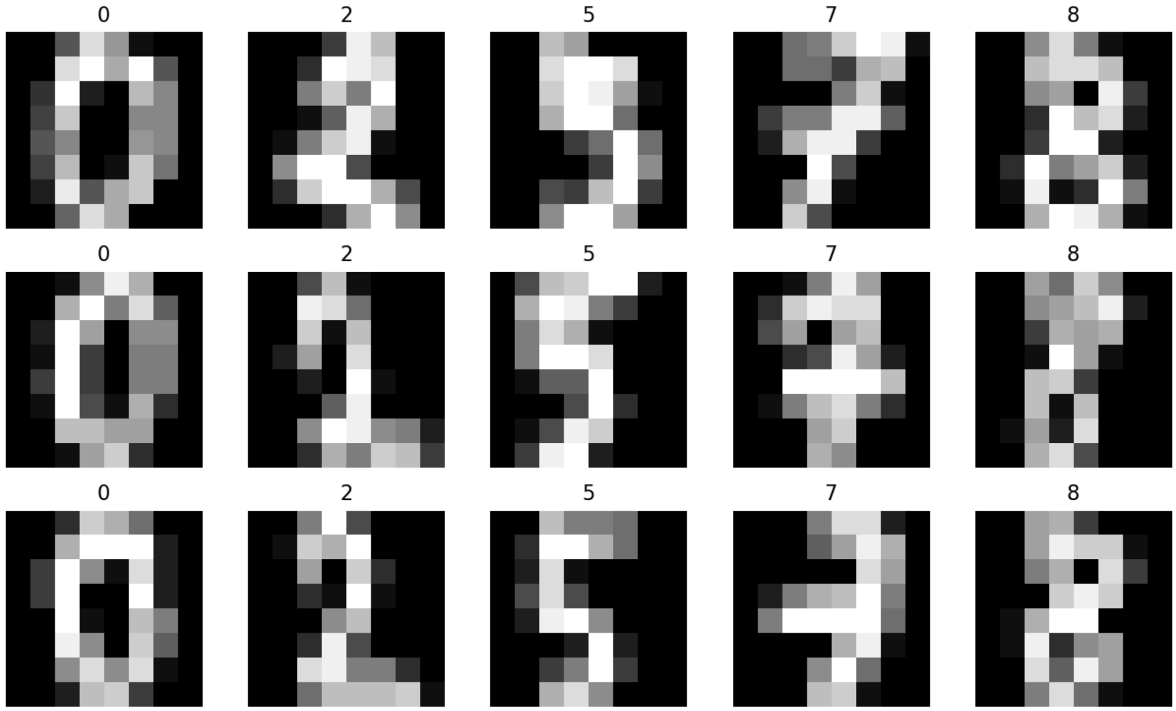
ECE 20875 Final Project - Path 3

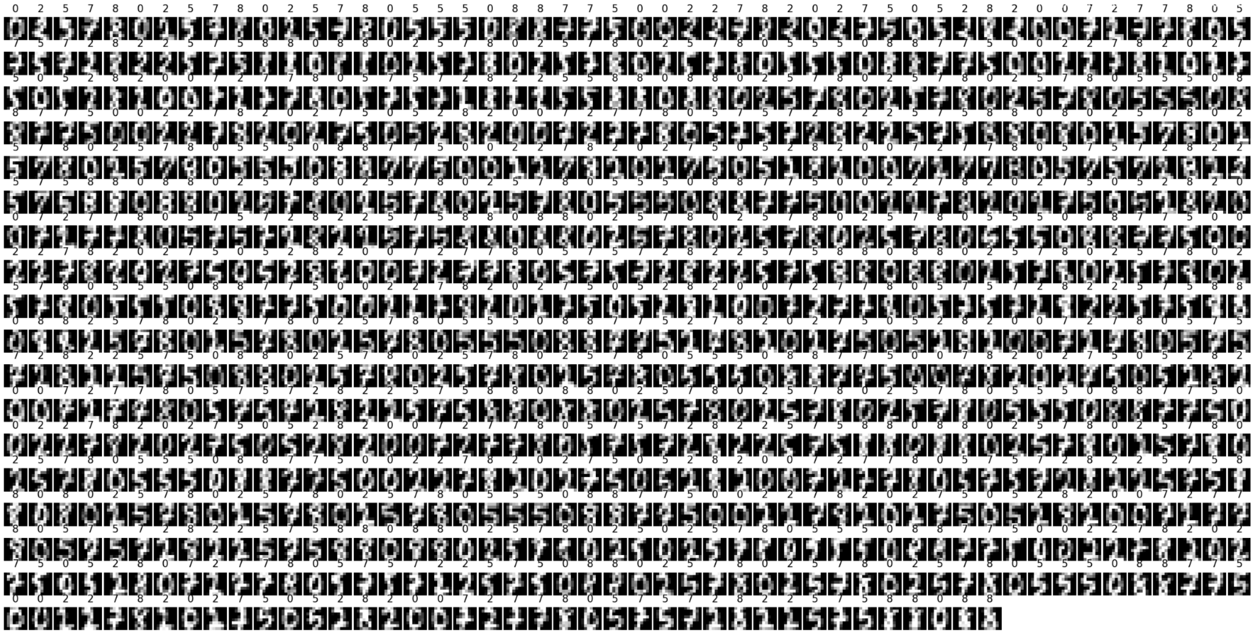
Team Members: Heng-I Chu, Xinyu Liu
Purdue Usernames: chu244, liu3680
Github Repository: https://github.com/Ivorchu/Data-Security-in-Model-Training.git

