ECE 20875 Mini Project Report

Hassan Alghanim
halghani
0034903514

Kasidej Chinpattanakul
kchinpat
0038243606

Mini project repository's github link: https://github.com/kchinpat/ECE20875-MiniProject-Path3

Path 3: Data Security in Model Training

Dataset Description

The dataset consists of 8 by 8 images of handwritten digits sourced from the sklearn library. The dataset has 1797 instances comprising 10 classes, where each class is a number from 0 to 9. Each instance in the dataset has 64 attributes, being each pixel within the image, and each attribute is assigned a shading level from 0 to 16, where 0 is white and 16 is black. A higher level of darkness corresponds to a higher prominence of the line at that point.

GaussianNB Model

The Gaussian Naive Bayes model (GaussianNB) determines the properties of a given class by calculating a mean and variance value for each feature of the instance within the training set. Once this has been performed for every instance within every class of the training set, a group of normal distributions is formed, with the probability of a feature having a certain shading level given it is within a certain class. A GaussianNB model was chosen over a typical Naive Bayes model because the data provided for each of the features is more appropriately treated as continuous data. For example, training data within the class "7" may contain shading levels of [12, 10, 8, 14, 14, 15, …] for the coordinate (2, 6) across various entries. Therefore, the model should consider a weighted probability, where the likelihood that a dataset falls within a given class is the likelihood that a coordinate has a shading level n, given that it is in the class m. Treating the data as if it was continuous rather than discrete is also essential to comparing the level of shading even if the datapoint is far from the mean and in cases where the training data has low variance. Force example, if the point (4, 4) for the class "8" has a high mean and low variance, the probability calculation would still take into consideration that a shading level of 7 is closer to the mean than a shading level of 6, which would yield more accurate results than a multinomial distribution where the likelihood of the shading levels 6 and 7 would be almost identical.

The GaussianNB model is also very computationally efficient as it can achieve low variance while making the assumption that the features are statistically independent. This also makes the GaussianNB useful for accepting or rejecting incorrect labeling because it is able to ignore redundant/irrelevant features rather than depending on covariance in the same way as the Gaussain Bayes model. Because of the additional information needed to calculate covariance, the dataset used with only 64 features would not be sufficient for the application of the Gaussian Bayes model (this is because covariance calculations would treat the data as having 64^2 features, and the provided dataset only has 1797 instances), which makes the GaussianNB a more appropriate choice for this analysis.

Based on the discussed properties of a GaussianNB model, it is expected that it will have an accuracy of around 80% due to multiple factors. Two significant factors are the overlap between the means and standard deviations of features in different classes which would lead to mislabeling, and the limited amount of information available due to the low number of features. Because of this, numbers that appear similar such as 7 and 1 or 6 and 0 are likely to cause inaccuracies, and since they are account for 40% of the classes, assuming the model is around 60% accurate when identifying them, and adding the errors for the other classes, the overall accuracy can be reasonably assumed to be 80%.

KNN Model

The K-Nearest-Neighbors (KNN) model identifies classes by determining the distance between randomly selected points, which is achieved by plotting each instance in a dataset as a point in an n-dimensional space, where each dimension corresponds to a feature of the instance. These points are already classed as they are labeled within the training set. When the model encounters a test, it plots it in the space with all the training inputs, and the classification of the test is based on a vote of the K instances closest to the test input. This vote method is more computationally efficient than the similar K Means Clustering method, as KNN does not require a calculation of the distance between the test input and all training inputs in the 64 dimensional space used to plot the points. Another advantage is that KNN requires very little time for training, as it must only store the pre-labeled data. The KNN method of comparing distances is effective for distinguishing between small differences in the inputs, such as between 3 and 8.

The expected accuracy of the KNN model is higher than the GaussianNB as it does not need to make assumptions about the shading level of a given feature being present within a given class. Because of this, the rate of misclassification is reduced. KNN also requires fewer comparisons to predict the class of a training input because it does not need to compute a probability for each class, although this is dependent on the number of training inputs used. Therefore, it is reasonable to expect the accuracy of the KNN model to be around 90%.

