Mini Project Report: Data Security in Model Training

1. Team information

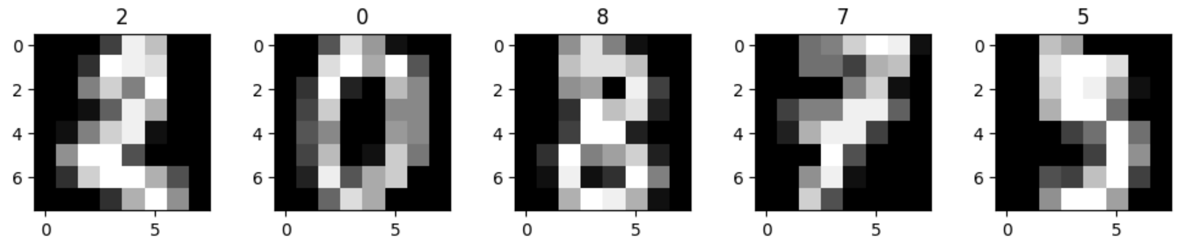
- Team Members: Abhiraj Jaswal
- Purdue Username: jaswal
- Path chosen: Path 3 - Data Security in Model Training
- Project Repository link: https://github.com/ECEDataScience/project-s25-Astromonkey2

:

2. Dataset Description:

This Project uses the digits dataset from scikit-learn, which consists of 8 by 8 pixel grayscale images of handwritten digits from 0 to 9. Each image is represented as an 8 by 8 matrix where the value of each cell indicates the darkness of that pixel. The dataset contains 1797 samples, with approx 180 samples per class. For this project I have split the data into training (40%) and testing (60%) without shuffling to maintain consistency.

Here is a sample visualization of digits 2, 0, 8, 7, 5 from the data set:



This dataset provides a good testing ground from model security as we can clearly observe the effects of poisoning and denoising the data.

3. Analysis:

1. Model Selection

For this project, I have selected 3 different classification models with different approaches

- Gaussian Naive Bayes: Gaussian Naive Bayes applies Bayes’ theorem under the assumption that each feature is normally distributed and conditionally independent given the class (scikit-learn developers, n.d.-a). It typically performs well on small datasets and has different weaknesses to noise compared to other models.

- K Nearest Neighbors (KNN) : k-Nearest Neighbors classifies a sample by finding its kkk closest neighbors (under Euclidean distance by default) in the training set and taking a majority vote of their labels(scikit-learn developers, n.d.-b).KNN with k = 10 was selected because its sensitive to local data patterns. This classifier is probably the most vulnerable to data poisoning because it relies on training distances.

- Multi Layer Perceptron(MLP): A Multi-Layer Perceptron is a feed-forward neural network consisting of one or more hidden layers of neurons with nonlinear activation functions. Training proceeds by back-propagating the classification error through the network and updating weights via stochastic optimization (scikit-learn developers, n.d.-c).

  For evaluation, we use overall accuracy (percentage of correctly classified instances) as our primary metric, as it provides a straightforward measure of performance.

2. <u>Data Poisoning Analysis</u>

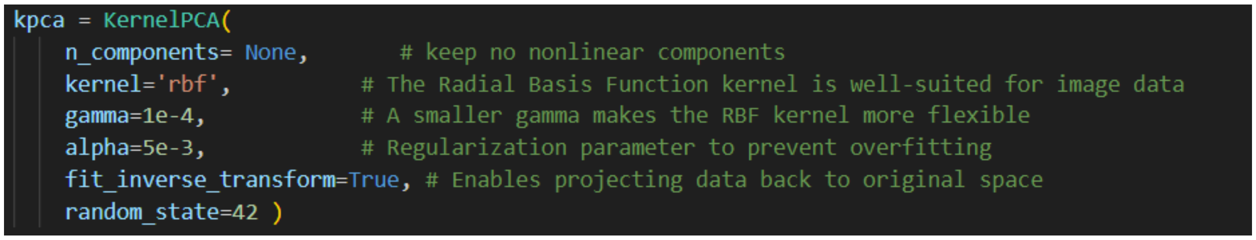
   Data poisoning is an adversarial attack in which an attacker deliberately tampers with the training data to degrade a model’s performance at test time (Wikipedia contributors, 2024). In this project , they poisoned the training data by adding Gaussian noise with a scale factor of 10.0 to each image. This significant noise level simulates a substantial attack on data integrity.