Dataset

In this project, we will be working with the scikit-learn digits dataset, which consists of 8x8 grayscale images of handwritten digits 0 through 9. The images were processed by flattening each 8x8 data into a vector with a size of 64, where each element represents the intensity of a pixel from 0 to 16. The dataset contains 1797 samples in total, that is evenly distributed across digits 0 through 9.



Print out and plot the numbers of the class [2, 0, 8, 7, 5]:



Analysis and Model Choices

We will train three different models: Gaussian Naive Bayes, K-Nearest Neighbors, and Multilayer Perceptron, using 40% of the total dataset. We will then test the accuracy of the three models using 60% of the total dataset and 100% of the dataset.

GaussianNB
Gaussian Naive Bayes is a simple yet powerful probabilistic classifier based on Bayes' Theorem. The model assumes that the feature values follow a Gaussian distribution for each class, and the probability of the labels can be predicted by combining multiple Gaussian distributions of the feature values.

KNN
K-Nearest Neighbors is a non-parametric learning algorithm that classifies a data point by finding the majority label among its k closest training samples in feature space.

MLP
Multilayer Perceptron is a model that simulates a neural network, consisting of an input layer, one or more hidden layers, and an output layer. Each layer contains neurons that contain linear and nonlinear components, allowing the model to learn complex relationships in the data.

Results on Normal Dataset

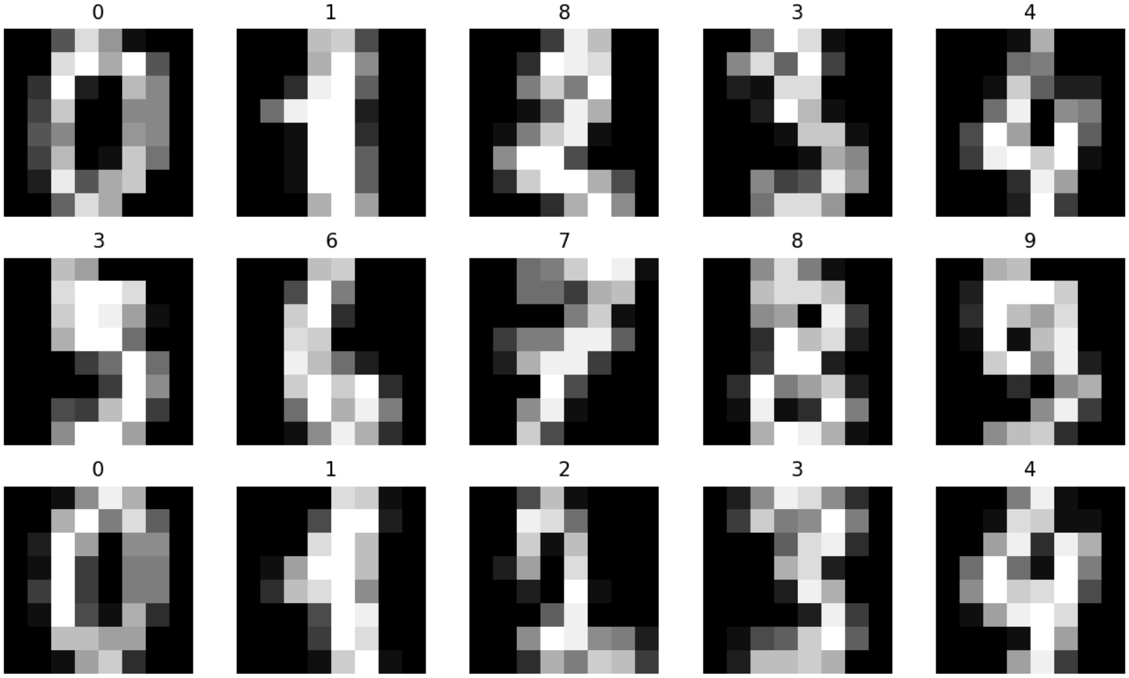
We have performed two tests using data from the same dataset to evaluate the accuracy of each model. In the first test, we used 60% of the original dataset, which was not used for training. In the second test, we used 100% of the original dataset, which contains 40% of the data that is used for training the model itself. We expect that the second test for any model will outperform the first test because it contains data that is used for training.