MLP Model

The Multi-Layer Perceptron (MLP) model which utilizes a neural network to iteratively change the weights and biases of its function. During training, an MLP model is self evaluated by computing the cost of training, which is the sum of squared error of the expected output compared to the model's output. The average of the cost of every neuron within the network is the total cost of the network. Once the total cost of the network is known, the negative gradient of the cost function is computed utilizing backpropagation, which identifies the changes to the network's weights and biases that would reduce cost, or in other words, increase the accuracy of the model. The size of the adjustments made are dependent on the difference between the confidence of the correct output, the correct label for the training input, and the level of confidence for the incorrect output. Assuming a normalized output, if an input with label 8 has a confidence level of 0.4 for class 8 and 0.9 for class 6, the weights from the previous layer which produced higher values for the training input, have the largest increase according to the expected output, which is 1. In order to further increase confidence, the neurons in the previous layer within the lowest values in reaction to the input have decreased weights, which further increases the confidence in the output the next time a similar training input is provided. A change to the activated neurons is also made proportional to the change in weight. Similar to the output neuron, this process is applied recursively to the neurons in the previous layers to iteratively train the model to be more accurate.

The MLP model is appropriate for predicting digits because it identifies patterns more complex patterns based on neuron activation patterns. For example, when distinguishing between a 2 and a 3, the MLP would identify that an additional group of neurons is consistently firing when the input is a 3 compared to the 2, because of the additional stroke that is made. Such distinctions allow an MLP to be highly accurate when classifying test inputs (expected to be 90%-95%), even more so in cases where there is a very large number of features in the input, although the process is very computationally expensive. The MLP is also dependent on the quality of the input, which will likely cause a substantial decrease in accuracy when the noise is introduced to the data. The accuracy of the MLP after the data is likely to increase to around 80% depending on the quality of the denoising. This is because of the relatively low number of features, the denoising will be less effective in restoring the image, and a high degree of overlap in the test inputs will decrease the accuracy of the model.

Objective

This project's goal is to evaluate the performance of the GaussianNB, KNN, and MLP models when encountering clean, noisy, and denoised data.

Code explanation (Step by step implementation)

Part 1-2: Data Extraction and Visualization, we implemented two helper functions:

1. dataset_searcher(): Fetches all image samples for selected digit classes.
2. dataset_searcher_one_each(): Fetches one representative image per digit for display

Image from digit classes [2,0,8,7,5] and [0-9] were visualized for sanity check.

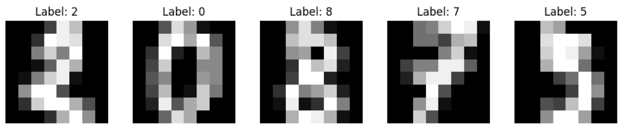


Figure 1: Display of the class [2,0,8,7,5]

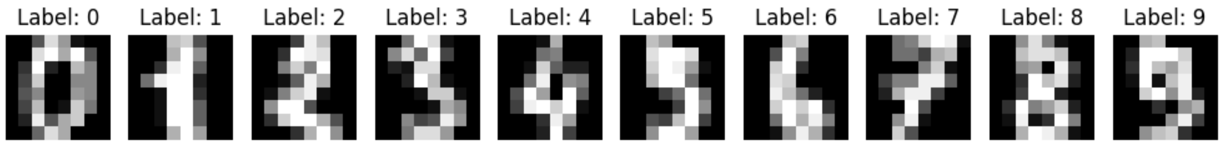


Figure 2: Display of the class [0,1,2…,9]

Part 3-6: Model Training and Evaluation on Clean Data Three classifiers were trained on clean, reshaped image data:

- GaussianNB
- KNN (k = 10)
- MLP (max iteration = 500)

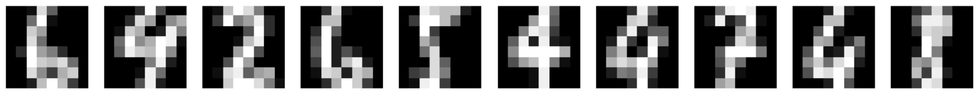


Figure 3: Example of clean data

Part 8-11: Impact of Poisoned Data Gaussian noise (scale = 10.0) was added to the training and test data to simulate poisoned conditions. All three models were retrained using this noisy data.

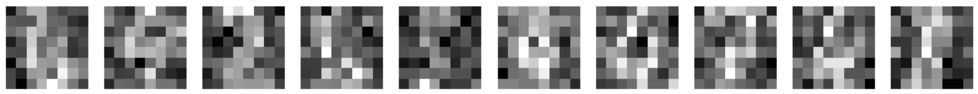


Figure 4: Example of Poisoned (Noisy) data

Part 12-13: Denoising with Kernel PCA To mitigate poisoning effects, we applied Kernel PCA to denoise the training and test datasets.