3. <u>Denoising Approach</u>

To counter the data poisoning, I applied Kernel Principal Component Analysis (KernelPCA) as a denoising technique. KernelPCA works by first mapping data into a high-dimensional feature space via a kernel (RBF) and then performs PCA in that space. The inverse_transform with regularization (alpha) reconstructs a smoothed version of the original inputs (scikit-learn developers, n.d.-d).

I selected KernelPCA because:

- It can capture nonlinear relationships in the data
- It's particularly effective for image denoising tasks
- It allows for tunable parameters to control the denoising strength



```
kpca = KernelPCA(
    n_components= None,        # keep no nonlinear components
    kernel='rbf',           # The Radial Basis Function kernel is well-suited for image data
    gamma=1e-4,             # A smaller gamma makes the RBF kernel more flexible
    alpha=5e-3,             # Regularization parameter to prevent overfitting
    fit_inverse_transform=True, # Enables projecting data back to original space
    random_state=42 )
```

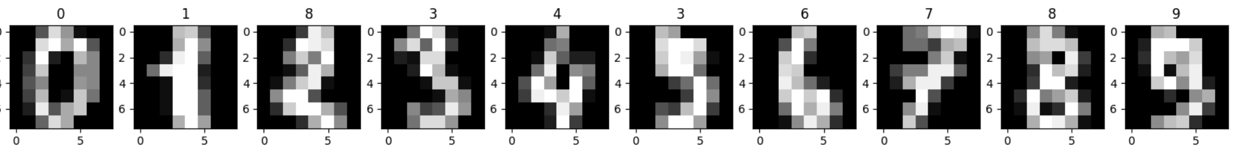
4. Results

1. Initial model Performance

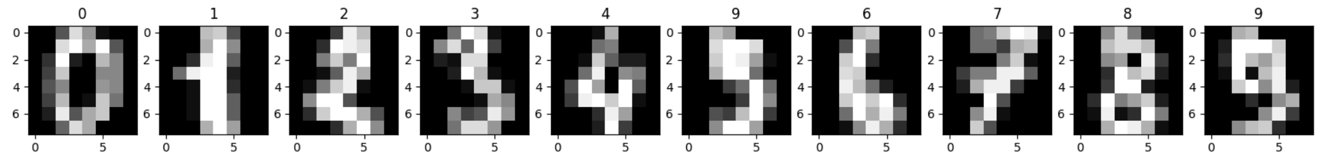
Model	Accuracy
GaussianNB	80.07%
KNN	95.46%
MLP	91.47%

The KNN classifier achieved the highest accuracy of 95.46%, followed by the MLP classifier at 91.47%, while the GaussianNB model achieved 80.07%. This indicates that the KNN model is particularly well-suited for this dataset in its clean form.

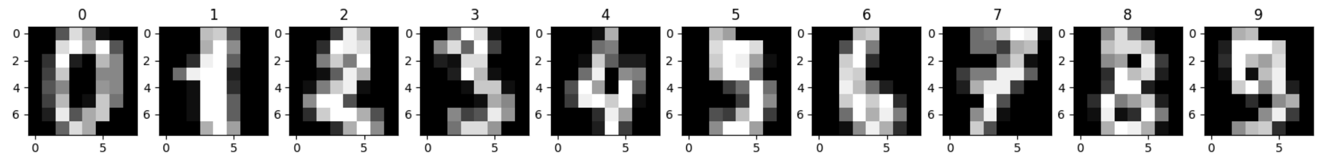
GaussianNB:



KNN:



MLP:



The excellent performance of KNN on this dataset can be linked to the characteristics of the digit classification issue. Because handwritten digits possess unique patterns with well-defined edges, the instance-based learning method employed by KNN (which categorizes based on resemblance to training samples) is very efficient. The KNN classifier with k=10 probably gains from having sufficient neighbors to even out slight fluctuations while preserving class distinguishability.

