THE REPUBLIC OF TÜRKİYE
MUĞLA SITKI KOÇMAN UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DEPARTMENT OF COMPUTER ENGINEERING

A MACHINE LEARNING-BASED APPROACH FOR AUTOMATIC THEFT DETECTION

İREM KARACA ULUOĞLU

MASTER'S THESIS

DECEMBER 2024
MUĞLA

MUĞLA SITKI KOÇMAN UNIVERSITY

Graduate School of Natural and Applied Sciences

APPROVAL OF THESIS

The thesis submitted by **İREM KARACA ULUOĞLU,** titled **"A MACHINE LEARNING-BASED APPROACH FOR AUTOMATIC THEFT DETECTION"** was unanimously accepted by the jury members on December 27, 2024, to fullfill the requirements for the degree of Master's in the Department of Computer Engineering.

THESIS JURY MEMBERS

Assist. Prof. Dr. Erdem TÜRK **(Head of Jury)** Signature:

Department of Computer Engineering,
Muğla Sıtkı Koçman University, Muğla _____

Assoc. Prof. Dr. Barış Ethem SÜZEK **(Supervisor)** Signature:

Department of Computer Engineering,
Muğla Sıtkı Koçman University, Muğla _____

Assoc. Prof. Dr. Gürhan GÜNDÜZ **(Member)** Signature:

Department of Computer Engineering,
Pamukkale University, Denizli _____

APPROVAL OF HEAD OF THE DEPARTMENT

Assoc. Prof. Dr. Barış Ethem SÜZEK Signature:

Head of Department Computer Engineering,
Muğla Sıtkı Koçman University, Muğla _____

Assoc. Prof. Dr. Barış Ethem SÜZEK Signature:

Supervisor, Department Computer Engineering,
Muğla Sıtkı Koçman University, Muğla _____

Date of Defense: 12/27/2024

I hereby declare that all information in this document has been obtained and presented in accordance with academic and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

İrem Karaca Uluoğlu

12/27/2024

ÖZET
OTOMATİK HIRSIZLIK TESPİTİ İÇİN MAKİNE ÖĞRENİMİ TABANLI BİR YAKLAŞIM

İrem KARACA ULUOĞLU

Yüksek Lisans Tezi

Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Barış Ethem SÜZEK

Aralık 2024, 94 sayfa

Hırsızlık, küresel olarak artan yaygın bir suçtur. Hırsızlığı önlemek için sıklıkla video gözetim sistemleri kullanılsa da bu sistemlerde hırsızlığın tespiti, sıkıcı ve tutarsız olabilen manuel izlemeye bağlıdır. Bu, otomatik hırsızlık tespit sistemlerine olan ihtiyacı ortaya koymaktadır. Bu çalışmada, derin öğrenme tabanlı önceden eğitilmiş insan eylemi tanıma modelleri kullanarak otomatik hırsızlık tespit sistemi için makine öğrenimi tabanlı bir yaklaşım geliştirmeyi amaçladık. Yaklaşımımız dört temel adımdan oluşuyordu: veri seti hazırlama, özellik vektörü oluşturma, hırsızlık tespiti için model eğitimi ve performans değerlendirmesi. Central Florida Üniversitesi-Suç kamu veri setinden türettiğimiz veri setlerini kullandık. İlk veri setinde stealing ve normal videolar, ikinci veri setinde shoplifting, stealing ve normal videolar ve üçüncü veri setinde ise shoplifting, stealing, robbery ve normal videolar yer alıyordu. Hırsızlıkla ilgili kategorileri içeren videolar "theft" olarak etiketlenirken, normal videolar "normal" olarak etiketlendi. Her videodan 400 özelliğe sahip öznitelik vektörleri oluşturmak için önceden eğitilmiş dört insan eylem tanıma modeli kullandık. Bu, adım üç verisetinden türetilen 12 ayrı veri seti ile sonuçlandı. Sonra farklı veri kümeleri arasında hırsızlığı normal videolardan ayırt etmek için ikili sınıflandırma yapıldı. Eğitim verilerine Support Vector Machine, Decision Tree, Neural Network, Random Forest, K-Nearest Neighbors, Gaussian Naive Bayes ve Gradient Boosting olmak üzere çeşitli makine öğrenimi algoritmaları hiperparametre ayarlaması ile uygulandı. Model performansı, test kümesine göre değerlendirildi. En iyi performans gösteren, 0,90'lık AUC, 0,90'lık doğruluk, 0,91'lık duyarlılık, 0,90'lık kesinlik ve 0,90'lık F1 score ile Neural Network modelidir. Bu yaklaşım, hırsızlık önlemede gerçek video gözetim verilerine uygulanabilir ve insan güvenliğini artırabilir. Bu çalışma, insan müdahalesini azaltarak, doğruluğu artırma ve video gözetimi işleme sürecinde tutarlılığı koruma potansiyeli sunmaktadır.

Anahtar Kelimeler: Hırsızlık Tespiti, Makine Öğrenimi, Eylem Tanıma, Video'da Otomatik Hırsızlık Tespiti, Video Gözetimi

ABSTRACT
A MACHINE LEARNING-BASED APPROACH FOR AUTOMATIC THEFT DETECTION

İrem KARACA ULUOĞLU

Master of Science (M.Sc.)
Graduate School of Natural and Applied Sciences
Department of Computer Engineering
Supervisor: Assoc. Prof. Dr. Barış Ethem SÜZEK
December 2024, 94 pages

Theft, a common crime, is increasing globally. Although video surveillance systems are frequently employed to prevent theft, the detection of theft in these systems depends on manual monitoring, which can be tedious and inconsistent. This reveals the need for automatic theft detection systems. In this study, we aimed to develop a machine learning-based approach for automatic theft detection systems using deep learning-based pre-trained human action recognition models. Our approach consisted of four key steps: dataset preparation, feature vector generation, model training for theft detection, and performance evaluation. We used datasets derived from the University of Central Florida-Crime public dataset. The first dataset contained stealing and normal videos, the second included shoplifting, stealing, and normal videos, and the third dataset had shoplifting, stealing, robbery, and normal videos. Videos involving theft-related categories were labeled as "theft," while normal videos were labeled as "normal". We utilized four pre-trained human action recognition models to generate feature vectors with 400 features from each video. This resulted in 12 distinct datasets derived from three datasets. Binary classification was then performed to distinguish theft from normal videos across the different datasets. Machine learning algorithms were applied to the training data, including Support Vector Machine, Decision Tree, Neural Network, Random Forest, K-Nearest Neighbors, Gaussian Naive Bayes, and Gradient Boosting, with hyperparameter tuning. Model performance was evaluated based on the test set. The best-performing model was the Neural Network, achieving an AUC of 0.90, accuracy of 0.90, recall of 0.91, precision of 0.90, and an F1 score of 0.90. This approach can be applied to real video surveillance data in theft prevention and increase human safety. Here, by lowering human involvement, there will be the potential to increase accuracy and maintain consistency in video surveillance processing.

