The Best Model to Recognize Handwriting

Haeun Kim and Jeongjae Han

University of Massachusetts – Lowell. COMP.5450 Machine Learning

Abstract:

As technology develops, people are shifting from analog to digital. Recently, due to covid, most of the tasks required digital devices, and many people started taking notes with their computers or tablets. People prefer tablets because they can write with a pen which feels like taking notes on a paper. However, to make the handwriting neater, the tablet needs to recognize the writing with machine learning algorithms and convert that to a text. To convert the letters, machine learning algorithms are applied. For this research, boosting, NN, KNN, Naïve Bayes' Classifier, PCA, and SVM models were used to classify the inputs, ROC was utilized to visualize the result with, and standardization was applied to increase the accuracy of the model. For the result, SVM was the best classifier to recognize handwriting because 97.58% was the accuracy of this model which was the highest overall.

Keywords: ANN, KNN, Naïve Bayes, PCA, ROC, SVM, and Validation

1. Introduction

The way that humans and machines recognize the letters are similar. Humans perceive with the eyes, and process in the brain. Since the neurons can perform an elementary cognitive function, it converts a complex pattern of inputs into a simple decision while it is transferring the input [1]. At a whole-brain level, the neurons in several posterior brain regions are triggered strongly to recognize handwriting and left primary motor cortex and the supplementary motor area get activated and help to recognize it [2].

With the machines, there are many methods to recognize handwriting. Most of the time, neural networks are used because they can approximate functions and dynamics by learning from examples. It is inspired by neurobiology, so it works close to our brain. We give input image to the machine and neural networks are used to recognize the writing. The machine verifies the letter by feeding data through the hidden layers that the model produced, and computes for the results [1].

However, our handwriting dataset from UCI Machine Learning Repository is csv files with 16 attributes instead of image files. Therefore, instead of just using MLP and SLP, which are NN models, we utilized other models like KNN and SVM.

2. Background

A. Supervised learning/ unsupervised learning

In machine learning, based on supervised learning and unsupervised learning, different models are used. While supervised learning is a machine learning algorithm training the dataset with labeled specific output like classification or regression, unsupervised learning is also machine learning algorithm with unlabeled dataset like clustering. My dataset is labeled dataset, this article mostly covers supervised learning algorithm, the last chapter of this article will cover one of the popular unsupervised learning algorithms.

B. NN

An artificial neural network is formed with perceptron. This model has an input layer and an output layer; figure 1 demonstrates it. The computation of a single layer perceptron is performed over the calculation of the sum of the input vector each with the value multiplied by the corresponding element of the vector of the weights, and the value gets computed by the activation function. To visualize the model, Logistic Regression is often used.

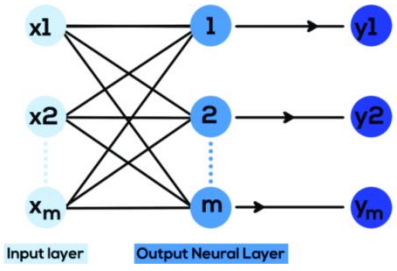


Figure 1. Single-Layered Feed-Forward Network [1].

The model attains the result by inserting input signals: {x1, x2, xm} into the input layer, when signal goes to output layer, the data gets multiplied with weights: {w1, w2, wm}. Then, the data gets decided if the data will have an impression over the output. If it is, the result will be assumed as the weighted sum of inputs, and bias (θ) will be produced to generate a trigger at the end of the output by a linear aggregator (Σ). With the result and the bias, activation potential (u) is obtained, and it characterizes the shape of the output. Finally, the activation function (g(u)) mathematically limits the value of the output to a range of values to determine whether the result will be positive or not and produces the output signals (y). This procedure can be represented as.

$$y = g(\Sigma w_i * x_i - \theta)$$

For activation functions, there are many types: Linear Function, Sigmoid Function, Tanh Function, ReLu. Sigmoid function was utilized for this project. It is especially used when probability is to be predicted as an output because the range of value produced by this function lies between 0 and 1. The function can find the slope between two points since it is differentiable. Unfortunately, there is a chance that the model gets stuck during training. The formula is shown as

$$f(x) = \frac{1}{1 + e^{-x}}$$

Perceptron is mostly used for the classification of linearly separable patterns, and it is supervised learning and can also be a binary classifier. Therefore, the size of the data will not affect the result, the result comes out quickly after the training, and it is well used to prioritize the impact of selected features over the others, unlike a decision tree algorithm and the nearest neighbor algorithm. Since decision tree uses a small number of features and nearest neighbor algorithm gives priority to all the features equally. Therefore, we chose to use MLP, multi-layer perceptron, and SLP, single-layer perceptron. These two are both Neural Networks and the difference is the hidden layer's existence in the model.