GaussianNB

Accuracy with test data: **0.8007414272474513**

Accuracy with the numbers [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]: **0.8408458542014469**

The Gaussian Naive Bayes model is the simplest model among the three. Its relatively low accuracy suggests that the pixel intensities in the digit data set do not follow a Gaussian distribution very strongly. Yet, it is still a decent baseline with over 80% accuracy. The results for the two tests were as expected, since the second test included some training data and had higher accuracy than the first test.

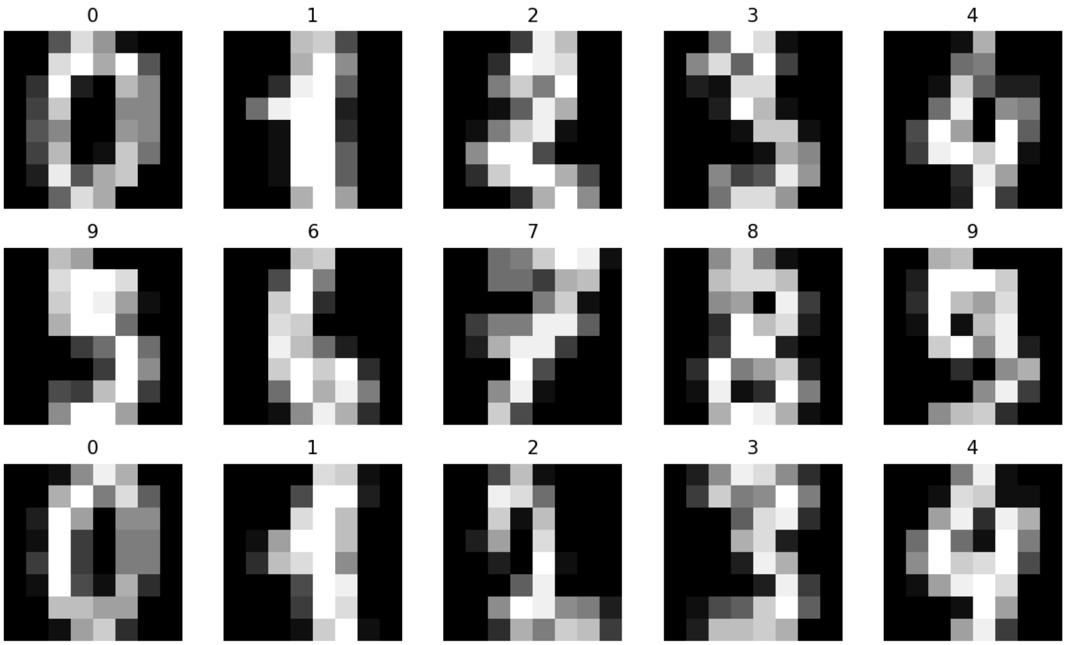


KNN

Accuracy with test data: **0.9545875810936052**

Accuracy with the numbers [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]: **0.9677239844184753**

The K-Nearest Neighbors performs the best among the three models. The model's high accuracy suggests that similar digits in pixel space are clustered closely, and it effectively captures the local structure of the data. The results for the two tests were as expected, since the second test included some training data and had higher accuracy than the first test.