Keywords: Theft Detection, Machine Learning, Action Recognition, Automatic Theft Detection in Video, Video Surveillance

ACKNOWLEDGEMENTS

First of all, I am grateful to my advisor Assoc. Prof. Dr. Barış Ethem Süzek for his knowledge, tolerance and guidance in the realization of this study. Your valuable feedback and continuous support enlightened my path and contributed greatly to completing my thesis in the best possible way.

Additionally, I would want to thank my spouse, Ömer Uluoğlu, whose constant love and support provided me with the emotional support I needed. I am grateful to my parents, Ayşe and Şahin Karaca, and brother, Anıl Karaca, for their constant backing and belief in me.

Finally, throughout the process of finishing this thesis, I would want to sincerely thank everyone who helped and advised me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

A crime is a violation of law that is illegal, detrimental to a person or society, and subject to legal consequences. Damage can be monetary, psychological, or physical. Crime causes enormous losses in terms of both people and money. The local and worldwide crime rates are increasing. Theft is among common crimes defined as taking one's property without the use of force. Theft is rising nationally and internationally. As a result of this increase, the use of surveillance video cameras in public places such as airports, train stations, schools, hospitals, banks, shopping malls, education institutions, and crowded streets has also increased to provide safety. Video surveillance systems are used as a preventive measure against theft. The purposes of surveillance video cameras are to maintain track of regular activity and identify unusual ones. According to the definition used in the surveillance sector, an anomaly is any unusual occurrence that deviates from normal behavior and is generally understood to be a deviation from standard norms, types, arrangements, or forms. Among the anomalies, the theft anomaly, which creates material and moral losses, is very important. In video surveillance systems, theft detection relies on manual monitoring and identification of suspicious activities from video feeds by humans. Therefore, continuous monitoring could be tedious, resource-constrained, cost-inefficient, inconsistent, and unscalable due to human involvement.

For many applications, like theft detection, preserving security depends on the ability to identify anomalies in video surveillance. Since anomalous occurrences only occur 0.01% of the time and 99.9% of the monitoring time is wasted, manually detecting anomalies is a labor-intensive and ongoing procedure that takes a lot of time and effort from human workers (Duong et al., 2023). Thus, intelligent systems that can automatically identify abnormal events in the video stream are desperately needed. The capacity to automatically identify security incidents or potentially dangerous occurrences occurring inside the field of vision of cameras is offered by Automated Video Surveillance technology (Jadhav et al., 2017).

There is a need for automatic theft detection systems capable of handling large volumes of data that are consistent, accurate, scalable, and continuously available.

Several applications that use action recognition and machine learning have surfaced in recent years for the detection of theft. Despite much research in this field, no transfer learning implementation has been done. In this work, we constructed feature vectors using a recently developed theft dataset by applying action recognition methods. For this process, we chose four separate action recognition algorithms, and we used machine learning methods on the feature vector datasets that were produced. We also included steps for feature extraction. Our work differs from earlier research in this subject because of these techniques, especially the application of transfer learning.

2. LITERATURE REVIEW

Researchers have carried out various studies to date on the detection of anomalies in surveillance videos. There are quite extensive publications on this subject. In 2018, Sultani and colleagues conducted a research on the deep anomaly ranking algorithm, which predicts high anomaly scores for anomalous video segments and offered a new large-scale, first-of-its-kind dataset of 128 hours of videos (Sultani et al., n.d.). A survey from that same year suggested using a one-class neural network model to identify anomalies in huge, complicated datasets (Chalapathy et al., 2018). In 2019, using multilevel representations of both intensity and motion data, Vu and colleagues created a framework for accurate anomaly detection (Vu et al., 2019). In 2021, Ullah and colleagues introduced a time-complexity-reduced anomaly recognition framework for surveillance that is effective and lightweight, utilizing convolutional neural networks (CNNs). To accurately detect anomalous behavior in surveillance videos, they extracted spatial CNN attributes from a sequence of video frames and fed them to the suggested residual attention-based long short-term memory network (Ullah et al., 2021). In 2020, in contrast to full-frame learning, the research presented a method for learning abnormal behavior in the video by identifying attention to the area using spatiotemporal information (Nasaruddin et al., 2020). In a survey in 2021, a multitask deep neural network was suggested as a solution to the anomaly detection problem, which was considered a fully supervised learning problem (Wan et al., 2021). In 2022, Zaigham Zaheer and colleagues presented a unique technique for video anomaly identification using unsupervised Generative Cooperative Learning (GCL) that builds a cross-supervision between a discriminator and a generator by taking advantage of the low frequency of anomalies (Zaigham Zaheer et al., n.d.). A survey that utilizes a model based on deep learning for surveillance system anomaly detection was released in 2022 (Amin et al., 2022) . In 2023, Liu and colleagues provided a two-stream spatio-temporal generative model to identify anomalous behavior in real-time from video surveillance (Liu et al., 2023).

Datasets have been created to be used for anomaly detection. In 2008, the subway dataset, which consists of two long video recordings that capture individuals entering and leaving a train station, became accessible (Adam et al., 2008). In 2009, the UMN dataset that simulated a populated area where actors walk to an exact location and escape with anomalous behavior was published (Mehran et al., n.d.). In 2013, the Avenue dataset that involves abnormal events that were recorded at CUHK campus avenue was offered (Lu et al., 2013). In 2013, the UCSD dataset that has two sub-datasets, Pedestrian 1 and Pedestrian 2, which involve events captured in various crowded stages, was published (*Anomaly Detection and Localization in Crowded Scenes*, n.d.). In 2016, the ShanghaiTech Campus dataset that includes 13 sequences, including intricate lighting, camera angles, and many kinds of anomalies, most of which are connected to unusual things, the wrong direction, and unusual behavior, was offered (Liu et al., n.d.). In 2018, the UCF-Crime dataset that contains anomalies about stealing, shoplifting, robbery, burglary, abuse, arrest, arson, assault, explosion, fighting, road accidents, shooting, and vandalism was published (Sultani et al., n.d.). In 2020, the Street Scene dataset that involves anomalies, for instance, bikers outside lanes and jaywalking, became accessible (Ramachandra & Jones, n.d.). In 2021, the HR-Crime dataset, which is a subset of the UCF-Crime dataset that includes Human-Related (HR) videos, was released (Boekhoudt et al., n.d.).

Review articles that describe the datasets and machine learning methods that have been published thus far in the anomaly detection field are also available. A review paper on methods based on deep learning for detecting anomalies in videos was released in 2021 (Nayak et al., 2021). In 2022, a review publication that summarizes anomaly analysis datasets and machine learning models was released (Tran et al., 2022). In 2023, a review paper was released about deep learning approaches to the detection of anomalies in surveillance videos (Chandrakala et al., 2023). In the same year, a paper that analyze the existing deep learning structures and machine learning methods used to detect anomalous cases in surveillance videos to examine their benefits and difficulties was published. Also, this survey summarizes anomaly detection datasets (Şengönül et al., 2023). Another review paper that explains all datasets that contain abnormal events and discusses deep learning models up to now was published (Duong et al., 2023).