a. SLP

Single-layer perceptron is an artificial neural network that is the first neural model created and neural network architecture's most basic form is perceptron without hidden layers; figure 1 is SLP. It is a feed-forward network and includes a threshold transfer inside the model. Due to a single layer, it only can learn linearly separable patterns, so cannot be trained to recognize multiple classes [3].

b. MLP

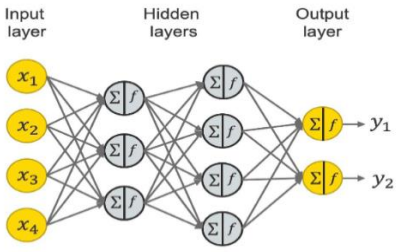


Figure 2. MLP Diagram [6].

Multi-layer perceptron is SLP with one or more hidden layers in between the input layer and output layer; figure 2 displays it. The perceptron in the layers iterates repeatedly to adjust the weighted and the threshold values to minimize the difference between the desired/targeted output and the obtained output; the equation for this model is

$$Z = \sum_{l=1}^{m} w_1 x_l + bias$$

The hidden layers allow the model to solve complex non-linear problems. Consequently, model differential and XOR functions are available with this model. Nonetheless, this model is computations are time-consuming and complex with many layers.

C. kNN

kNN, k-Nearest Neighbor, is an instance-based learning algorithm. When there is data and x_q is given, looks for the k numbers of nearest neighbors or takes the mean of f values of k numbers of nearest neighbors to predict the result; the equation is

$$f(x_q) \leftarrow \frac{\Sigma_{i=1}^{k} f(x_i)}{k}$$

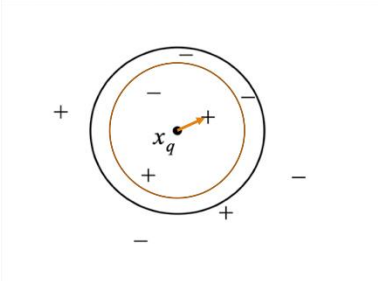


Figure 3. kNN Example [7].

According to figure 3, when k is 3, x_q is labeled as positive because there are two positive points and one negative point.

kNN is well used because it trains very fast, learns complex target functions, and does not lose the information, but testing for this model is slow and gets influenced by irrelevant attributes or outliers. Also, depending on the value of k, when k is small, the model might overfit, so patterns are detailed, but when k is large, it might underfit, so it has a stable classification. Consequently, it is important to use appropriate k to get the optimal results.

D. Naïve Bayes Classifier

Naïve Bayes' Classifier is the Bayesian with an assumption of independence among predictors. It tries to classify the new instance, so it can assign the most probable target value. The new instances described by the tuple of attribute values such as $< a_1, a_2, a_n >$ and the equation can be written as

$$v_{NB} = \arg \max_{v_j \in V} P(v_j) \Pi_i P(a_i | v_j)$$

When Bayes' assumption of conditional independence is satisfied, v_{NB} is identical to the MAP classification which is Maximum A Posteriori that allows us to incorporate prior belief and information.

Naïve Bayes' theorem is useful because it is easy and fast to predict a class of test datasets and even for multi-class prediction. When the assumption of independence holds, the classifier performs better at comparing other models like logistic regression because less training data is required. In contrast, if the input has a new category that was not in the training dataset, the result is going to be 0, but can be fixed by the smoothing technique. Also, the limitation of this model is the assumption of independent predictors because realistically, it is impossible to get a completely independent set.

E. SVM

SVM is the support vector machine. It is the model which defines a line for classification. So, unseen data is put, based on the line, which classifies this new data. We call this line decision boundary. The decision boundary is good when the distance between the data and the decision boundary are far. We call this distance margin. This margin is determined by the support vector. Support vector is data point where the margin maximizes.

