Classifying Images as AI Generated or Real

1st Andrew Greensweight
Stevens Institute of Technology
Hoboken, USA
agreensw@stevens.edu

2nd Michael Salek
Stevens Institute of Technology
Hoboken, USA
msalek@stevens.edu

3rd Ronald "Joey" Rupert
Stevens Institute of Technology
Hoboken, USA
rrupert@stevens.edu

***Abstract*—The quality of AI-generated images has rapidly increased, leading to concerns of authenticity and trustworthiness. The aim of this project is to investigate whether computer vision techniques can effectively detect when images have been generated by AI. By addressing this problem, we can contribute to the development of algorithms that enhance the authenticity verification of images. It is important to review all of the possible machine learning techniques and find the best one to do image classification for these types of images.**

I. Introduction

The problem we have chosen to attack is the modern problem of the authenticity of images. AI-generated images have become more and more realistic in recent years, as AI has become more powerful, and more resources have been devoted to improving it and accelerating its progress further. We are now reaching a place where AI-generated images can be almost indistinguishable from authentic images. The problems that this could cause are clear. The quality of AI-generated images has seen a significant advancement, raising important concerns surrounding their authenticity and trustworthiness. As AI technology continues to evolve, there is a growing need to develop reliable methods to differentiate between AI-generated images and real images. The objective of this project is to explore the capabilities of computer vision techniques in effectively detecting AI-generated images.

The main focus of our investigation is to implement machine learning algorithms that can accurately classify images as either AI-generated or real. By addressing this problem, we aim to contribute to the field of computer vision by enhancing the authenticity verification process for images and highlight the advantages and disadvantages of particular machine learning algorithms.

To achieve our goal, we will leverage machine learning algorithms. We will compare and contrast the following machine learning algorithms: Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Convolutional Neural Network (CNN). With these three algorithms in mind, we will compare their advantages and disadvantages in the realm of image of classification.

For the dataset, we used a public database known as Kaggle. We were able to locate a large set of labeled images to implement our algorithms on. For pre-processing we ensured the images were all the same size and in the proper color format. We then used a pre-trained CNN to extract features to support processing. Following this, we were able to flatten the image data and create randomly shuffled training dataset.

One of the challenges experienced came in the sense of resources. Certain members of the group were using Google Colaboratory which has limited resources in the free version. Additionally, the availability of Google's resources also varies depending on the day and time. Due to resource limitations, we had to use a subset of the full Kaggle dataset.

In summary, the primary objective of this project is to investigate the effectiveness of machine learning algorithms in detecting AI-generated images.

II. Related Work

This subsection describes related techniques for problems of similar nature. The binary classification problem is one that is well understood in the space of machine learning. The variability and noise in natural data, regardless of the medium, typically makes it difficult to build an accurate classification system by hand. Most classification systems are built using a combination of automatic learning techniques and machine learning algorithms [1]. A simple block diagram can represent this traditional method of building a classification system; the block diagram is as follows: a raw input is provided to a feature extraction module, the feature extraction module produces a feature vector, the feature vector is fed into a trainable classifier module, and class scores are determined. The appropriate feature extractor used to be required because the learning techniques used by the classifiers were limited to low-dimensional spaces with easily separated classes [2]. In modern machine learning applications, this is no longer the case because robust machine learning techniques are readily available for use. These robust techniques can handle high-dimensional inputs and can generate decision functions when fed with these large datasets [1].

One highly researched method for building a classification system is to use a Convolutional Neural Network (CNN), which is a specialized neural network architecture that utilizes knowledge about the invariances of two-dimensional shapes by using local connection patterns and by imposing constraints on the weights [3]. In this approach for image classification, the CNN can be generalized for an image of dimensions x and a filter matrix w as described below [4], [5], [6]:

$$(x * w)(i,j) = \sum_{m=1}^{M} \sum_{n=1}^{N} x(i+m-1, j+n-1) w(m,n) \quad (1)$$