The field of human action recognition is fast developing, with notable progress made in the creation of models and systems. Review papers that summarize the state of the field have emerged as academics explore deeper. In 2024, an extensive review of Human Action Recognition systems is given in the paper. It discusses deep learning and machine learning techniques, benchmark datasets for human action recognition, metrics to assess the effectiveness and architecture of systems for human action recognition, current problems, and potential future paths for the field (Karim et al., 2024). In the same year, a review of deep learning techniques for multimodal vision-based human action recognition is presented in a publication. Shafizadegan and colleagues present a four-level category for human action recognition techniques that take into account the architecture, similarities, availability, and modalities of the frameworks. They identify benchmark human action recognition as well. The study examines possible future research opportunities in the field and highlights the top-performing techniques on well-known and current benchmark datasets (Shafizadegan et al., 2024).

When all the publications are considered, it is concluded that there are a lot of papers related to anomaly detection or human action recognition systems. However, few papers have been conducted to automatically detect theft or subcategories of theft detection. In 2018, a publication that provides theft detection with machine learning was released. In this paper, a motion was detected using convolution neural networks and sent a warning to the owner (Kushwaha et al., 2018). In 2023, a paper about the detection of theft using deep learning was published. This survey uses real-time object identification on live video feeds to improve security protocols and enable quick reactions to possible threats (Shirole & Virdhe, 2023). In the same year, a paper indicates using a hybrid neural network to identify shoplifting. To compare their model with other approaches, Muneer and colleagues built a new dataset, which they used to compare and find that their model performed better (Muneer et al., 2023). In 2024, an article presents a new technique for identifying theft in surveillance video. This approach tries to lower false positives and is meant to work well both in the daytime and at night. Waddenkery and Soma emphasize the importance of the system's capacity to gain knowledge from tiny quantities of data and adapt to changing circumstances for real-time applications (Waddenkery & Soma, 2024).

Transfer learning, as everyone knows, is an approach to machine learning that applies the information of a trained model developed for one task to another that is related but different. This allows for the transfer of knowledge from task one to the second model, which is task-specific. It decreases the quantity of data required to train a model and increases efficiency. You can accomplish high accuracy in your particular domain by utilizing pre-trained models on huge data sets. Pre-trained models significantly decrease the amount of time needed for training. Models may overfit the training set in situations where there is insufficient data. Through the application of the more comprehensive knowledge gained from the original dataset, transfer learning helps to reduce this risk. It enables simple adjustment with minor changes to new operations or domains for a specific application, a pre-trained model can be efficiently adjusted. By encouraging knowledge sharing across many tasks, transfer learning makes it possible to apply ideas from one work to another. For instance, video analysis can benefit from the features acquired in image recognition.

In this research, we will work on theft detection. We will discuss the definition of theft by Article 141 of the Turkish Criminal Law. In the direction of article 141, we wanted to evaluate theft in 4 groups. However, since burglary does not always result in theft, we determined the number of our groups as 3. These groups are stealing, shoplifting, and robbery. To achieve this, firstly, we created datasets from the UCF-Crime dataset that contains three theft groups that are determined and normal videos. There is no publication in the literature on theft detection, considering novel datasets that contain three categories. Additionally, there are no publications in the literature about running several action recognition algorithms and making predictions using transfer learning as a result and using them as features. There are only a few products in this field that are commercially developed abroad and sold.

3. METHODOLOGY

Our methodology was organized into five main steps. Dataset development and collection is the first step, during which we acquired and selected the information required for analysis. In the second step, data transformation, we prepared the data. To find the most relevant parameters which would improve model performance, we then carried out feature selection. After that, we trained models and used a variety of classification algorithms. Lastly, we evaluated the model's performance to determine its overall efficiency and accuracy.

3.1. Dataset Collection and Creation

This study aims to identify theft in surveillance videos. To carry out this study, a dataset was developed. The section on data creation and gathering involved several procedures. Under subheadings, the specifics of these steps are arranged and clarified.

3.1.2. Theft Datasets

Following a comprehensive analysis of all theft and criminality-related research, we have found two open-source datasets that are significant to our study. A high-resolution, multi-camera dataset called the Charlotte Anomaly Dataset (CHAD) was created specifically for anomaly detection in commercial parking lots. Four cameras simultaneously capture the same scene: one HD camera and three Full-HD cameras. CHAD is made up of 412 high-quality videos totaling about 1.15 million frames. Among these, 1.09 million frames exhibit routine activity, while 59,000 frames contain anomalies (Danesh Pazho et al., 2022). These 412 videos had not been categorized. We didn't know which video had which anomaly. We just knew videos that contain anomalies. For this reason, we excluded and didn't utilize this dataset. UCF-Crime dataset is open-source, and it includes 13 different categories of anomalies related to

crimes, such as shoplifting, stealing, robbery, burglary, arrest, arson, assault, abuse, road accident, fighting, explosion, shooting, and vandalism. Additionally, there are videos called normal that don't have any anomalies. A significant collection of 128 hours of video makes up this dataset. It has 1900 untrimmed real-world surveillance videos (Sultani et al., n.d.). We used this dataset to construct our unique datasets. Our datasets include data gathered from four separate categories: shoplifting, stealing, robbery, and normal. We might have used the burglary category and the data in this category because we are interested in theft, but since burglaries are typically armed, after analyzing videos, we decided to consider the types of thefts that are carried out unarmed. We eliminated this category.

Figure 3.1. Shoplifting sample video view

We created three different datasets for our study. First, there is the NS dataset, which has 200 videos evenly split into two categories: 100 normal videos and 100 stealing videos. Expanding on this, we produced a second dataset called NSS, which offers three categories: 150 normal videos, 100 stealing videos, and 50 shoplifting videos. Lastly, an even more comprehensive third dataset, NSSR, includes four categories: 300 videos of normal, 50 videos of shoplifting, 100 videos of stealing, and 150 videos of robbery.

3.1.2. Pre-trained models

In our research, we used Apache MXNet, which is an open-source deep-learning framework. It enables neural networks with deep layers to be defined, trained, and used. It works with a wide range of platforms, including the infrastructure used by cloud computing. MXNet is well-known for its scalability, which is crucial for managing huge data sets and complicated models since it enables users to train models across distributed systems and many GPUs. Additionally, it serves as the foundation for several high-level libraries, including Gluon, which makes deep learning model construction and training easier. A compact, easy-to-understand API for deep learning is offered by the Gluon library in Apache MXNet. It enables the quick development, building, and training of deep learning structures without reducing training speed. State-of-the-art (SOTA) deep learning methods for computer vision are developed using GluonCV. Based on the Gluon interface of the Apache MXNet framework, GluonCV is an open-source computer vision library. It makes it simpler for developers and researchers to apply and experiment with cutting-edge techniques by offering a collection of pre-trained models, datasets, and tools made especially for computer vision work. To recognize and categorize actions in video sequences, GluonCV provides several action recognition models. Applications like sports analysis, video surveillance, and human-computer interaction depend on these models. Among the significant action recognition models in GluonCV are I3D and SlowFast Networks. By expanding 2D filters into 3D, I3D models enable 2D convolutional networks to learn spatiotemporal properties directly from video inputs. In tasks requiring action recognition, this method has significantly improved performance. SlowFast networks can efficiently capture both slow motion and quick motion by processing video at several temporal levels. While the Fast pathway concentrates on high-frequency movements, the Slow pathway records intricate spatial details. The models can be easily applied to user-provided datasets or customized for particular applications by using the pre-trained weights and implementations that GluonCV offers.