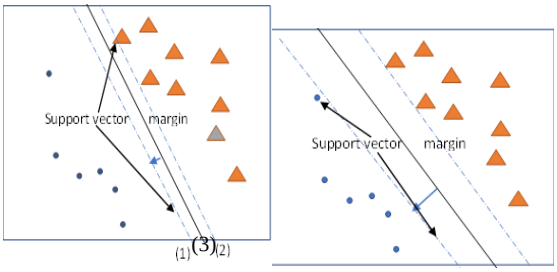


Figure 4. SVM Examples 1.

$$w^T x_i + b = -1 - (1)$$

$$w^T x_i + b = 1 \ - (2)$$

$$w^T x_i + b = 0 - \ (3)$$

D(distance)= $\frac{|w^T x_i + b|}{||w||}$ (w is a vector and orthogonal to the hyper plane b is a scale value(bias))

Margin = $\frac{2}{||w||}$

If the dataset is well recognized linearly, it means the line or plane is well operated, but not all data is not divided into the hyperplane in low dimension in this case, when some dataset is mapped into the high dimension, we can have more proper hyperplane (decision boundary), we call this method kernel trick

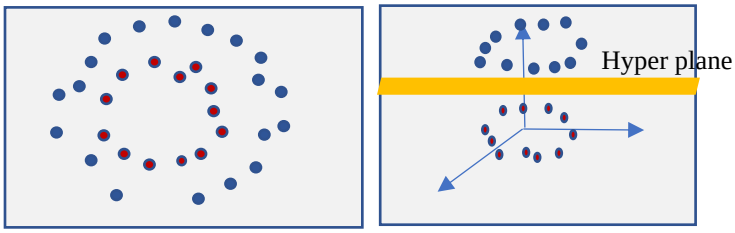


Figure 5. SMV example 2.

F. xgBoosting

Ensemble learning trains multiple learning algorithms and combines their output [4]. There are

three machine learning techniques: bagging, stacking, boosting. Bagging is a method where multiple models train the same dataset, and each result combines and makes an output. Stacking is that multiple models train a dataset for train and take the predicted output as input of the final model [5]. Boosting is one of the ensemble models combining more than two weak models (learners) sequentially and making a strong model. It weighs wrong predicted data so improves the accuracy of the model. Using the gradient descent updates the weight, ADAboosting weights the wrong prediction data. Unlike ADAboosting, gradient boosting weights use a gradient descent decreases the loss. But it has some disadvantages: overfitting, time-consuming. To solve this problem, XGboosting is developed.

G. Confusion Matrix

To show how this dataset is well classified, we used a confusion matrix, table 1.

	Predict		
Actual		Positive	Negative
	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Table 1. Confusion Matrix Example

In confusion matrix, for example. true positive means the case where among the predicted data are positive, the data are actual positive. True negative means the case where the predicted data are negative, and the data are actual negative. False negative means the case where among the predicted data are negative, but the data are actual positive. False positive means the case where among the predicted data are positive, but the data are actual negative. So, accuracy is the $\frac{TP+TN}{TN+FP+FN+TM}$

H. ROC-AUC

In an imbalanced dataset, accuracy is not the best way to show the performance, so we consider different methods to show the performance. The new method is ROC-AUC curve. To know ROC (Receiver Operating Characteristic curve) and AUC (Area under the Curve) have good performance. Whereas specificity is the true negative / (true negative + false positive). FP is the wrong prediction, so Low FP means low error and it means good performance. Therefore, high specificity is low FP and good performance. In roc curve, X axis represents the 1- specificity, and y axis sensitivity. 1-Specificity is the ratio where negative case is predicted as positive. AUC is the area under the roc curve. When the roc curve is top and left, it is a good model, and AUC is closer to 1, it shows good performance. If AUC is 0.5 the model's accuracy is pretty low, and this is not a proper prediction model.

I. PCA

Previous models have every feature as input, but if the number of features is large, it decreases the performance, we call it curse of dimension. When the dataset is mapped into the graph, if the feature increases, the dimension also increases, and the graph has sparse data points. Sparsity can make errors. So, if the dimension is big, users must decrease the dimension, to prevent curse of dimension. We cover one of the decreasing dimension algorithms called PCA.

