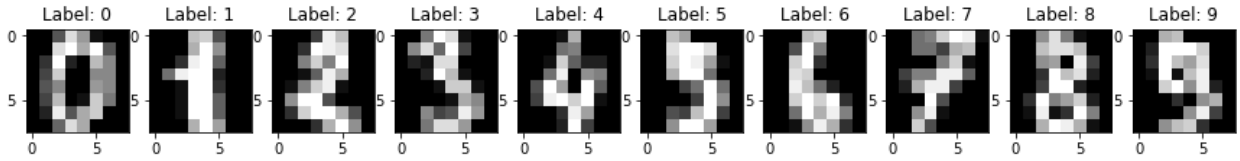
The overall accuracy of the Gaussian model is 0.8007414272474513

The KMeans model:



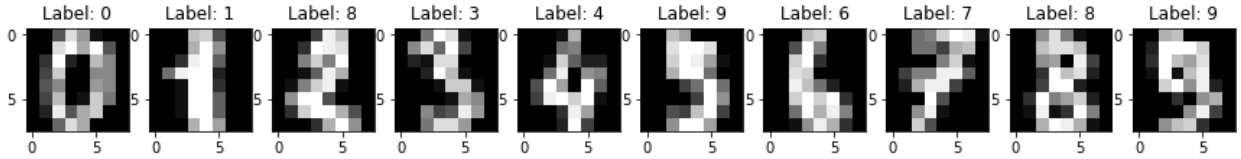
The overall accuracy of the K-Means model is 0.9545875810936052

The MLP model:



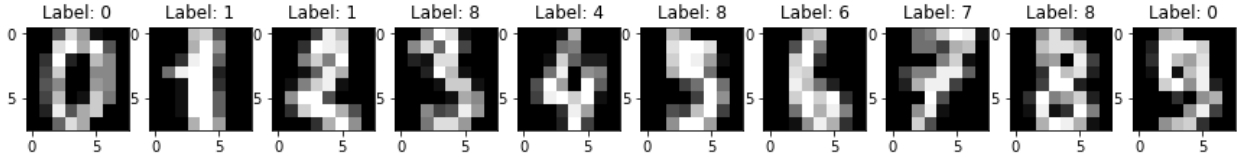
The overall accuracy of the MLP model is 0.9147358665430955

The Gaussian model with noise:



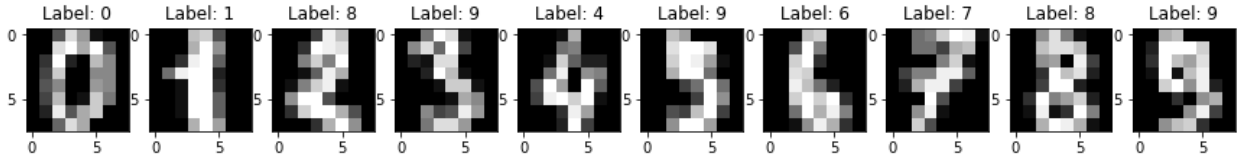
The overall accuracy of the Gaussian model with noise is 0.8146431881371641

The KMeans model with noise:



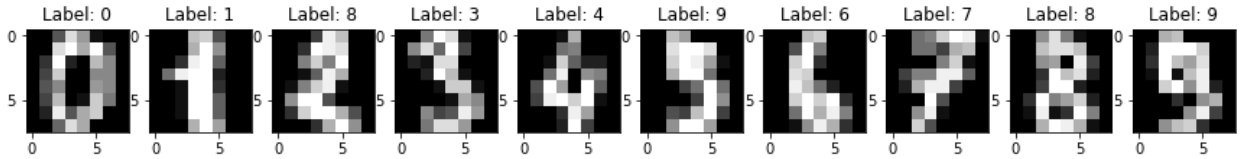
The overall accuracy of the KMeans model with noise is 0.6051899907321594

The MLP model with noise:



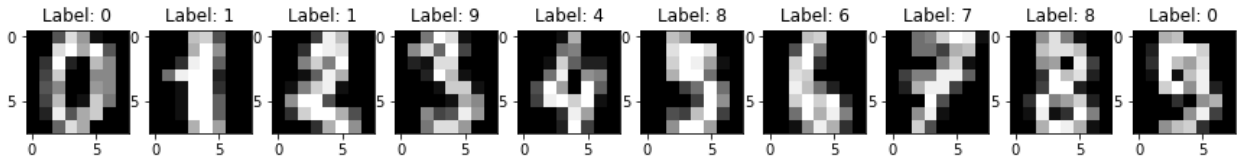
The overall accuracy of the MLP model with noise is 0.794253938832252

The Gaussian model with denoising:



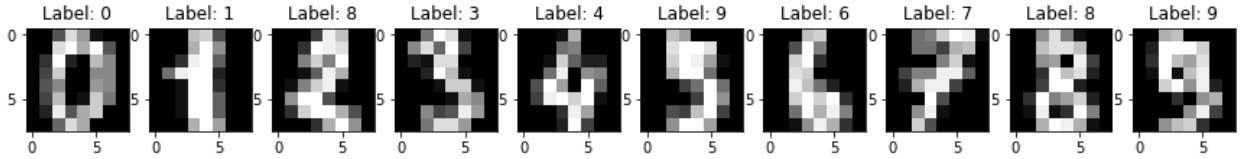
The overall accuracy of the Gaussian model with denoising is 0.8239110287303059

The KMeans model with denoising:



The overall accuracy of the KMeans model with denoising is 0.659870250231696

The MLP model with denoising:



The overall accuracy of the MLP model with denoising is 0.8035217794253939

Analysis

When looking at the accuracy of each of the models' performances with the raw training data, the K-Means model performs the best, followed by the MLP model, with the Gaussian model having the least accurate results. This makes sense because the K-Means model is a model that relies heavily on grouping together inputs that have very similar features. Because each number has very distinct features, this model will train very well in a classification task such as this. This is the same reason that the MLP model trains very well. Because the MLP model is a neural network that is able to slowly improve its accuracy over many iterations by changing its internal weights, it makes sense that this accuracy is also high, and that it correctly predicted all of the sample images. In the case of the Gaussian model, the accuracy is the lowest because it is a model based on probability, meaning that if it detects that a data point is close in probability to being two different numbers, there is a chance that it will misclassify this point, as seen in the mislabelling of the number 2.

After testing the models trained with the poisoned data, the Gaussian model showed the most robustness against the noise, with a small increase in accuracy, while the MLP model had a drop in accuracy of about 11%, and the KMeans model performed the worst, with a drop in accuracy of about 30%. The performance of the Gaussian model makes sense, because the noise is not systemic, so it is still able to create accurate distributions for different features in each class, without overfitting, meaning that it can perform somewhat well when testing with the unchanged testing data. The MLP model did relatively worse because the presence of noise in the training data means that the model fits well for testing data that has noise, as seen in the misclassification of the number 5, where it may interpret the white middle of the 5 as noise where the hole of the 9 is supposed to be. This is similar to the reason for the drop in accuracy for the KMeans model. Because its training is based on similarities within the noisy training data, it overgeneralizes and can't train well to the unchanged data.

After testing the models with the denoised training data, there was some improvement in accuracy across all three models, but it wasn't by a significant amount. This is most likely for the same reasons the models didn't perform significantly well for the poisoned data. When the training data is denoised, it probably removes the inherent

noise that the original dataset has. This essentially means that the models were trained on clean images and tested on noisy images. Some of the important features are still present in both the denoised and original dataset, which is why the accuracy is a little higher overall when compared to the models trained on the poisoned data, but the issue of overfitting still occurs. In order to improve the accuracy of these models, one possible solution would be to train each model on a combination of the original, noisy, and denoised images so that it can find the most relevant commonalities across all three types, and filter out the irrelevant features.