We used several GluonCV action recognition models. These models are an effective resource for anyone trying to identify actions in videos. The first is i3d_nl10_resnet101_v1_kinetics400. The Inflated 3D ConvNet (I3D) architecture is specifically implemented in the model i3d_nl10_resnet101_v1_kinetics400.

Its purpose is to recognize actions in video data. By extending 2D convolutional networks into 3D by inflating its filters, I3D, represented by the Inflated 3D ConvNet architecture, enables the model to learn spatiotemporal properties directly from video frames. This method efficiently records information about appearance as well as action. The model aims to employ a non-local operation using ten input frames, as denoted by the nl10 indicator, improving its capacity to capture long-range dependencies in the video data. This I3D model's foundation is built upon the 101-layer ResNet-101 design. ResNet, which is renowned for its effectiveness in image recognition applications, uses skip connections to lessen the impact of the gradient disappearing issue and make it possible to train deeper networks. V1: This abbreviation often denotes the ResNet architecture version that was utilized in the model. Kinetics400: This 400-class dataset, which is frequently used to train action recognition models, is utilized to pre-train the model. i3d_resnet50_v1_kinetics400 is the second one. The second action recognition model that we utilized was i3d_resnet50_v1_kinetics400. This model is an I3D model. It also uses the Kinetics 400 dataset. However, this model has resnet50. ResNet makes use of residual learning, a theory that facilitates the training of extremely deep networks. It uses shortcut connections, also called skip connections, to add the input to the result of a deeper layer without going through a few levels. Typically, images with a resolution of 224 by 224 pixels are fed into ResNet-50. For tasks involving classification, the output is typically a probability distribution across a collection of classes. The utilized third action recognition model was slowfast_4x16_resnet50_kinetics400. It is a SlowFast model architecture. 4x16 represents the video input sampling structure. That usually means that in this instance, the Fast pathway processes frames at a rate of 16 frames per second, whereas the Slow pathway processes frames at a rate of 4. This design facilitates the balancing of motion capture and detail. The Kinetics 400 dataset and resnet50 architecture are used in this model. This technique recognizes motions in videos by efficiently utilizing both slow and fast motion information. We used the slowfast_8x8_resnet101_kinetics400 model. This model utilizes the SlowFast architecture, too. 8x8 indicates the model's sample method. In particular, it usually indicates that the Fast pathway functions at a rate of 8 frames per second, enabling a more in-depth comprehension of both fast and slow movements. The Slow pathway analyzes video frames at a rate of 8 frames per second.

The Kinetics 400 dataset and resnet101 architecture are used in this model. This model is an advanced model for identifying actions in videos. It uses a dual-stream architecture to leverage both spatial and temporal information, and it uses a strong backbone (ResNet101) that has been trained on a large dataset (Kinetics-400). If we explain the Kinetics400 dataset in more detail, a collection of original action videos from YouTube is called Kinetics400, and it is used for action recognition. With 306,245 short-trimmed movies from 400 action categories, it is one of the largest and most often used datasets in the academic area for evaluating the effectiveness of the most sophisticated video action detection models. Usually, each video clip lasts for ten seconds or less. It offers enough time information to accurately represent the actions' dynamics. Action classes are used to label the dataset, enabling models trained on it to identify particular actions in videos. It is widely used for deep learning model training and evaluation, particularly for models with complex architectures.

3.1.3. Feature creation

To generate features, firstly, we developed a Python script that takes input in the form of text files ending in .txt. The path to a unique video, its frame count, and its associated video name are all included in each row of this text file. The last two fields, which are just placeholders and do not function in the code, are the number of frames and the video label. As a result, to satisfy the script's input requirements, arbitrary positive integers were inserted into these fields. The script returns human activity predictions in videos specified in a text file. We have run 4 GluonCV action recognition models across three distinct datasets with the usage of this script. A total of 12 distinct feature datasets were produced using this process. For ease of identification and further analysis inside our study, each of these feature datasets has been named systematically. The names of the feature datasets are as follows, in order. NS - i3d - nl10 - resnet101- v1- kinetics400, NS - i3d - resnet50 - v1- kinetics400, NS - slowfast - 4x16 - resnet50 - kinetics400, NS - slowfast - 8x8 - resnet101 - kinetics400, NSS - i3d - nl10 - resnet101- v1- kinetics400, NSS - i3d - resnet50 - v1- kinetics400, NSS - slowfast - 4x16 - resnet50 - kinetics400, NSS - slowfast - 8x8 - resnet101 - kinetics400, NSSR - i3d - nl10 - resnet101- v1- kinetics400, NSSR - i3d - resnet50 - v1- kinetics400, NSSR - slowfast - 4x16 - resnet50 - kinetics400, NSSR - slowfast - 8x8 - resnet101 - kinetics400. NS refers to the dataset containing Normal and Stealing videos, while

NSS includes Normal, Stealing, and Shoplifting videos. NNSR includes Normal, Stealing, Shoplifting and Robbery videos. The remaining part of the dataset's name is the name of the pre-trained model. In each feature dataset, each row contains the category of the video and a series of action inference probability values that are 400 vectors. The categorical label (theft/normal) of the video is indicated by the first value in each entry, and the next values are the inference probabilities over 400 different action classes. There is a table that includes 400 different action classes in the Appendix B section.

3.2. Data Transformation

In this study, we used several data transformation techniques, such as binarization, max absolute scaling, and Z-score normalization (standard scaling). Standardization is a statistical technique that rescales and modifies the distribution of data to yield a mean of zero and a standard deviation of one. Z-score normalization was the first preprocessing method used on the data. This preprocessing stage successfully changed every value in the dataset, guaranteeing that the distribution that resulted had a standard deviation of 1 and a mean of 0. This type of it reduces the impact of various scales on various variables, facilitates feature comparison, and increases the effectiveness of machine learning algorithms. The data distribution becomes more stable and comparable as a result. Furthermore, it increases machine learning algorithms' effectiveness by decreasing their sensitivity to the size of the input features. Additionally, during the preprocessing stage of the analysis, the data normalization method known as max absolute scaling was utilized. To scale features to a range between -1 and 1, max absolute scaling is a data normalization approach that takes into account the maximum absolute value of each feature. When working with datasets that have both positive and negative values in their characteristics, this method is especially helpful. After these data transformations, binarization was used. The process of converting numerical data into binary form is known as binarization.