Principal Component Analysis (PCA) is an unsupervised learning algorithm reducing the complexity of datasets while preserving data covariance. PCA creates new data set with relationship between features. It transforms high dimension dataset into low dimension dataset keeping the variance of the dataset to prevent the loss of dataset. This variance is determined by the eigen value, and covariance shows the relationship, if the covariance >0 when one dataset increases, the other dataset also increases. Principal component is the component representing the variance well. Therefore, the principal component is the value who has the largest variance and largest eigen value.

$$X = \begin{pmatrix} x1 \\ x2 \\ x3 \\ . \\ . \\ . \\ x16 \end{pmatrix}$$

x is the value (k column - mean of k column where k=1,2,3…n, n is the number of features)

$$X^{\mathrm{T}}X = \begin{pmatrix} x1 \\ x2 \\ x3 \\ . \\ . \\ . \\ x16 \end{pmatrix} (x1 \;\; x2 \;\; x3 \;\; \ldots x16)$$

X^{T} means transpose of x

For example, Matrix A = $\begin{pmatrix} 3 & 4 \\ 5 & 6 \end{pmatrix}$, A^{T}=$\begin{pmatrix} 3 & 5 \\ 4 & 6 \end{pmatrix}$ that is A=$a_{i,\,j.}$ A^{T} =$a_{\,j\,,i}$

$COV(X)/N$

$$= \begin{pmatrix} dot(x1,x1) & dot(x1,x2) & dot(x1,x3)\ldots & dot(x1,x16) \\ dot(x2,x1) & dot(x2,x2) & dot(x2,x3)\ldots & dot(x2,x16) \\ dot(x3,x1) & dot(x3,x2) & dot(x3,x3)\ldots & dot(x3,x16) \\ & & . & \\ & & . & \\ & & . & \\ dot(x16,x1) & dot(x16,x2) & dot(x16,x3)\ldots & dot(x16,x16) \end{pmatrix}$$

Dot operation is an inner product. for example,

A = (1,2,3), b= (3,5,7)

dot (a, b) = (1*3+2*5+3*7) = (3+10+21) = 34

To find Λ(lambda)

Det(X-ΛI) = 0 (Λ is eigenvalue)

J. Standardization

Standardization is one of scaling techniques where the values are centered around the mean with a unit standard deviation; the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation. The formula for standardization is

$$X' = \frac{(X - \mu)}{\sigma}$$

μ: The mean of the feature values.

σ: The standard deviation of the feature values.

Standardization can be helpful in cases where the data follows a Gaussian distribution, but it does not always have to be true, so when there are outliers in the data, they will not be affected by standardization.

3. Methodology/ Proposed Method

For this project, Jupyter Lab was used to compile the program and the language was python. Therefore, scikit learn package was used for all the models because the package has all the functions to build the models. Also, the ratio for training and testing for all the models is 8:2.

For ROC-AOC curve, ROCAUC library from yellowbrick package was brought to draw the curve easily.

A. SLP Model

For SLP model, the result was displayed as a confusion matrix. There are two results: one without normalization and with normalization. Standardization was applied to check if there is a significant role in normalization. For the result, ROC-AUC curve was activated.

B. MLP Model

For this model, 8 hidden layers were assigned, and logistic regression was applied for the activation function. Then, standardization was enforced to increase accuracy and like SLP model, ROC-AUC was used to visualize the result.

C. Naïve Bayes' Classifier

This Classifier utilized GaussianNB library was used because it is a probabilistic classification algorithm that is Bayes' theorem with strong independence assumption. For normalization, standardization happened to the model. To display the result, ROC-AUC was applied.

D. SVM

This model is not divided with hyperplane or line, so kernel trick is used. If gamma value is high, it makes a complex model and if gamma value is low, it makes simple model. If c value is high, it classifies dataset correctly and if c value is low, the shape of decision boundary is smooth or linear

E. KNN

This model was trained by KNN algorithm, displayed the relationship between the accuracy and k value, and ROC curve was applied.

F. xgBoosting

xgBoosting has enormous parameters to train the dataset. This model uses gradient descent to decrease the loss. This algorithm prevents overfitting and time consuming. It offers an early stopping parameter. SoftMax is used for classification, the learning rate is 0.,1 max depth is 5, early stoppings are 10 and it classifies 26 alphabet characters.

G. PCA

This model decreases the dimension using the covariance between features to prevent curse of dimension. To find the proper number of dimensions, screen plot was used. 4 components were chosen because the point of the slope of the graph is smooth.