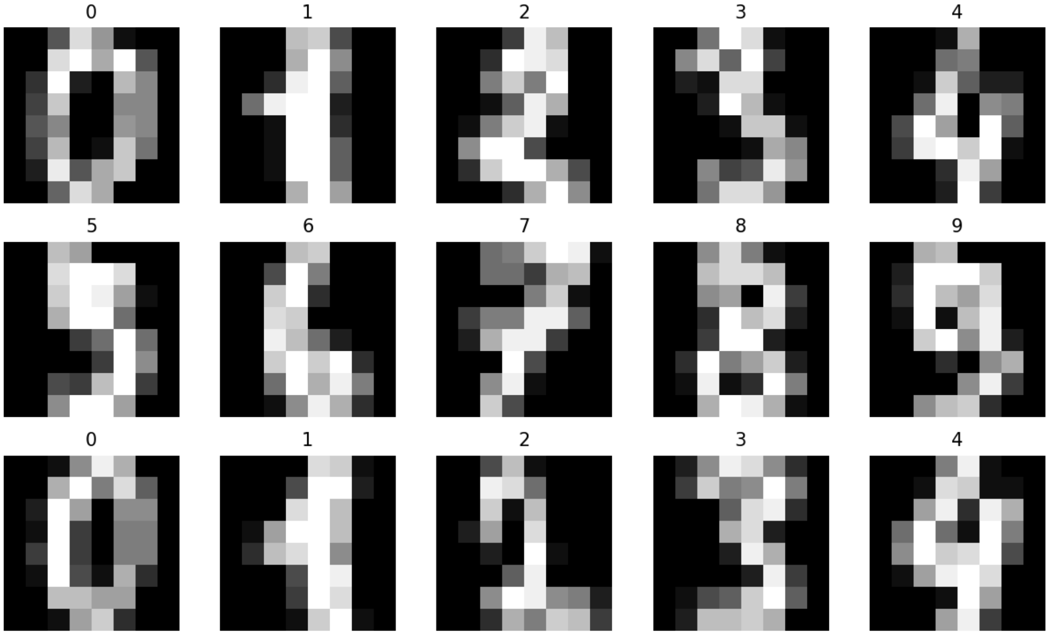


MLP

Accuracy with test data: **0.9147358665430955**

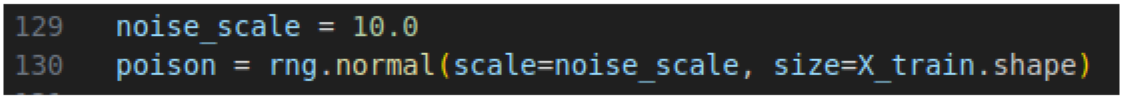
Accuracy with the numbers [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]: **0.9488035614913745**

The Multilayer Perceptron performs better than GaussianNB but not as well as KNN. Although the MLP model is more flexible and powerful, it did not outperform KNN in this experiment because the number of training iterations was relatively low. The MLP did not have enough time to fully converge and demonstrate its full potential. The results for the two tests were as expected, since the second test included some training data and had higher accuracy than the first test.



Poison Dataset

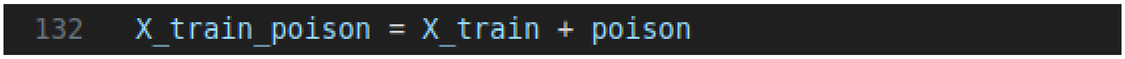
The poison in this project is represented by: on lines 129, 130



```
129   noise_scale = 10.0
130   poison = rng.normal(scale=noise_scale, size=X_train.shape)
```

We set poison to be a 3D nparray, size of samples, from a normal distribution of mean=0 and std=noise_scale.