The 0.001, 0.0001, 0.00001, and 0.000001 are the four threshold values that have been established. The datasets were binarized using these thresholds, converting continuous feature values into binary representations that met the preset standards.

In this study, we employed the Scikit-learn package, an open-source Python machine learning and data modeling library, for data transformations. The scikit-learn version was 1.2.0. We constructed a StandardScaler object for Z-score normalization. Using the fit_transform method from the StandardScaler object, the standard deviation and mean for each feature were specified to apply Z-score normalization. To apply max absolute scaling, we created an instance of the MaxAbsScaler. Max absolute scaling was applied by calculating the maximum absolute value for each feature with the fit_transform method that is from the MaxAbsScaler object.

3.3. Feature Selection

Feature selection is the process of identifying and selecting a subset of relevant attributes (or variables) from the larger range of features in a dataset. Feature selection is primarily used to reduce overfitting, increase accuracy, and shorten training times in machine learning models. In this study, we used the Scikit-learn library for feature selection operations. The subsections provide a detailed explanation of the advancements we made in our study about feature selection.

3.3.1. Filter-based feature selection

Feature selection methods that rely on filters assess the importance of features by analyzing their statistical characteristics, regardless of machine learning algorithms. They can be scaled and are fast. SelectKBest is included in them since it uses a scoring function for evaluating features. To evaluate features, SelectKBest frequently uses statistical tests such as correlation coefficients (for regression) or the ANOVA F-test (for classification). Based on the selected criterion, that is, f_classif for classification tasks (ANOVA F-value), this approach scores and then chooses the top K features with the highest scores. As might be expected, this method accepts two different parameters, which are "scoring function" and "k."

The integer "k" indicates how many top characteristics to choose. The default for k is 10. Some of the scoring functions (f_classif, chi2, and f_regression) are explained. f_classif is ANOVA F-value between label/feature. Each feature's linear relationship to the target variable is measured by this function, which is utilized for classification tasks. It facilitates the identification of characteristics that differ noticeably between classes. chi2 is chi-squared statistics of non-negative features. This evaluates the independence of features about the target variable and is applied to categorical characteristics. It evaluates if there is a difference between the distribution of categorical variables and what would be predicted if they were independent. f_regreesion is the F-value between label/feature. It is used for regression tasks. To find features that significantly affect the prediction of continuous outcomes, this calculates the linear correlation between the target variable and each feature.

The SelectKBest feature selection approach is applied in our study. To evaluate the impact of various feature subsets, we applied this approach with three distinct K values. These are 20, 30, and 50. We choose to use the ANOVA F-value (f_classif) for the scoring function because it is especially appropriate for classification tasks. We implemented this selection method because it has the advantage of handling big datasets efficiently while preserving computing efficiency. Furthermore, its simple implementation makes it easy to interpret.

3.3.2. Feature importance quantification

Feature importance quantifies each feature's contribution to a machine learning model's prediction performance. Gaining knowledge about the underlying data and increasing model efficiency can be achieved by determining which attributes have the most influence. There are several approaches to assess a feature's importance. These are coefficient-based, tree-based, or permutation importance. In this research, we used a based feature importance method that is a random forest to calculate. A method for assessing the worth of features in decision tree algorithms, such as random forests and decision trees, is called tree-based feature importance. To increase accuracy and manage overfitting, random forests, a collaborative learning technique, generate numerous decision trees and aggregate their predictions.

Random forest features are valued using Mean Decrease Impurity (MDI), a critical technique for assessing each feature's impact on a random forest machine learning model's predictive power. The quality of a split is assessed in decision trees using impurity metrics such as entropy or Gini impurity. There is a better split when the impurity is lower. Every decision tree in the random forest helps to lower the node's impurity when a feature is utilized to split the tree. Each time a feature is utilized in a split, the magnitude of the impurity reduction it achieves is noted. Once every tree has been constructed, the overall impurity decrease for every feature has been added up. To obtain a proportionate score, the feature importance score for each feature is then normalized, often by dividing by the total impurity reduction across all features. Each feature's proportion to the model's overall decision-making process is shown by the score that results. The more important a feature is in establishing predictions, the higher its score. All things considered, random forest feature significance is a useful model assessment. Interpreting the model and learning more about the data is made easier by knowing which features have the biggest influence on the predictions. It assists in choosing the most essential aspects, which improve model performance and simplify it. Underlying connections between features and the target variable are revealed by feature importance.

Using the scikit-learn library to compute random forest feature importance, firstly, we created and fitted the random forest model. We got feature importances using feature_ importances attributes directly from the fitted model. It returns the importance scores. Then, we accessed the important feature names. Additionally, we visualized feature importance using a bar plot.

3.3.3. Feature extraction

In the machine learning process, feature extraction is an essential phase that turns raw data into a collection of useful features. With the help of these features, predictive models are constructed, allowing algorithms to recognize patterns and generate accurate predictions. The goal of feature extraction is to enhance the quality and usefulness of input data while decreasing dimensionality. PCA, or principal component analysis, is a feature extraction technique. This dimensionality reduction technique is commonly used to reduce the number of variables in huge data sets while

maintaining much of the original data. Accuracy naturally decreases as a data set's variables are reduced, but the key to dimensionality reduction is to compromise a little accuracy in favor of simplicity. As a result of smaller data sets being simpler to examine and visualize, machine learning algorithms can analyze data points considerably more quickly and easily because fewer variables need to be processed.

Using the scikit-learn library to apply PCA, we created a PCA object. We identified the number of components to keep. Then, we fitted and transformed the data. The number of components was taken as 5, 10, 15, and 20, and experiments were carried out on the data sets, respectively.

3.4. Classification Algorithms and Model Training

Classification machine learning algorithms to be utilized for model training were identified. KNNeighbors, Gaussian Naive Bayes, Decision Trees, Support Vector Machines, Random Forests, Neural Networks, and Gradient Boosting are among the machine learning techniques that have been specified. A supervised machine learning algorithm known as a support vector machine (SVM) classifies data by identifying the best line or hyperplane in an N-dimensional space that optimizes the distance among each class. It works well in high-dimensional spaces, is sensitive to overfitting, and is adaptable because of its kernel functions (linear, RBF, etc.). One supervised learning technique for classification is the Decision Tree algorithm. It displays choices and their potential outcomes as a tree-like structure, with each internal node standing for a feature-based decision, each branch for the decision's result, and each leaf node for a class label. The objective is to learn basic decision rules derived from the data features to build a model that estimates the value of a target variable. It is easy to understand and interpret. In addition to handling both numerical and categorical data, it requires minimal data preprocessing. Based on the idea that comparable data points are likely to provide similar results, the supervised classification method is the K-Nearest Neighbors (KNN) machine learning algorithm. As an illustration of instance-based learning, KNN does not build an explicit model; instead, it bases its predictions on the complete training dataset. Achieving optimal performance requires careful evaluation of the distance metric and K selection.

The K parameter indicates how many nearest neighbors should be considered when making a prediction. To assess the "closeness" of data points, KNN uses distance measurements.