where (i,j) is the output for the feature vector map, and (m,n) represents the location of the filter w. The next step is to apply convolutional operations to the input with each of the filters and to apply an activation function. Following this, output is flattened and the spatial dimensions are reduced, which is then input into the connected layers of the CNN [4]. In this described approach, the network outputs a single neuron with a S-Shaped Sigmoid activation function. For the case of a binary classification problem such as the one addressed in this report, the value of the neuron represents the likelihood of that class, so the values are rounded to the nearest value for inference. The advantages of using CNN for image classification is that it is it provides very efficient image processing capabilities, it has high accuracy rates because of its superb ability to recognize patterns in images, and it does not require manual feature engineering because it can learn from raw pixel data. The disadvantages of CNN is that it requires large datasets, it can be computationally expensive, and it is prone to overfitting [5].

Another technique commonly used in image classification is KNN. For example, for automated cervical cancer screening from pap-smear images, some of the most cited literature lends itself to filtering, thresholding, and KNN as being the most frequently used techniques for pre-processing, segmentation, and classification of pap-smear images [7]. KNN is a non-parametric classifier used for regression and classification [8]. For the KNN classifier, there is no need for a training phase. With the KNN classifier, each image's nearest neighbors based on the distance from the training set points are examined, and the classification is determined by majority voting. This approach involves extracting features, normalizing them, and applying KNN for classification. KNN is known for its robustness, speed, and tolerance to noise. However, it becomes more complex as the number of attributes increases and assumes similarity among instances with the same attributes [7].

III. Our Solution

A. *Description of Dataset*

The dataset contains two classes - REAL and FAKE. For REAL, the images are collected from Krizhevsky and Hinton's CIFAR-10 dataset. For the FAKE images, they were generated to be the equivalent of CIFAR-10 with Stable Diffusion version 1.4.There are 100,000 images for training (50k per class) and 20,000 for testing (10k per class). These images were pulled from Kaggle and can be found referenced in the References section[9], [10].

For this particular implementation, only a subset of the dataset was used due to the limitations of Google Colaboratory. The training set comprised of 5,000 real images and 5,000 AI generated images (fake), and the test set consisted 1,500 real images and 1,500 AI generated images. In addition to extracting a subset of the Kaggle dataset, labels appended to each image to ensure their true classification was not lost before they were randomly shuffled. Following the shuffling of the training set, the image dimensions were resized to 32 by 32 pixels and the images were converted to RGB (red, green, blue) format. We then applied geometry augmentations to each image as a pre-processing step to increase the robustness and variability of the training data. By applying geometry augmentations, the training dataset can be expanded with variations of the original images. Flipping an image horizontally or vertically can create new instances with different orientations or viewpoints, making the classifier more tolerant to variations in object positioning or orientation[11].

B. *Machine Learning Algorithms*

Depending on the objective of the Machine Learning Algorithm there are multiple options to choose from. To best classify images Artificial Neural Networks, Convolutional Neural Networks, K Nearest Neighbor, Decision Trees, and Support Vector Machines can all be implemented[12]. The options from this list that were utilized were Support Vector Machine (SVM), K-Nearest Neighbor (KNN), and Convolutional Neural Network(CNN).

1) Algorithm I: Support Vector Machine: Support Vector Machines are a set of supervised learning methods used for classification, regression and outliers detection[13]. Advantages and disadvantages of Support Vector Machines can be seen in Table III.

TABLE I. Advantages and Disadvantages of Support Vector Machines

Advantages	Disadvantages
• Useful for high dimensional spaces	• Over-fitting due to too many features and not enough samples
• Memory efficient	• Expensive cross-validation
• Ability to utilize different Kernel functions	

Support Vector Machines classify items by constructing a hyper-plane or set of hyper-planes that achieves the best functional margin, largest distance from the nearest training points of any class, to best classify new data points. An example of a hyper-plane and its functional margin can be seen in Figure 1. In this figure the dotted lines represent the best margin boundaries that can be achieved, these boundaries will run through the data points that will provide the greatest distance.