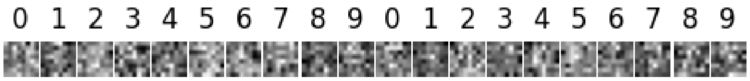
Then: on line 132



```
132   X_train_poison = X_train + poison
```

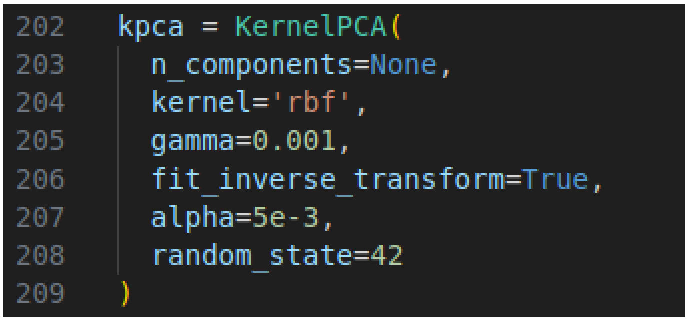
We add them up to mess with original intensity pixel data.

We get the result looks like picture below:



Denoised Dataset

Kernel PCA is used in this project: on lines 202-209



```
202   kpca = KernelPCA(
203       n_components=None,
204       kernel='rbf',
205       gamma=0.001,
206       fit_inverse_transform=True,
207       alpha=5e-3,
208       random_state=42
209   )
```

Kernel PCA (Principal Component Analysis) is able to explore high-dimensional features, instead of only linear, and the RBF (Radial Basis Function) kernel is used here.

γ parameter:

$k(x, y) = e^{(-\gamma||x-y||^2)}$

According to the formula, we can know that γ is a parameter to control the distance of the inputs, and small γ means underfitting, linear-like behavior, whereas big γ means overfitting.

α parameter:

This is a ridge regression, L2 penalty for RBF, in this case.

According to the lecture for regularization, we know that:

$$min_{\beta}||X\beta - y||_2^2 + \lambda||\beta||_2^2$$

In this case, we don't use linear regression, but with matrix for RBF and KPCA feature coefficients for inverse_transform, and we use α to represent λ for ridge penalty here. The effect of it is just like to smooth out the image, and makes it easier to identify.

We get the result looks like picture below:

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9

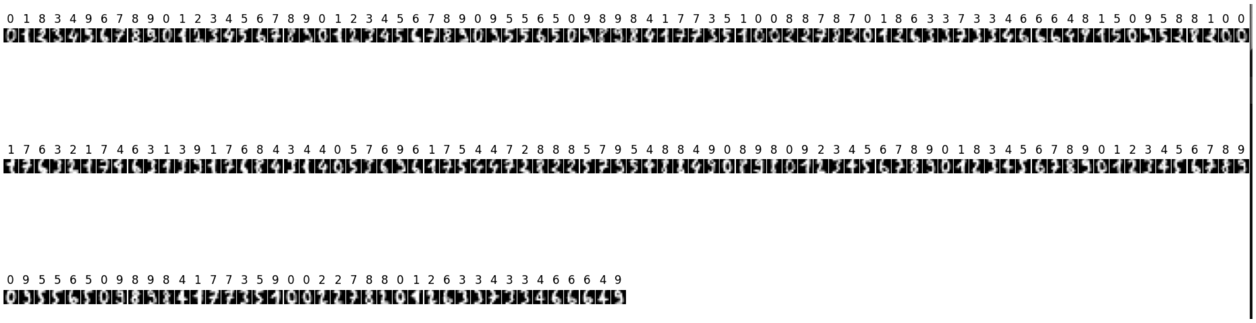
Results on Poisoned/Denoised Dataset

Poisoned Dataset

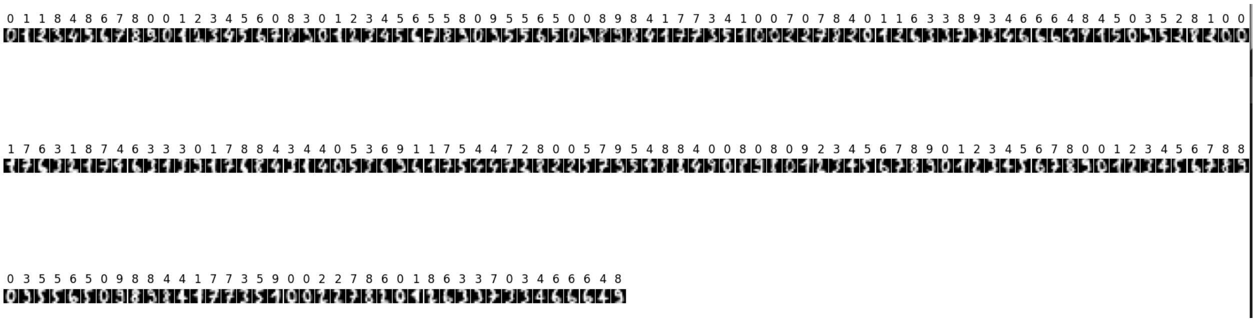
Table 1:

Model/Data	Original on Test	Poison on Test	Original on Num	Poison on Num	Denoise on Test	Denoise on Num
Gaussian	0.8007414272474513	0.8146431881371641	0.8408458542014469	0.8319421257651641	0.8489341983317887	0.8653311074012242
KNN	0.9545875810936052	0.6051899907321594	0.9677239844184753	0.6505286588759043	0.8266913809082483	0.8619922092376182
MLP	0.9147358665430955	0.794253938832252	0.9488035614913745	0.8046744574290484	0.8257645968489342	0.8553144129104062

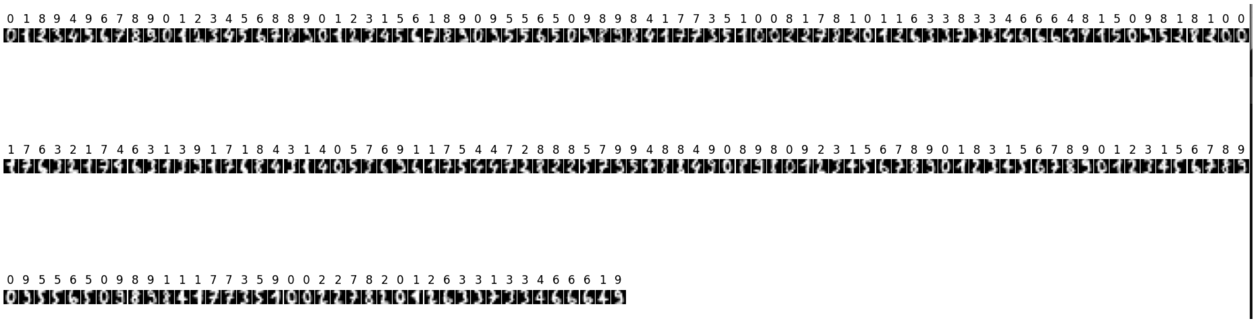
Poisoned Gaussian: [:200]



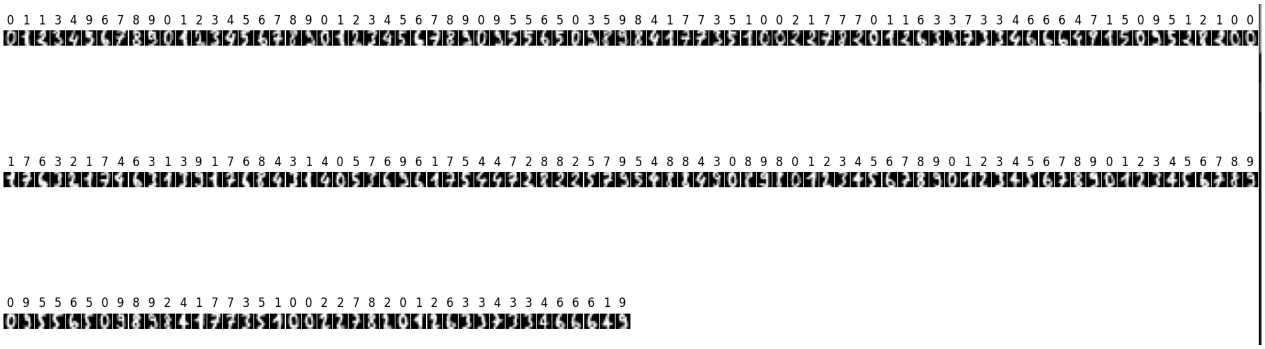
Poisoned KNN: [:200]