One ensemble learning technique that is often utilized for classification is the Random Forest algorithm. To increase accuracy and manage overfitting, it aggregates the predictions of several decision trees. The Random Forest algorithm relies on a set of decision trees. Utilizing a random portion of the training data, each tree is trained to produce a variety of models that can identify various patterns. By utilizing the advantages of several decision trees, it mitigates the weaknesses of individual models while achieving high accuracy and robustness. For classification tasks, Gaussian Naive Bayes is a probabilistic machine learning method based on the Bayes principle. The term "Gaussian" denotes the assumption that the characteristics have a Gaussian (normal) distribution. A simple and effective approach for classification problems, especially when working with big datasets, is Gaussian Naive Bayes. Gradient Boosting, an ensemble method that creates models one after the other in an attempt to correct the errors made by the previous models used for classification problems, is one of the most effective machine learning algorithms. Gradient Boosting is able to generate extremely accurate predictions by integrating the results of these models. A neural network is a classification model for machine learning that is based on the structure and functions of the human brain. In this model, there are some key components. Some of them are layers and activation functions. The layer that gets the input data is called the input layer. Weighted connections are used in the processing of hidden layers, which are intermediate layers. The final classifications or predictions are generated by the output layer. Since activation functions establish each neuron's output and add non-linearity to the model, they are essential parts of neural networks. Neural networks are capable of learning intricate patterns and relationships in the data because of this non-linearity. Twelve feature datasets were trained using all the machine learning algorithms we mentioned above. Then, the datasets were re-trained by applying all preprocesses steps.

3.5. Performance Evaluation

After the model training, we assessed the model's performance using evaluation metrics that are AUC, precision, accuracy, recall, and F1 score. The following paragraphs explain the definitions and formulas of evaluation metrics.

Accuracy, which is a major measurement metric, is utilized for assessing the efficiency of classification models. It refers to the proportion of accurate predictions to all predictions. This assessment indicates that if 100 predictions were made, 70 of them would have come true, indicating a 70% accuracy rate for the model.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Recall is utilized as a performance metric to assess the classification model's efficiency. It is also named true positive rate or sensitivity. It is obtained with true positive predictions divided by the sum of true positive and false negative predictions. It shows the model's ability to identify all relevant instances. If the model determines 60 out of 100 true positive cases correctly, the recall will be 60%.

$$\text{Recall} = \text{Sensitivity} = TPR = \frac{TP}{TP + FN}$$

Precision is a crucial measurement metric. By showing the proportion of expected positive cases that are true positives, it evaluates how accurate the model is at making positive predictions. The model's capacity to accurately specify pertinent instances is indicated by the ratio of true positive predictions to the sum of true positive and false positive predictions. For instance, a model's precision is 0.6 (or 60%) if it predicted 50 positive instances, but just 30 of those were accurate.

$$\text{Precision} = \frac{TP}{TP + FP}$$

The F1 score, which provides a single measure that balances precision and recall, is the fundamental evaluation metric. The F1 score is calculated by taking the harmonic mean of recall and precision metrics. Its range is between 0 and 1. While 0 shows the worst performance, 1 shows the best precision and recall.

$$\text{F1-Score} = \frac{2 * Precision * Recall}{Precision + Recall}$$

The Area Under the Curve assesses the ability of the models to discriminate between positive and negative classes with different thresholds. This crucial performance metric is the area under the Receiver Operating Characteristic (ROC) curve. It is referred to as AUC. Its values vary from 0.5, which indicates random performance, to 1.0, which indicates excellent categorization. If the model discriminates positive and negative classes better, the AUC will be higher. If AUC is 70%, it shows that the possibility of a positive instance chosen at random having a higher estimated value than a negative instance chosen at random is 70%.

$$AUC = \int_{x=0}^{1} TPR(FPR^{-1}(x))dx$$

Python libraries were utilized for model training and performance evaluation. The scikit-learn, which is a machine learning library, was used. Its version was 1.2.0. Pandas, which is a data analysis library, was used. The version of the panda's library was 1.5.2. A Python library for data visualization called Seaborn was used. Its version was 0.12.2. NumPy, which is a Python library for scientific computing, was also used. The version of NumPy was 1.24.1. The extensive Python visualization library Matplotlib was utilized to create static, animated, and interactive displays. The version used is 3.6.2. Also, the Python version was 3.10.

3.5.1. Hyperparameterazation

First, each of the 12 datasets was divided into train and test sets. The test size was 20 percent of the dataset, while 80 percent of the dataset was for the training part. This separation is essential for the later assessment of model performance since it enables the training set to be used for model fitting and parameter adjustment and the testing set to be used as a separate sample for evaluating how well the created models generalize. After the division of datasets as train and test, we used 5-fold cross-

validation in the training part. It is a machine learning model training and evaluation technique. By ensuring that the model is trained on many subsets of the data and reducing the overfitting probability, this approach produces a more accurate assessment of the model's performance. We utilized the scikit-learn library's cross-validation technique to apply 5-fold cross-validation in the training section. The function enables the computation of multiple measurements simultaneously and is helpful for cross-validation. We obtained the average of 5-fold train performance results on 80 percent of datasets. Then, we trained 80% of the datasets with the appropriate classification models. After this process, we predicted it with the test datasets and generated the performance results.

In training, we used multiple machine learning algorithms that were mentioned in the classification algorithms and model training part. For each machine learning algorithm, we utilized different parameter sets. To use algorithms from scikit-learn, firstly, we imported associated library lines. Then, we created classifiers and set the parameters of classifiers. The following paragraphs explain the parameters of these machine learning algorithms in the scikit-learn library. These parameters are the parameters used in this research.

In the SVM algorithm, four important parameters are used. The trade-off between maximizing the margin and decreasing classification error is managed by the regularization parameter that is named "C". It needs to be completely positive. A high C value aims to accurately classify every training example, whereas a small C value permits more misclassifications. The default value is one for C. "kernel" designates the kind of kernel that will be utilized in the algorithm. These include linear, poly, rbf, sigmoid, and precomputed. If none is provided, rbf is used by default. To facilitate class separation, kernels convert the input data into higher-dimensional space. "gamma" is the kernel coefficient for kernels of values rbf, poly, and sigmoid. It takes two values that are scale, auto, or float. The default is scale. According to these, the value of gamma is assigned. "degree" refers to the polynomial kernel function's degree (or "poly"). It must not be negative. All other kernels disregard it. The default value of degree is three.

In the Decision Tree algorithm, five crucial parameters are considered. The first parameter is "criterion," which specifies the function to measure the quality of a split. Criterion values are gini, entropy, log_loss. The default value of the criterion parameter is gini. The "splitter" parameter is utilized at each node to select the split. Values of this parameter are best and random. Best is used for selecting the best split, and random is used for selecting the best random split. The default value is best. The argument "max_depth" indicates the tree's maximum depth. It limits the number of splits in the tree. Its values can be integer. The default value is None. This parameter helps prevent overfitting. "min_samples_leaf" is another important option that is essential. It indicates the least amount of samples required at a leaf node. Values of this parameter can be integer or float. The default value is 2. By keeping the model from learning too particular patterns, it aids in smoothing the model. The "random_state" parameter regulates the estimator's level of randomness. Its values are integer or None. The default value is None.