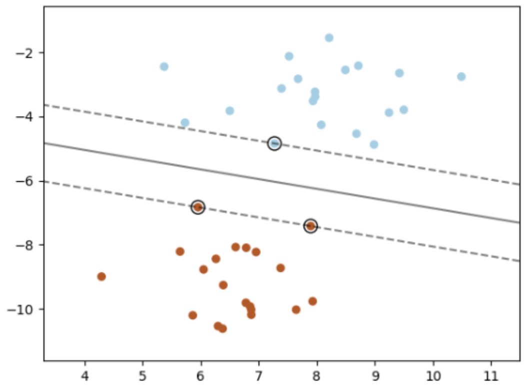

Fig. 1. A hyperplane that has been generated for a dataset[13].

For image classification Support Vector Classification is used. The problem that it solves to is the following primal problem:

$$min_{w,b,\zeta}\frac{1}{2}w^Tw + C\sum_{i=1}^{n}\zeta_i \tag{2}$$

subject to $y_i(w^T\phi(x_i) + b) \geq 1 - \zeta_i, \zeta_i \geq 0, i = 1, ..., n$

In this equation the margin is trying to be maximized by minimizing $||w||^2 = w^Tw$. This will also get a penalty for any misclassified samples or any samples that fall within the boundary. In the ideal solution all samples would have a value ≥ 1. Due to it being nearly impossible to achieve a value of ≥ 1, a penalty term C is used to act as an inverse regularization parameter. The dual problem that goes with the primal problem is:

$$min_\alpha\frac{1}{2}\alpha^TQ\alpha - e^T\alpha \tag{3}$$

subject to $y^T\alpha = 0, 0 \leq \alpha_i \leq C, i = 1, ..., n$

where e is the vector of ones, and Q is an n by n positive semidefinite matrix built using the kernal K where $K(x_i, x_j) = \phi(x_i)^T\phi(x_j)$ is the kernel [13]. What this shows is that the training vectors are mapped into a higher dimensional space.

Once the values for the equations above are found through training, a decision function can be made shown by:

$$\sum_{i\epsilon SV} y_i\alpha_iK(x_i, x_j) + b \tag{4}$$

This equation is then used for new samples and predicts the class based on whether the sign is positive or negative.

2) Algorithm II: K-Nearest Neighbor: KNN's flexibility and non-parametric nature make it suitable for image classification in real versus AI generated images. Table II lists the Advantages and Disadvantages of using KNN for classification.

TABLE II. ADVANTAGES AND DISADVANTAGES OF K-NEAREST NEIGHBOR

Advantages	Disadvantages
• Simple and easy to implement	• Computationally expensive
• Non-parametric so it does not make assumptions about distribution of data	• Sensitivity to feature scaling as larger values will dominate distance calculations
• Robust to noisy data because it relies on local neighborhood	• Storage requirements for storing entire dataset in memory
• Non-linear decision boundaries to capture complex relationships in data	• Not suitable for high-dimensional data as performance of distance based algorithms degrades with high-dimensional spaces

In KNN, the classification decision is based on the majority vote of the k nearest neighbors in the feature space. Given an unlabeled instance, KNN identifies the k nearest neighbors from the training set based on some distance metric (e.g., Euclidean distance or Manhattan distance). The class label of the unlabeled instance is then determined by the majority class among its k nearest neighbors [11]. This can be expressed mathematically as follows: for a training set as X_train, where each instance x_i is a feature vector with d dimensions. The corresponding labels are y_i, indicating whether the instance is real or AI generated. For a test instance x_test, the goal is to predict its label y_test. To determine the k nearest neighbors of x_test, KNN calculates the distances between x_test and all instances in X_train. This can be done using various distance metrics, such as the Euclidean distance:

$$\text{Euclidean Distance}(x, x') = \sqrt{\sum_{i=1}^{d}(x_i - x'_i)^2} \tag{5}$$

or the Manhattan distance:

$$\text{Manhattan Distance}(x, x') = \sum_{i=1}^{d}|x_i - x'_i| \tag{6}$$

Once the distances are computed, the k instances with the smallest distances to x_test are selected as the nearest neighbors.

To determine the class label of x_test, KNN employs the majority voting scheme[8]. Each nearest neighbor contributes a vote towards their corresponding class label, and the class with the highest number of votes is assigned as the predicted label for x_test.