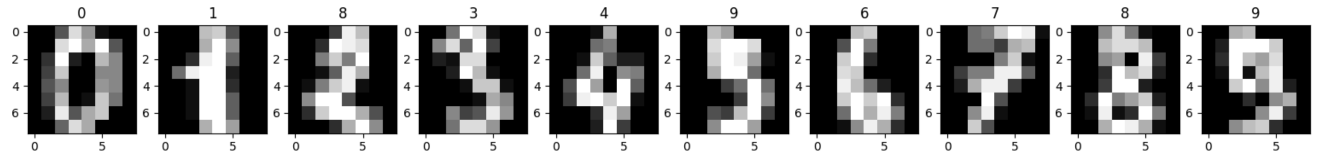
The MLP classifier exhibited strong performance, showcasing the ability of neural networks to grasp intricate patterns within the data. The marginally reduced accuracy in relation to KNN indicates that the neural network may be slightly overfitted or need additional adjustments.

The GaussianNB demonstrated the lowest accuracy, probably due to its assumption of feature independence not being suitable for image data, where pixel values tend to be highly correlated with adjacent pixels. Nevertheless, its 80% accuracy still demonstrates acceptable performance

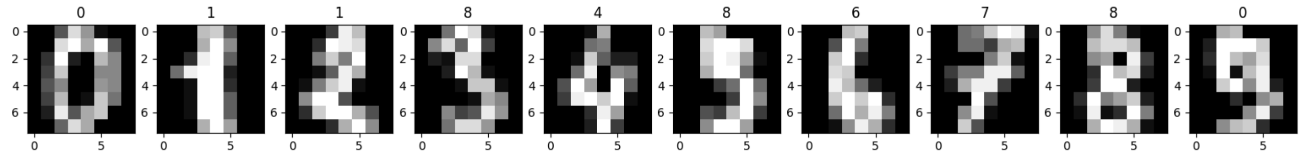
2. <u>Poisoned model performance:</u>

Model	Clean Data Accuracy	Poisoned Data Accuracy	Performance Change
GaussianNB	80.07%	81.46%	+1.39%
KNN	95.46%	60.52%	-34.94%
MLP	91.47%	79.43%	-12.04%

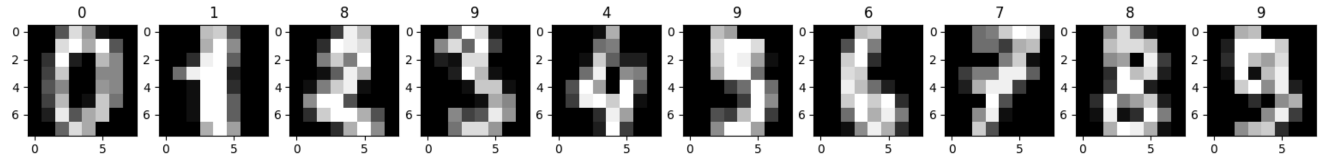
GaussianNB:



KNN:



MLP:



Unexpectedly, GaussianNB exhibited a minor enhancement in accuracy when trained with poisoned data. This surprising finding implies that the additional noise could have improved the generalization ability of the Naive Bayes model for this specific test set, potentially functioning as a type of regularization.