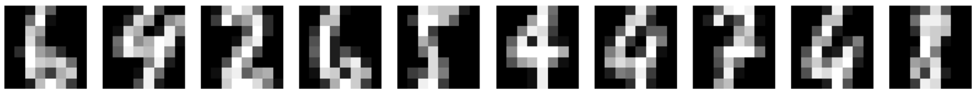
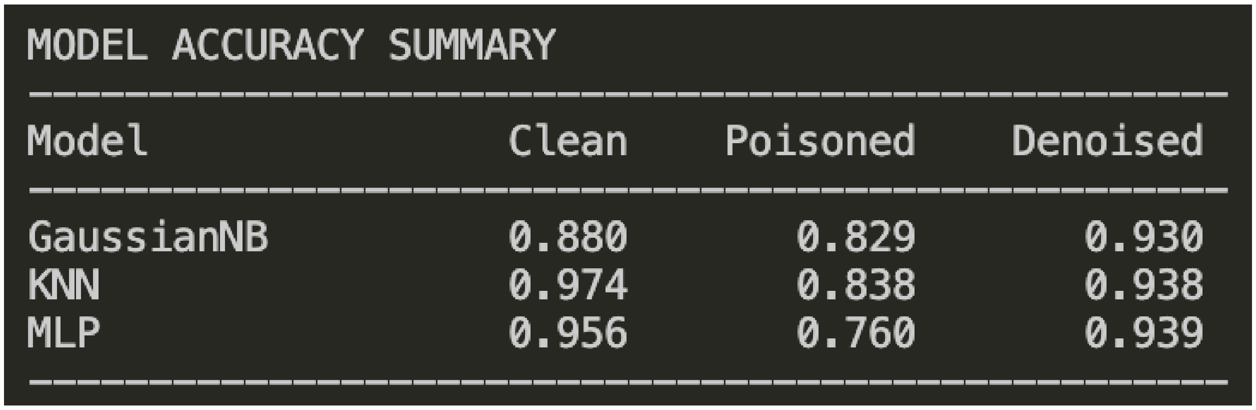


Figure 5: Example of Denoised Data

Part 14-15: Visualization and Summary, Prediction results of each model under all three scenarios were visualized side-by-side. This provide insights into how models interpret images under clean vs noisy and vs denoised conditions.



```
MODEL ACCURACY SUMMARY
--------------------------------------------------
Model                 Clean    Poisoned    Denoised
--------------------------------------------------
GaussianNB            0.880       0.829       0.930
KNN                   0.974       0.838       0.938
MLP                   0.956       0.760       0.939
--------------------------------------------------
```

Figure 6: Example of Formatted Output

When run the program the expected output will be consisting of 11 Figures including

2 Samples, 3 of clean data, 3 of poisoned data, and 3 of denoised data.

Then the program will also analyze the accuracy of each of the models respectively with the quality of image.

Result/Calculations

The function of this program is to test the performance of three classification models, being the GaussianNB, the KNN, and the MLP, which utilize different approaches to make predictions. This is clearly observed in the change to accuracy made when they encounter the poisoned data, and the degree of improvement seen when the data is denoised. The overall goal is to evaluate the strengths and weaknesses of the model, and this was achieved by running the program 10 times and taking the average of their accuracy scores to ensure a fair comparison.

Clean Data Analysis

Clean data is the undistorted input, which in most cases, yields high accuracy scores from the models. This is because the digits in this dataset are clear and show more distinct features, allowing the models to classify them more easily.

Test #	GaussianNB	KNN	MLP
1	0.847	0.966	0.955
2	0.846	0.977	0.968
3	0.851	0.973	0.954
4	0.826	0.969	0.957
5	0.85	0.975	0.963
6	0.85	0.966	0.955
7	0.86	0.973	0.951
8	0.827	0.973	0.692
9	0.854	0.97	0.965
10	0.816	0.968	0.956
AVG	0.8427	0.971	0.9316

After running the tests 10 times the MLP and KNN models performed the best. This is likely because of the nature of the data used, where the MLP and KNN were able to identify certain traits of the digits and reliably isolate their characteristics to make an accurate prediction. As mentioned within the model descriptions, the use of the vote system in the KNN model gave more weight to distinctive factors in the test input, and this was similarly present for the MLP. The GaussianNB model used a distribution to identify the likelihood of each feature to be present for a given class, which falls short because it doesn't take into account the covariance of the features, which is needed for an image representing a number.