4. Experiments/ Results

A. SLP Model

According to figures 6 and 7, the accuracies were very low even with normalization. ROC graphs did not have big differences either. The reason was that the dataset has too many attributes and SLP has a single layer, so the model cannot run complex datasets.

```
#without standarization
from sklearn.linear_model import Perceptron
perceptron = Perceptron(random_state=42,
                        max_iter=10,
                        tol=0.001)
perceptron.fit(x_train, y_train)


print(perceptron.score(x_test, y_test))
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(perceptron, x_test, y_test)
plt.show()
```

```
C:\Users\haeun\anaconda3\lib\site-packages\sklearn\linea
of iteration reached before convergence. Consider increa
  warnings.warn(
C:\Users\haeun\anaconda3\lib\site-packages\sklearn\utils
precated; Function `plot_confusion_matrix` is deprecated
sionMatrixDisplay.from_predictions or ConfusionMatrixDis
  warnings.warn(msg, category=FutureWarning)
```

```
0.43025
```

Figure 6. SLP Accuracy Before Normalization.

```
sc = StandardScaler(copy=True, with_mean=True, with_std=True)
x_sctrain = sc.fit_transform(x_train)
x_sctest = sc.fit_transform(x_test)
```

```
perceptron.fit(x_sctrain, y_train)


print(perceptron.score(x_sctest, y_test))
```

```
0.5155
```

Figure 7. SLP Accuracy After Normalization.

B. MLP Model

This model's accuracy was very low. Even after scale, the accuracy was 75% which means this model is not valid. To figure out the problem, a research about MLP and letter recognition was found, and its' accuracy was 91% [8]. Then, we modified the model by increasing the size of the hidden layer and changing the activation function. Chaning the size of the hidden layer was the key to increase the accuracy, but as the size got bigger, the time-consuming increased drastically. As a conclusion, it was not an efficient model.

```
mlp = MLPClassifier(hidden_layer_sizes=(10,), activation='logistic'
                    solver='sgd', alpha=0.01, batch_size=32,
                    learning_rate_init=0.1, max_iter=500)

mlp.fit(x_train, y_train)
print(mlp.score(x_test, y_test))
```

```
0.63075
```

Figure 8. MLP Accuracy Before Normalization.

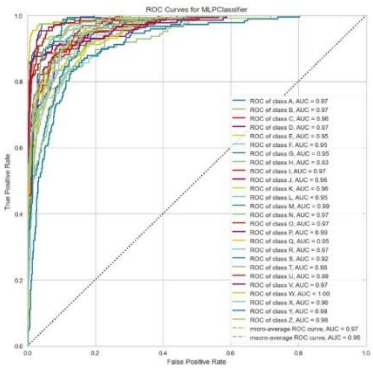


Figure 9. ROC-AUC Before Normalization.

```
In [34]:
scaler = StandardScaler(copy=True, with_mean=True, with_std=True)


x_sctrain = scaler.fit_transform(x_train)
x_sctest = scaler.fit_transform(x_test)
```

```
In [35]:

mlp.fit(x_sctrain, y_train)
print (mlp.score(x_sctest, y_test))

```

```
0.75325
```

Figure 10. MLP Accuracy After Normalization.

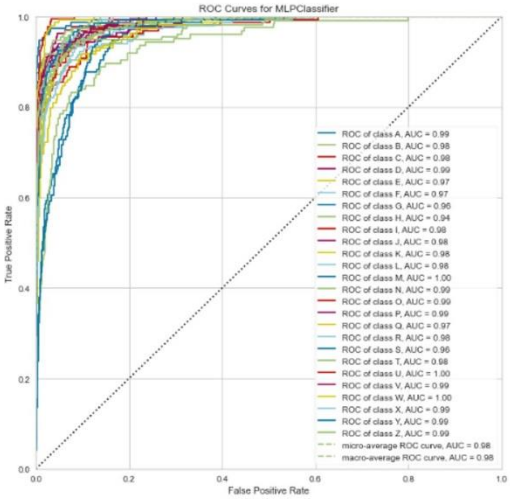


Figure 11. ROC-AUC After Normalization.

```
mlp = MLPClassifier(hidden_layer_sizes=(100,), activat

mlp.fit(x_sctrain, y_train)
print (mlp.score(x_sctest, y_test))

0.92175
```

Figure 11.a. MLP After Modification.