"n_neighbors" was the first important parameter in the K-Nearest Neighbors (KNN) method that determined how many neighbors would be utilized for classification. Its values are integer. The default value is 5. Usually, values are odd numbers to prevent classification ties. To compute the nearest neighbors, an "algorithm" parameter is utilized. Values of the parameter are "auto", "ball_tree", "kd_tree", and "brute". The default is "auto". The leaf size provided to the KDTree or BallTree algorithms is known as the "leaf_size" parameter. This may impact memory utilization and the speed of the query and building. When determining the distance between two points, the distance metric is defined by the "p" parameter. Generally, p is 1 for Manhattan, and p is 2 for Euclidean distance. These are the parameters that are most frequently utilized.

"n_estimators," or the entire number of trees in the forest, was the first important parameter in the Random Forest algorithm. Values of parameters are integer. The default is 100. More trees generally increase both performance and computation time. The next parameter, "max_features", indicates how many features should be taken into consideration when determining the optimal split. Parameter values can be integer, float, sqrt, log2, or None. The default value is None, which means all features are considered. Other parameters that we also mentioned in the decision tree classifier are "criterion" and "min_samples_leaf".

In Gaussian Naive Bayes, it had one crucial parameter, that is "var_smoothing". We adjusted this parameter. This parameter enhances numerical stability by adding a small amount of variance to the features. When a feature's variation is relatively minimal, it helps avoid zero division errors.

In Gradient Boosting algorithm has six important parameters that are considered. The first parameter is "n_estimators", which is the quantity of boosting stages(trees) that must be executed. Although they can improve accuracy, more boosting stages run the danger of overfitting. Its values are integer. The default is 100. The next crucial parameter is "max_depth", which is the deepest point of each individual tree. With the use of this parameter, overfitting can be avoided; deeper trees can represent more complex relationships. Values can be integer or None. The default value is 3. Another parameter is "min_samples_split", which is the bare minimum of samples needed to separate an internal node. Values are integer and float. The default is 2. "min_samples_leaf" is the minimum number of samples needed to reach a leaf node. It aids in reducing overfitting. Values can be integer or float. The default is 1. "subsample" is the fraction of samples that will be utilized to fit each base learner individually. Values are float. A value below 1.0 that is default can aid in avoiding overfitting. "max_features" is the number of features to take into account when trying to find the ideal split. Values can be "sqrt", "log2", integer, float, or None. The default value is None.

Three key parameters are taken into consideration in the Neural Network algorithm. This variable "hidden_layer_sizes" describes how many neurons are hidden in each layer. Values are array-like (n_layers,). The default is (100,). The next parameter is "activation", which is the function that activates the hidden layers. Values are "identity", "logistic", "tanh", and "relu". The default is "relu". The last parameter is "max_iter", which is the most iterations (epochs) that can be used to train the model. If the loss doesn't get better, the solver might quit sooner. Values are integer. The default value is 200.

Taking into account the parameters of the algorithms we mentioned in the above paragraphs, we determined the best parameter sets and fine-tuned them accordingly in the hyperparameterization step. Subsequently, the best classification algorithm models and parameter sets were determined based on their AUC, F1 score, accuracy, precision, and recall over the test sets. The best-performing algorithms and hyperparameters are shown in Table 1.1. All machine learning algorithms used in this study and their hyperparameters are included in the Appendix C section.

Table 3.1. Machine learning algorithms and best parameter sets

Machine learning algorithm	Parameter set
Neural Network	{'hidden_layer_sizes': (50, 50), 'activation': 'relu', 'max_iter': 50}
Random Forest	{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 5}
Gradient Boosting	{'n_estimators': 300, 'max_features': 'log2', 'max_depth': 3, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}
KNeighbors	{'leaf_size': 40, 'n_neighbors': 7, 'p': 2, 'algorithm': 'auto'}
Decision Tree	{'criterion': 'gini', 'random_state': 0, 'max_depth': None, 'min_samples_leaf': 1, 'splitter': 'best'}
Gaussian Naive Bayes	{'var_smoothing': 1e-12}
Support Vector Machine	{'kernel': 'rbf', 'gamma': 1, 'C': 1.0}

4. RESULTS

4.1. Effects of Data Transformation

In this research, we performed data transformations to all datasets before the execution of machine learning algorithms. We found that these transformations contribute to improving machine learning algorithms' performances. Since we know how well a model can learn from the data, how interpretable the model is, and the overall quality of predictions directly depends on the preprocessing steps to be applied, we carefully tried different preprocessing techniques and recorded the results by selecting the ones that were suitable for our study. In our study, we tried to make the correct choice and application of data transformations important and to make them effective on the results.

The results show that z-score normalization is quite useful in improving performance in the Gaussian Naive Bayes model. Because normalization makes sure that features are on the same scale and are more likely to satisfy the Gaussian distribution assumption, it enhances performance. The model's performance has improved in comparison to the pre-normalization level. The results show that max absolute scaling applied to the datasets is more useful in the Neural Network model. Because this model is sensitive to the magnitude of the data but does not require the data to be shifted to zero or restricted to a specific range. It is also seen that the performance is better after applying maximum absolute scaling in Gradient Boosting and Random Forest models. Binarization has been applied with various threshold values, and among these threshold values, 0.0001 has given the best results. The second best threshold was 0.00001. After applying binarization, the most effective results have been obtained with Random Forests, Neural Network, and Gradient Boosting models.

4.2. Effects of Feature Selection

In this study, we implemented feature importance, feature selection, and PCA that play crucial roles in improving model performance, reducing overfitting, and enhancing interpretability.

The results show us that when we apply feature selection, the model focuses on the most important factors and increases the prediction accuracy as we remove irrelevant features. At the same time, the complexity of the model is reduced. We observed that training times were reduced as the number of features decreased. Hyperparameters automatically accelerated the optimization time. When the results obtained after applying feature selection are compared, the best algorithms for each data set are explained with their values. The best classification algorithm models were determined based on their AUC, recall, precision, accuracy, and F1 score on the test set. The model with the highest performance was the Random Forest with an AUC of 0.88, accuracy of 0.88, precision of 0.88, recall of 0.88, and F1 score of 0.88 on NS - i3d - nl10 - resnet101- v1- kinetics400 dataset with used SelectKBest method that K is 20. The best was the Random Forest with an AUC of 0.85, accuracy of 0.85, precision of 0.85, recall of 0.86, and F1 score of 0.85 on NS - i3d - resnet50 - v1- kinetics400 with used SelectKBest method that K is 30. The best was the Gaussian Naive Bayes with an AUC of 0.84, accuracy of 0.83, precision of 0.83, recall of 0.87, and F1 score of 0.82 on NS - slowfast - 4x16 - resnet50 - kinetics400 with same method and K value. The best was the Gaussian Naive Bayes with an AUC of 0.81, accuracy of 0.80, precision of 0.82, recall of 0.80, and F1 score of 0.80 on NS - slowfast - 8x8 - resnet101 - kinetics400 with used SelectKBest method that K is 50. The best was the Decision Tree with an AUC of 0.75, accuracy of 0.75, precision of 0.75, recall of 0.75, and F1 score of 0.75 on NSS - i3d - nl10 - resnet101- v1- kinetics400 with used SelectKBest method that K is 20. The best was the Random Forest with an AUC of 0.77, accuracy of 0.77, precision of 0.77, recall of 0.77, and F1 score of 0.77 on NSS - i3d - resnet50 - v1- kinetics400 with used SelectKBest method that K is 30. The best was the Random Forest with an AUC of 0.83, accuracy of 0.83, precision of 0.83, recall of 0.84, and F1 score of 0.83 on NSS - slowfast - 4x16 - resnet50 - kinetics400 with used SelectKBest method that K is 50.