This approach allows KNN to capture the underlying patterns in the training data without making assumptions about the data distribution. It can effectively handle complex decision boundaries and nonlinear relationships between features, making it suitable for distinguishing between real and AI generated images where the statistical characteristics may differ significantly.

By considering the local structure of the data and the majority voting scheme, KNN provides a flexible and adaptable approach to image classification that can be applied to various scenarios, including the classification of real versus AI generated images.

3) Algorithm III: Convolutional Neural Network: Convolutional neural networks are a type of neural network designed to handle images more efficiently than typical neural networks, to reduce overfitting and model complexity.

TABLE III. ADVANTAGES AND DISADVANTAGES OF CNNS

Advantages	Disadvantages
• Less complex than traditional neural nets with reduced connections and shared weights	• Require lots of training data
• Extract features automatically	• Design can be difficult
• Reduced risk of overfitting	

Convolutional neural networks are fundamentally similar to other artificial neural networks, using layers of neurons connected by weighted edges, the neurons performing different operations on the image to arrive at the final result. Typical neural networks are fully-connected, meaning that all neurons of one layer are connected to all neurons of the previous layer. While this is useful for many types of problems, it becomes extremely impractical for images, and can cause severe overfitting as well as huge performance costs.

CNNs, however, are specifically designed for use on images, utilizing convolutional layers to reduce the complexity of large images. These layers use small image filters to test for the presence of specific patterns across the image or some subset of it, using them essentially as small feature detectors. Convolving these feature detectors with the image forms a ”feature map”, showing the presence or lack thereof of features in different parts of the image. They also use a different type of layer called a ”pooling” layer, where the outputs of several neurons are combined into a single neuron in the next layer.

These techniques, when combined, allow for drastically more efficient handling of images, being able to quickly and inexpensively detect key features throughout the image similarly to how human eyes and brains process images.

C. Implementation Details

1) Implementation I: Support Vector Machine: To implement SVM the scikit learn module was utilized along with GridSearchCV to find the best hyperparameters. Due to the large size of the dataset a smaller selection of data was pulled to plug into GridSearchCV to discover the best parameters to use. For this model the parameters in Table IV were utilized. It also is noted that the kernel used for all tested models was Radial Basis Function(RBF) which requires a C and gamma parameter.

TABLE IV. SVM PARAMETERS USED FOR MODEL SELECTION.

C	gamma
0.1	0.0001
1	0.001
10	0.1
100	1

When all of these parameters were plugged into the smaller dataset of images(500 Real and 500 Fake images for training) the best parameters found were C = 10 and gamma=0.0001. This model was then run against the smaller dataset of test images(150 Real and 150 Fake images) and an accuracy score of 72.0%. Now that the best parameters were found they were utilized against the full dataset that was utilized for this project which was 5000 real and 5000 fake images for training and 1500 real and 1500 fake images for testing. When these same parameters were run with the larger dataset an accuracy of 79.0% was achieved. The training time for the model in the larger dataset to significantly longer to run, it took approximately 10 minutes for the small dataset and it took approximately 650 minutes for the large dataset. These results were all found with a simple resizing adjustment to all of the images to a size of 150 by 150. Preprocessing was also utilized and would output the images at a size of 32 by 32, this meant the model ran much faster at a time of 40 minutes, but the accuracy decreased, most likely due to there being less features to account for, to a total of 70.9%.

2) Implementation II: K-Nearest Neighbors: The code starts by defining a parameter grid that specifies the values to be explored during hyperparameter tuning. The parameters include the number of neighbors $n\_neighbors$, the weight function (weights), and the distance metric (p). The $n\_neighbors$ parameter determines the number of neighbors to consider when making predictions. The ”weights” parameter defines the weight assigned to each neighbor when making predictions. It can be either ”uniform” or ”distance”. Uniform means all neighbors are weighted equally, while distance means that closer neighbors have a higher influence on the prediction. The distance metric parameter determines the power parameter for the Minkowski distance calculation, where $p = 1$ corresponds to Manhattan distance, and $p = 2$ corresponds to Euclidean distance [14]. By trying different combinations of these parameters, the implementation aims to find the optimal configuration for the KNN classifier.