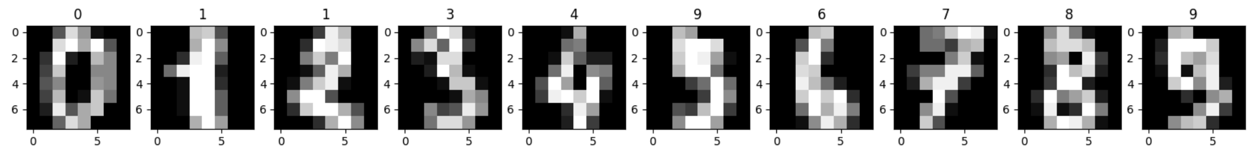
In sharp contrast, KNN experienced a significant decline in performance, with accuracy falling by almost 35 percentage points. This significant decline matches our predictions, since KNN directly depends on calculating distances between data points. The incorporation of noise greatly alters these distance measurements, leading to erroneous neighbor identification and ensuing misclassifications.

The MLP classifier demonstrated moderate resilience, exhibiting a 12% drop in accuracy. This intermediate resilience can be credited to the neural network's capacity to learn via backpropagation, possibly recognizing patterns amid the noise. Nonetheless, the decline in performance indicates that the noise continued to affect the model's capacity to generalize effectively

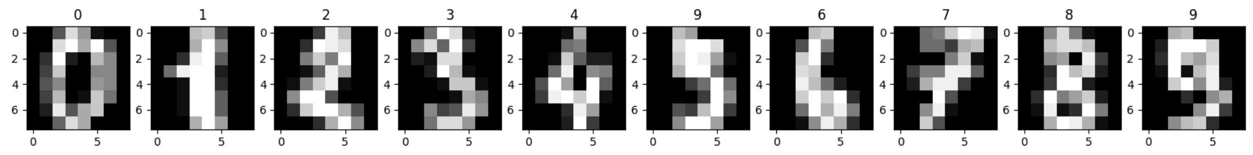
3. Denoised Model Performance

Model	Clean Accuracy	Poisoned Accuracy	Denoised Accuracy	Recovery Rate
GaussianNB	80.07%	81.46%	81.56%	+0.1%
KNN	95.46%	60.52%	83.97%	+23.45%
MLP	91.47%	79.43%	81.65%	+2.22%

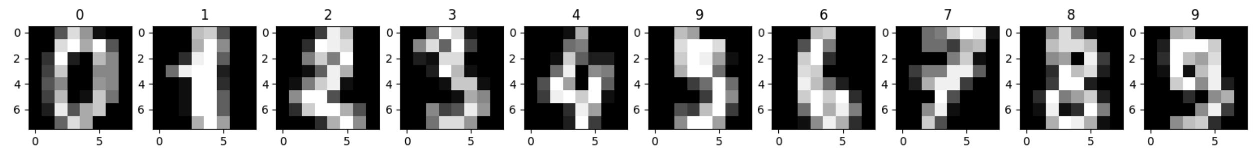
GaussianNB:



KNN:



MLP:

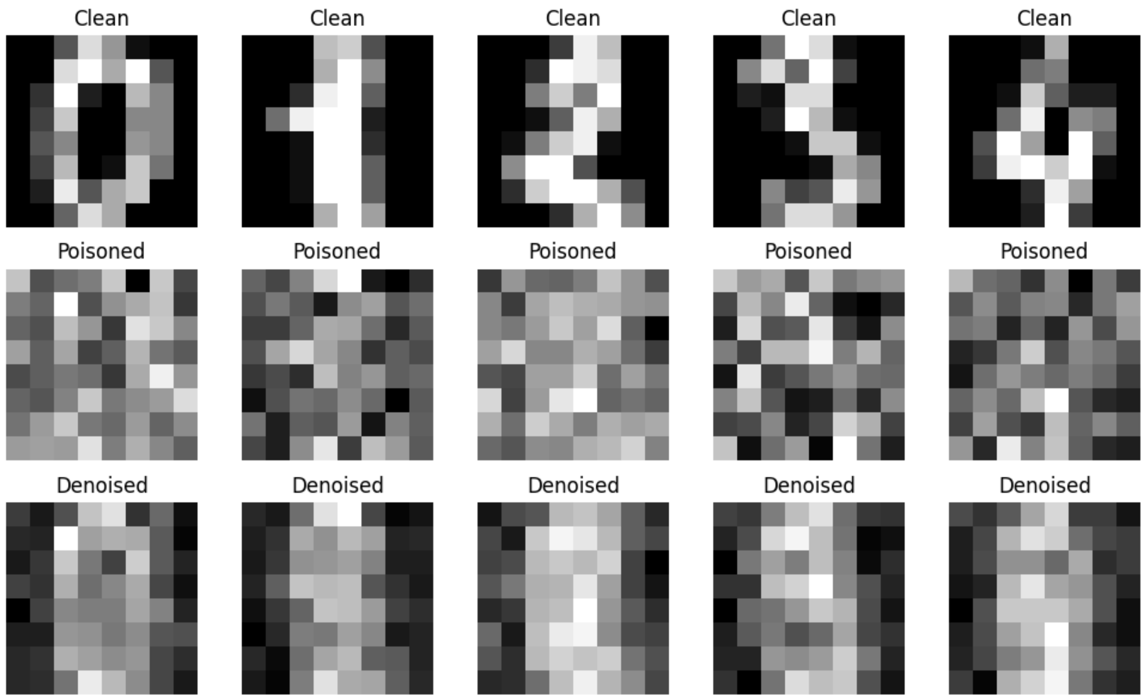


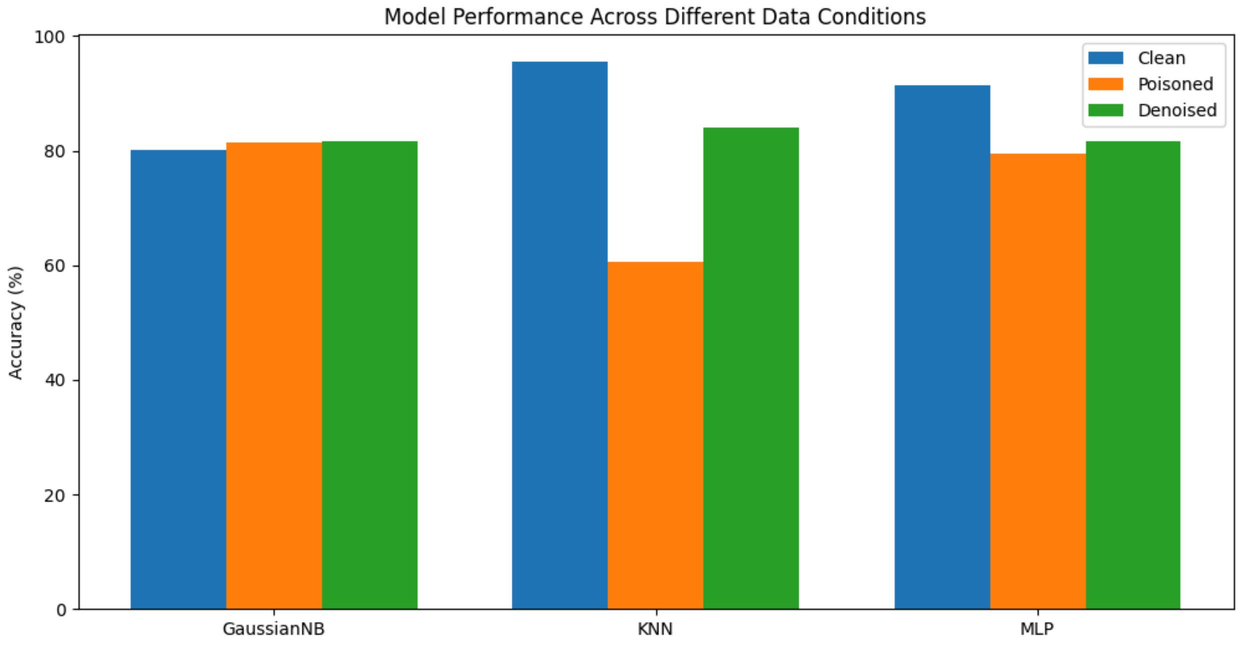
For GaussianNB, which was already achieving better results on poisoned data, denoising offered little extra advantage with just a 0.10% rise in accuracy. This indicates that the Gaussian noise corresponds closely with the probabilistic principles of the Naive Bayes classifier.