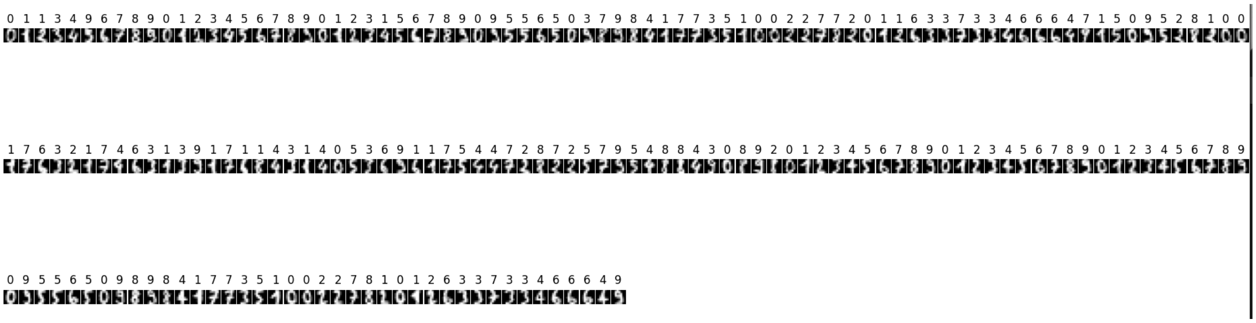
Poisoned MLP: [:200]



Denoised Gaussian: [:200]



Denoised KNN: [:200]



Denoised MLP: [:200]

0 1 2 3 4 9 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 9 5 5 6 5 0 3 5 9 8 4 1 7 7 3 5 1 0 0 2 7 7 8 7 0 1 2 6 3 3 7 3 3 4 6 6 6 4 7 4 5 0 9 5 2 2 7 0 0
1 7 6 3 2 1 7 4 6 3 1 3 9 1 7 1 5 4 3 1 4 0 5 7 6 9 1 1 7 5 4 4 7 2 8 2 2 5 7 9 5 4 8 2 4 9 0 2 9 2 0 9 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 4 9
0 9 5 5 6 5 0 9 8 9 2 4 9 7 7 3 5 9 0 0 2 2 7 8 2 0 1 2 6 3 3 7 3 3 4 6 6 6 4 9

According to table 1, we notice that for both of the test conditions, most of the models have a drop in accuracy because of poisoning, which is reasonable, because adding in outside data to the training set will cause models to perform worse.

However, the result of the Gaussian model on the test set has higher accuracy. I think this is because the Gaussian model will have a bigger variance with training of poisoned data, and this may cause 60% of testing data to fit in the variance.

Therefore, a worse model actually generalizes for the test set, and becomes better, while KNN and MLP both have a decrease in accuracy drastically, because they are not flexible enough, and misguided by the poisoned data.

Denoise Dataset
According to table 1, all of the models on both of the test conditions have a higher accuracy. The KNN model has the highest gain from poison to denoise, which is because, I think, poison will destroy the essence of KNN to only determine based on distance, while denoised data brings back the meaning of distance in the image by implementing the Inverse Transform feature.

Terminal Output

```
Model 1 trained...
The overall results of the Gaussian model with test data is 0.8007414272474513
The overall results of the Gaussian model with the numbers [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] is
0.8408458542014469
Model 2 trained...
The overall results of the KNN model is 0.9545875810936052
The overall results of the KNN model with the numbers [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] is
0.9677239844184753
Model 3 trained...
The overall results of the MLP model is 0.9147358665430955
The overall results of the MLP model with the numbers [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] is
0.9488035614913745
Model 1 poison trained...
The overall results of the Gaussian model poison is 0.8146431881371641
The overall results of the Gaussian model poison with the numbers [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] is
0.8319421257651641
Model 2 poison trained...
The overall results of the KNN model poison is 0.6051899907321594
The overall results of the KNN model poison with the numbers [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] is
0.6505286588759043
Model 3 poison trained...
The overall results of the MLP model poison is 0.794253938832252
The overall results of the MLP model poison with the numbers [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] is
0.8046744574290484
Samples of poisoned training sets:
Some denoised training samples:
Model 1 denoised trained...
The overall results of the Gaussian model denoised is 0.8489341983317887
The overall results of the Gaussian model denoised with the numbers [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
is 0.8653311074012242
Model 2 denoised trained...
The overall results of the KNN model denoised is 0.8266913809082483
The overall results of the KNN model denoised with the numbers [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] is
0.8619922092376182
Model 3 denoised trained...
The overall results of the MLP model denoised is 0.8257645968489342
The overall results of the MLP model denoised with the numbers [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] is
0.8553144129104062
```