The SciKit Learn method KNeighborsClassifier is then initialized without any specific parameter values. This creates a KNN classifier with default settings.

Grid search is performed using SciKit Learn Module GridSearchCV, which is a technique for systematically searching the parameter space and finding the best combination of parameters. The estimator parameter of GridSearchCV is set to the previously created KNN classifier. The scoring parameter is set to ’accuracy’ to evaluate the performance of different parameter combinations based on accuracy. Cross-validation with 3 folds (cv=3) is used to estimate the accuracy. Using a cross-validation value of ”3” strikes a balance between computational efficiency and obtaining a reasonable estimate of the model’s performance.

The grid search fits the KNN classifier to the training data and systematically evaluates the performance of different parameter combinations. Once the grid search is complete, the best parameters and corresponding accuracy are obtained using the $best\_params\_$ and $best\_score\_$ attributes, respectively. The best parameters found in this case are shown in Table VII.

TABLE V. **BEST KNN PARAMETERS**

Parameter	Best Value
No. of Neighbors	”31”
Weights	”Uniform”
Distance Metric	1 (Manhattan Distance)

A new KNN classifier, knn_best, is then created using the best parameters obtained from the grid search. This classifier is fitted to the training data to learn the underlying patterns. The largest value for the number of neighbors in the parameter grid was based on the square root of the number of training samples, which was 10,000, yielding a good experimental k value of 31.

Next, the labels for the test set are predicted using the fitted knn_best classifier. The accuracy of the predictions is calculated by comparing the predicted labels with the true labels of the test set using the $accuracy\_score$ function.

Finally, the test accuracy is printed to evaluate the performance of the tuned KNN classifier on the unseen test data. This provides an indication of how well the classifier generalizes to new, unseen images. With this implementation, the measured accuracy was found to be 69.03%. Table VI provides a confusion matrix which summarizes the performance of the classification model.

TABLE VI. KNN CONFUSION MATRIX

		True Class	
		Fake	Real
Predicted Class	Fake	913	587
	Real	342	1158

Based on the confusion matrix, the precision, which can be measured as how well the classifier performs in correctly identifying the positive instances is computed to be 72.78%.

3) Implementation III: Convolutional Neural Network: To build and train a convolutional neural network for image classification, the keras python library was utilized, as well as the same preprocessing systems as the other algorithms for the sake of equalized testing. With this library, the implementation of a simple CNN was very straightforward. Relatively little hyperparameter tuning was required, with the initial set of layers already performing exceptionally well on the smaller images. A basic netowrk was creating, with two convolution layers, a pooling layer, and a dense output layer. The primary tuning done was varying the number of epochs run with the data by the network, from 5 to 25, with the varying numbers of epochs (3 selected results shown below).

The results from the CNN were extremely promising. Conveniently, the keras model measures validation accuracy, allowing us to easily track the accuracy of the model as we go. With this, we were able to see that while the number of epochs run did not significantly change the effectiveness of the model, the model was extremely accurate relative to the others we tested.

TABLE VII. **EPOCHS VS. ACCURACY FOR CNN**

Epochs	Accuracy
5	.9356
10	.9283
25	.9366

The training time for the algorithm was also excellent, averaging 25 seconds per epoch.

IV. COMPARISON

Comparing our methods, factoring in difficulty/complexity of implementation, training time, and most of all accuracy, it seems apparent that the convolutional neural network is the best-suited method for distinguishing real images from AI-generated ones. Its accuracy was significantly higher with the preprocessed data than either of the other two methods, with a relatively simple implementation and a fast training time.

The accuracy for the CNN was, in fact, slightly higher than the Bird and Lotfi results with the same dataset, who achieved an accuracy of .9298 with a CNN. Given that our accuracy varied slightly between runs, the actual performance is likely similar between the two; but this does further illustrate the point that a CNN is an excellent method for this specific test, and the best tested here by far.