C. Naïve Bayes' Classifier

This Classification was not useful for handwriting recognition because the accuracy of this model is only 64.85% and even with normalization the accuracy went down abnormally because this classifier is based on probability instead of the distances.

```
In [8]:
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB().fit(x_train, y_train)
predictions = gnb.predict(x_test)

# accuracy on X_test
accuracy = gnb.score(x_test, y_test)
print(accuracy)

0.6485
```

Figure 12. NB Before Normalization.

```
In [14]:
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler(copy=True, with_mean=True, with_std=True)


x_train = scaler.fit_transform(x_train)
x_test = scaler.fit_transform(x_test)

In [16]:
gnb.fit(x_train, y_train)
predictions = gnb.predict(x_test)

# accuracy on X_test
accuracy = gnb.score(x_test, y_test)
print(accuracy)

0.64575
```

Figure 13. NB After Normalization.

D. SVM

SVM resulted in the highest accuracy and the curve was the most ideal among other models. Its' accuracy was 97.58% even without normalization. This came from the characteristic of SVM. It tends to minimize the structural risk instead of empirical risk used in the learning discipline of classic methods, so it generates better performance [9].

```
In [17]:
prediction=svm.predict(x_test)
svm.score(x_test, y_test)

Out[17]: 0.97575
```

Figure 14. SVM Accuracy

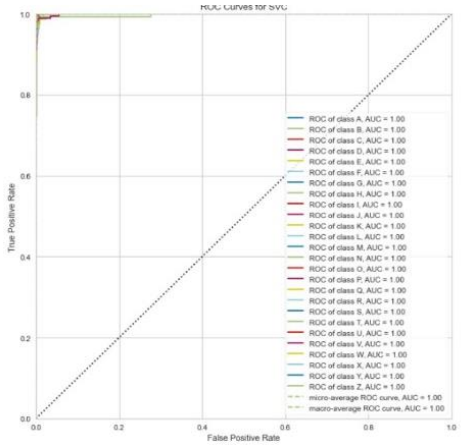


Figure 15. SVM ROC-AUC Graph

E. KNN

This classifier was the third most accurate. According to figure 16, as the k value decreased the accuracy decreased because of underfitting. Therefore, k value was assigned as three because when k = 3, the accuracy was the highest.

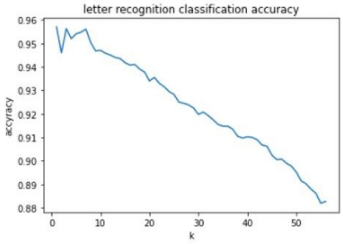


Figure 16. KNN Accuracy vs k value.

```
In [29]:
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(x_train, y_train)
print(knn.score(x_test, y_test))

0.95625
```

Figure 17. KNN Accuracy When k = 3.

F. xgBoosting

The Accuracy of xgBoosting was the second highest: 96.05%

```
xgb_model = xgb.XGBClassifier(params)


xgb_model.fit(x_train, y_train)

print(xgb_model.score(x_test,y_test))
```

```
0.9605
```

Figure 18. xgBoosting Accuracy.

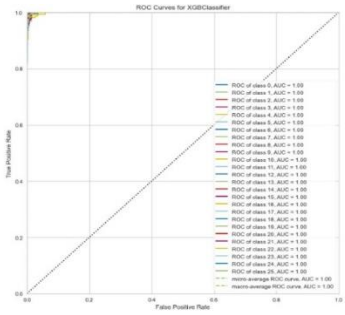
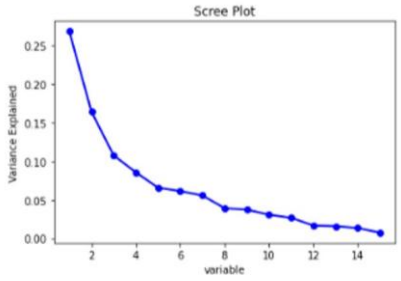


Figure 19. xgBoosting ROC-AUC.

G. PCA

This model decreases the features. To determine how many features are used we use screen plot. In scree plot, when the variable is 4 the slope of graph is smooth. The sum of the four components is 0.64. It means these 4 components explained 64% of total variance. Using the PCA, the new dataset has 4 features, and this dataset can be used to train a machine learning model.




```
[0.26846009 0.16409742 0.10757239 0.08557186 0.06571382 0.06125425
 0.05558286 0.039111   0.03721695 0.03074366 0.0266375  0.01663726
 0.01587223 0.01340897 0.00743974]
```

Figure 20. PCA Screen Plot.

```
:  pca.explained_variance_ratio_
: array([0.2905623 , 0.15487765, 0.11625481, 0.08682245])
```

Figure 21. PCA Variance Ratio.