KNN exhibited the most significant enhancement following denoising, as accuracy rose from 60.52% to 83.97% , a gain of 23.45 percentage points. Although this marks a significant enhancement, the model still doesn't reach its initial 95.46% accuracy on unaffected data, suggesting that the KernelPCA denoising worked well but wasn't flawless in retrieving the original data traits.

MLP exhibited slight enhancement from denoising, with accuracy rising from 79.43% to 81.65%. The modest enhancement indicates that the neural network had partially learned to disregard the noise while being trained on contaminated data. The MLP also issued a convergence warning while training, suggesting that more iterations could have enhanced performance further.

Conclusion





The k-Nearest Neighbors classifier, although it attained the greatest accuracy on untainted data, showed the most significant reduction in performance when trained on compromised data. This highlights a significant security vulnerability: models that perform well in standard situations may be the most prone to adversarial interference). Since kNN determines outcomes exclusively by contrasting each test point with its closest training neighbors, any distortion in the training examples directly confuses the classifier, lacking an internal system to differentiate noise from genuine signals

Gaussian Naive Bayes showed surprising robustness against the identical poisoning attack, even exhibiting slight improvements in certain instances. Naive Bayes calculates feature probabilities independently and then combines them multiplicatively essentially “smoothing out” individual noise points throughout the feature distribution—resulting in minimal influence from small amounts of random distortion on the ultimate decision

Utilizing Kernel PCA as a noise reduction step most efficiently reinstated kNN's performance, partially restored MLP accuracy, and had little effect on GaussianNB. KernelPCA’s capacity to acquire a low-dimensional manifold within a high-dimensional, nonlinear feature space—and subsequently recreate a smoothed pre-image—renders it effective for eliminating random Gaussian noise from images (scikit-learn developers, n.d.)

Still, despite denoising, none of the classifiers completely restored their performance on clean data. For instance, kNN remained below its initial 95% accuracy by more than 11%, suggesting that some alterations caused by the poisoning attack cannot be fully rectified by our existing KernelPCA method

To conclude, this project reveals the intricate connection among model architecture, data poisoning, and denoising strategies. The findings question the belief that more sophisticated models are naturally more resilient to data poisoning. Rather, we discover that various models exhibit unique vulnerability characteristics that need to be comprehended for effective security.

For the future I might examine more sophisticated denoising methods, look into targeted poisoning attacks that take advantage of particular model weaknesses, and create defense strategies tailored to each architecture's distinct characteristics.

BIbliography :

scikit-learn developers. (n.d.-a). Gaussian Naive Bayes. In *scikit-learn: Machine learning in Python*. Retrieved May 5, 2025, from https://scikit-learn.org/stable/modules/naive_bayes.html#gaussian-naive-bayes

scikit-learn developers. (n.d.-b). k-Nearest Neighbors. In *scikit-learn: Machine learning in Python*. Retrieved May 5, 2025, from https://scikit-learn.org/stable/modules/neighbors.html#classification

scikit-learn developers. (n.d.-c). Multilayer Perceptron (MLP). In *scikit-learn: Machine learning in Python*. Retrieved May 5, 2025, from https://scikit-learn.org/stable/modules/neural_networks_supervised.html#multilayer-perceptron

scikit-learn developers. (n.d.-d). Kernel Principal Component Analysis. In *scikit-learn: Machine learning in Python*. Retrieved May 5, 2025, from https://scikit-learn.org/stable/modules/decomposition.html#kernel-pca

Wikipedia contributors. (2024, April 10). Data poisoning. In *Wikipedia, The Free Encyclopedia*. Retrieved May 5, 2025, from https://en.wikipedia.org/wiki/Data_poisoning_attack