V. FUTURE DIRECTIONS

To better improve the results of this project it would be recommended to utilize all available data to better generate the models. This would require better hardware and more efficient systems to reduce the calculation time required. Another source of improvement would be to utilize the system GPU to further improve the calculation times.

In terms of preprocessing, further research would need to be done to find the ideal preprocessing to clean the images up into the ideal input for the models. Once the ideal form of preprocessing is found it could be applied to all utilized samples and further increase the accuracy of the models.

Future value could be provided by reading in questionable images from the internet and helping web users determine if the images they are looking at are real or if they are generated by AI in real time.

VI. CONCLUSION

In conclusion, this project aimed to investigate the effectiveness of machine learning algorithms in detecting AI-generated images. Three machine learning algorithms, namely Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Convolutional Neural Network (CNN), were implemented and compared for image classification.

The SVM algorithm, with the RBF kernel, achieved an accuracy of 79.0% on the test set. It demonstrated the ability to handle high-dimensional data and provided a flexible decision boundary. However, it required a longer training time and was computationally expensive compared to other algorithms. The SVM algorithm was suitable for image classification tasks where there is a need for robustness and the ability to handle high-dimensional feature spaces.

The KNN algorithm, with a k value of 31 and Manhattan distance metric, achieved an accuracy of 69.03% on the test set. It demonstrated simplicity and robustness, but it was computationally expensive and sensitive to feature scaling. KNN was suitable for image classification tasks where the local structure of the data is important, and it can capture complex relationships in the data.

The CNN algorithm, with two convolutional layers and a dense output layer, achieved an accuracy of approximately 93.56% with only 5 epochs on the test set. It demonstrated high accuracy, reduced complexity, and fast training time. CNN was particularly suitable for image classification tasks, providing efficient image processing capabilities, high accuracy rates, and the ability to learn from raw pixel data.

Based on the results and considering the performance metrics, the CNN algorithm emerged as the most suitable technique for detecting AI-generated images in this project. It provided the highest accuracy and demonstrated the capability to learn complex patterns from the data. The other techniques, SVM and KNN, also showed reasonable performance but had limitations such as computational complexity and sensitivity to certain factors.

Further improvements could be made by utilizing more available data, optimizing preprocessing techniques, and exploring other neural network architectures. Additionally, leveraging GPUs for faster computations and real-time image classification could be a valuable direction for future research.

Overall, this project contributes to the development of algorithms for enhancing the authenticity verification of images, addressing the growing concerns surrounding AI-generated images' authenticity and trustworthiness.

REFERENCES

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 11 1998.

[2] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.

[3] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, Winter 1989.

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[5] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai *et al.*, "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, pp. 354–377, 2018.

[6] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[7] W. William, A. Ware, A. H. Basaza-Ejiri, and J. Obungoloch, "A review of image analysis and machine learning techniques for automated cervical cancer screening from pap-smear images," *Computer Methods and Programs in Biomedicine*, vol. 164, pp. 15–22, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169260717307459

[8] M. Sharma, S. K. Singh, P. Agrawal, and V. Madaan, "Classification of clinical dataset of cervical cancer using knn," *Indian Journal of Science and Technology*, vol. 9, no. 28, 2016.

[9] J. J. Bird and A. Lotfi, "Cifake: Image classification and explainable identification of ai-generated synthetic images," 2023.

[10] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[11] A. VAILAYA, A. JAIN, and H. J. ZHANG, "On image classification: City images vs. landscapes," *Pattern Recognition*, vol. 31, no. 12, pp. 1921–1935, 1998. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S003132039800079X

[12] T. Jain, "Basics of image classification techniques in machine learning," 2023, [Online; accessed 07-July-2023]. [Online]. Available: https://iq.opengenus.org/basics-of-machine-learning-image-classification-techniques/

[13] scikit learn, "1.4. support vector machines," 2023, [Online; accessed 07-July-2023]. [Online]. Available: https://scikit-learn.org/stable/modules/svm.html

[14] ——, "sklearn.neighbors.KNeighborsClassifier," 2023, [Online; accessed 07-July-2023]. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html