	pca_comp1	pca_comp2	pca_comp3	pca_comp4
0	-10.179766	4.779905	3.828877	-1.562944
1	-5.992342	-7.238026	1.671875	1.948644
2	-0.703867	3.653414	0.688039	2.687970
3	1.815223	-2.926889	1.733275	-0.007666
4	7.454277	-0.111523	-3.566622	-3.702580
...	...	...	...	...
3995	1.947687	3.263570	-0.416978	3.176136
3996	1.671012	-3.140165	-2.894346	0.799008
3997	7.917657	-1.867193	4.473090	-2.794370
3998	1.456951	-0.664148	-1.580093	7.137927
3999	5.294825	-0.398287	4.373007	-1.767523

4000 rows × 4 columns

Figure 22. PCA Data Frame.

Many different classifiers have been tested. There was an unexpected result like MLP because the accuracy of the model was too low for the model. Despite that, this project concluded that the best model for writing recognition was SVM, the second was xgBoosting, and the third was KNN. There was a research that had the same result that SVM was faster and more accurate [10], so it confirmed that SVM is better than kNN.

5. References

[1] Cezary Biele, Cezary Biele, Hand Movements Using Keyboard and Mouse, Human Movements in Human-Computer Interaction (HCI), 10.1007/978-3-030-90004-5_4, (39-51), (2022). https://onlinelibrary.wiley.com/doi/full/10.1002/hbm.21105?casa_token=W-alcs-viUAAAAAA%3AMpm9s9_uFbaJFKQY2njuonwsDY5JDph-sZQ_EeTxm2WtqvGMdGOvBmYZj3O2Gozj6ATgOwmtNUv9Ediy

[2] Nikolaus Kriegeskorte, Tal Golan, Neural network models and deep learning, Current Biology, Volume 29, Issue 7, 2019, ISSN 0960-9822, https://doi.org/10.1016/j.cub.2019.02.034.

[3] J. Singh and R. Banerjee, "A Study on Single and Multi-layer Perceptron Neural Network," *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, 2019, pp. 35-40, doi: 10.1109/ICCMC.2019.8819775. https://ieeexplore.ieee.org/abstract/document/8819775?casa_token=cNNUrDSsvxoAAAAA:IolnvA1Yeu

Xgl0hJcVOT_6jsvsmMREPboCa-G5Jn1sa9O7w7EaJS-Yh0BJhDuogC1lvfw_hMfw

[4]G. Brown, "Ensemble Learning," in Encyclopedia of Machine Learning, C. Sammut and G. I. Webb Eds. Boston, MA: Springer US, 2010, pp. 312-320.

[5]M. Shahhosseini, G. Hu, and H. Pham, "Optimizing ensemble weights and hyperparameters of machine learning models for regression problems," Machine Learning with Applications, vol. 7, p. 100251, 2022/03/15/ 2022, doi: https://doi.org/10.1016/j.mlwa.2022.100251.

[6] R, Ma, “08 Artificial Neural Networks”, COMP.5450 Machine Learning, Fall 2022. COMP.5450 Machine Learning, 2022.

[7] R, Ma, “06 Instance Based Learning”, COMP.5450 Machine Learning, Fall 2022. COMP.5450 Machine Learning, 2022.

[8] Q. Feng and G. Daqi, "Dynamic learning algorithm of multi-layer perceptrons for letter recognition," *The 2013 International Joint Conference on Neural Networks (IJCNN)*, 2013, pp. 1-6, doi: 10.1109/IJCNN.2013.6706896.

[9] C. Zhaolong and W. Fuyong, "SVM Arithmetic and It's Application in Many Species Letter Image Recognition," *2006 Chinese Control Conference*, 2006, pp. 1862-1866, doi: 10.1109/CHICC.2006.280873.

[10] P. Ghadekar, S. Ingole and D. Sonone, "Handwritten Digit and Letter Recognition Using Hybrid DWT-DCT with KNN and SVM Classifier," *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, 2018, pp. 1-6, doi: 10.1109/ICCUBEA.2018.8697684.

6. Task Distribution

Overall, we contributed equally to this project. For the code, we both worked on it together. Whenever we face errors, we both worked on it separately, and once one person found the solution, it was shared, so we both have same code. Like this way, we built 7 different classifiers. For the report we divided the backgound part in half and worked on it separately, and Jeongjae did the rest of the report. Due to that, Haeun created the slide for the presentation and for the presentation, we both equally presented the project.