Poisoned (Noisy) Data Analysis

Poisoned data is produced by adding random noise to images, causing the distinguishing factors of the inputs to become less identifiable, which substantially reduces the accuracy of all the models due to the substantial overlap in the training inputs. However, because the GaussianNB is not dependent on the covariance of features when predicting an output, it is the least impacted by the introduction of noise to the data. The KNN and MLP models, which rely on the covariance of features to classify an input were more impacted by the introduction of noise, which shows they are more dependent on the quality of the input data.

Test #	GaussianNB	KNN	MLP
1	0.55	0.412	0.422
2	0.583	0.444	0.458
3	0.58	0.421	0.411
4	0.578	0.438	0.414
5	0.565	0.418	0.456
6	0.557	0.424	0.42
7	0.576	0.411	0.447
8	0.563	0.427	0.433
9	0.561	0.416	0.442
10	0.553	0.441	0.431
AVG	0.5666	0.4252	0.4334

Table 2: Result of poisoned data accuracy (10 iterations)

Denoised Data Analysis

Denoised data is data that has been filtered to reduce the significance of random error within it. This data remains distorted because the information provided to the filter used limits the image that can be resolved. It should be noted that the level of noise introduced within this dataset is far greater than the amount typically encountered by these models, which further demonstrates their resilience to noise.

Test #	GaussianNB	KNN	MLP
1	0.804	0.698	0.756
2	0.697	0.623	0.652
3	0.798	0.691	0.753
4	0.774	0.634	0.746
5	0.793	0.684	0.743
6	0.797	0.594	0.668
7	0.768	0.646	0.749
8	0.773	0.629	0.742
9	0.818	0.708	0.768
10	0.805	0.746	0.766
AVG	0.7827	0.6653	0.7343

Table 3: Result of Denoise data accuracy (10 iterations)

The average accuracy for all three models improved after denoising with KernelPCA, as expected. The GaussianNB showed the best performance, with a level of accuracy closer to the clean data. This can be considered another demonstration of the model's resilience to noise because it does not rely on covariance. Because the MLP increased in accuracy, it can also be determined that the denoising was effective, as it is more dependent on the quality of data to produce higher accuracy outputs. The model that showed the least improvement was the KNN, as it is most dependent on the quality of the data.

Conclusion

In conclusion, KNN has the best performance when the data is clean, although it was most impacted by the introduction of noise to the data due to it's application of covariance when classifying a test input. Once the data was denoised it showed a smaller improvement compared to the other models as it is most dependent on data quality.

The GaussianNB model was the most resilient to the introduction of noise, although it also has the highest variance even when clean data is input. This suggests that this model would be best for classifying data where the quality of the input may be lower. The dataset used may be less appropriate for this model as accurate classification is highly dependent on the detection of hierarchical patterns such as lines and curves rather than the variance of a single feature, so the GaussianNB would be more appropriate for applications either with binary data, or those with fewer features.

The MLP was the most versatile model for this application, and likely the best option if digit prediction is used in a real world context. This is because this model is best used for the detection of hierarchical patterns, which is ideal for classifying elements of an image, and also because it is more adaptable due to the high number of configurations the neurons may be set to. While the model demonstrated relatively low performance when the data was denoised compared to the clean data, this issue may be addressed in two ways. Firstly, the quality of the denoising may not have been sufficient to provide the MLP with high quality information. This may be tested by altering the method used to denoise the data. Secondly, the MLP may still perform well with the noisy data, but it may require more neurons, and consequently, more training examples to optimize the cost of the model.

Overall, this project was successful in testing the performance of the different models, and identifying their strengths and weaknesses by analyzing the method through which each model makes predictions, and evaluating it in the context of the chosen dataset. Because image detection requires higher-level pattern recognition, or in other words, seeing the "big picture", it is fitting that the MLP, which is optimized for detecting such patterns, performed the best overall. This experiment could be extended upon by utilizing different denoising tools to evaluate their performance, and using a higher resolution image with the MLP to understand how the number of layers and the quantity and quality of information supplied determines its performance under clean, noisy, and denoised data.