The best was the Random Forest with an AUC of 0.72, accuracy of 0.72, precision of 0.74, recall of 0.72, and F1 score of 0.71 on NSS - slowfast - 8x8 - resnet101 - kinetics400 with the same method and K value. The best-performing model was the Gradient Boosting with an AUC of 0.73, accuracy of 0.73, precision of 0.73, recall of 0.73 and F1 score of 0.73 on NSSR - i3d - nl10 - resnet101- v1- kinetics400 with the same method and K value. The best was the Gradient Boosting with an AUC of 0.73, accuracy of 0.73, precision of 0.73, recall of 0.74, and F1 score of 0.73 on NSSR - i3d - resnet50 - v1- kinetics400 with the same method and K value. The best was the Gradient Boosting with an AUC of 0.80, accuracy of 0.79, precision of 0.79, recall of 0.80, and F1 score of 0.79 on NSSR - slowfast - 4x16 - resnet50 - kinetics400 with used SelectKBest method that K is 20. The best was the Gradient Boosting with an AUC of 0.71, accuracy of 0.71, precision of 0.72, recall of 0.71, and F1 score of 0.71 on NSSR - slowfast - 8x8 - resnet101 - kinetics400 with used SelectKBest method that K is 30. It is inferred from the findings that the best results of using the SelectKBest method are obtained by using K 50. The best model is seen as a Random Forest after feature selection is applied. In addition, Gradient Boosting, Gaussian Naive Bayes, and Decision Tree are the other best-performing models.

Feature importance was applied to all datasets. The results indicated which feature was more important. Since these features corresponded to actions in our dataset, they demonstrated which action had a better score in which theft category. We used random forest feature importance, and this returned importance scores. We obtained importance scores for all datasets. By analyzing feature importance scores, we identified the two most significant actions for each dataset. Below, we detail these datasets along with their top-ranked actions. For the NS-i3d-nl10-resnet101-v1-kinetics400, the most important actions are pumping_gas and pushing_car. The NS-i3d-resnet50-v1-kinetics400 highlights pumping_gas and building_cabinet, while the NS-slowfast-4x16-resnet50-kinetics400 emphasizes changing_wheel and bookbinding. In the case of the NS-slowfast-8x8-resnet101-kinetics400, pumping_gas, and changing_wheel are the most critical. Pumping_gas and answering_questions are notable actions for the NSS-i3d-nl10-resnet101-v1-kinetics400, whereas pumping_gas and changing_wheel are important actions for the NSS-i3d-resnet50-v1-kinetics400.

Pumping_gas and changing_wheel are highlighted in NSS-slowfast-8x8-resnet101-kinetics400, whereas changing_wheel and pumping_gas are ranked as the top actions in the NSS-slowfast-4x16-resnet50-kinetics400 dataset. In the NSSR-i3d-nl10-resnet101-v1-kinetics400, texting and extinguishing_fire are the most important, followed by texting and motorcycling in the NSSR-i3d-resnet50-v1-kinetics400. The NSSR-slowfast-4x16-resnet50-kinetics400 prioritizes news_anchoring and riding_unicycle, while assembling_computer and motorcycling are the key actions for NSSR-slowfast-8x8-resnet101-kinetics400. To visually present these findings, we have created bar plots for the feature importance of each dataset, which are shown below.

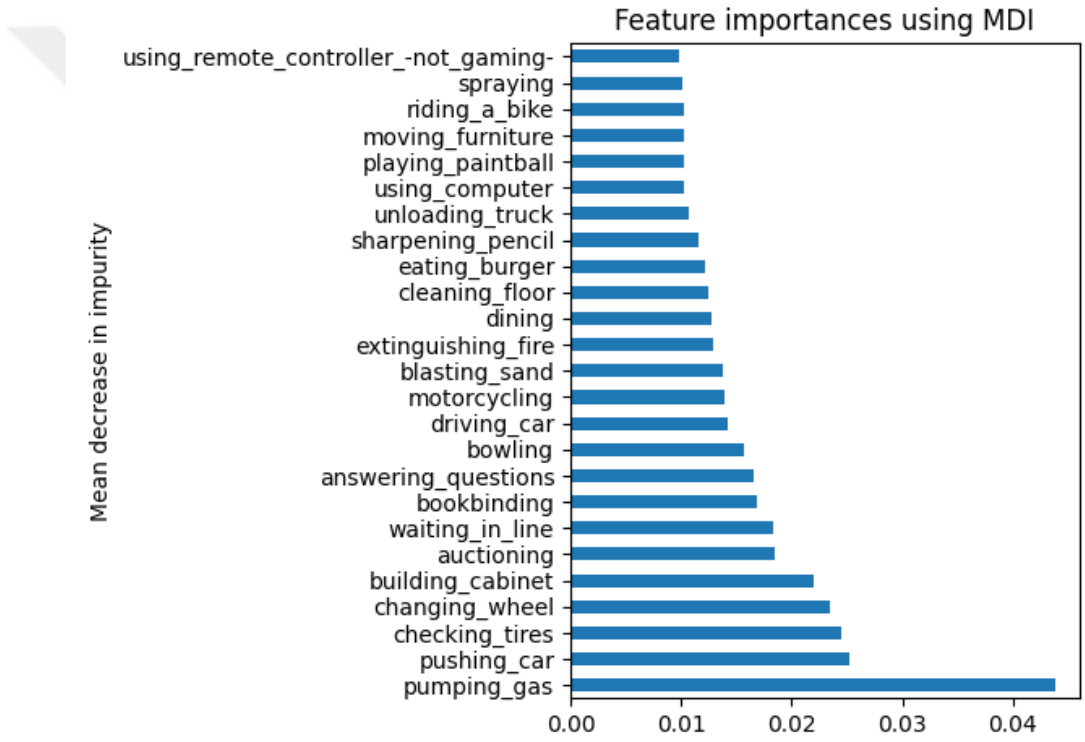

Figure 3.2. NS-i3d-nl10-resnet101-v1-kinetics400 dataset feature importance plot

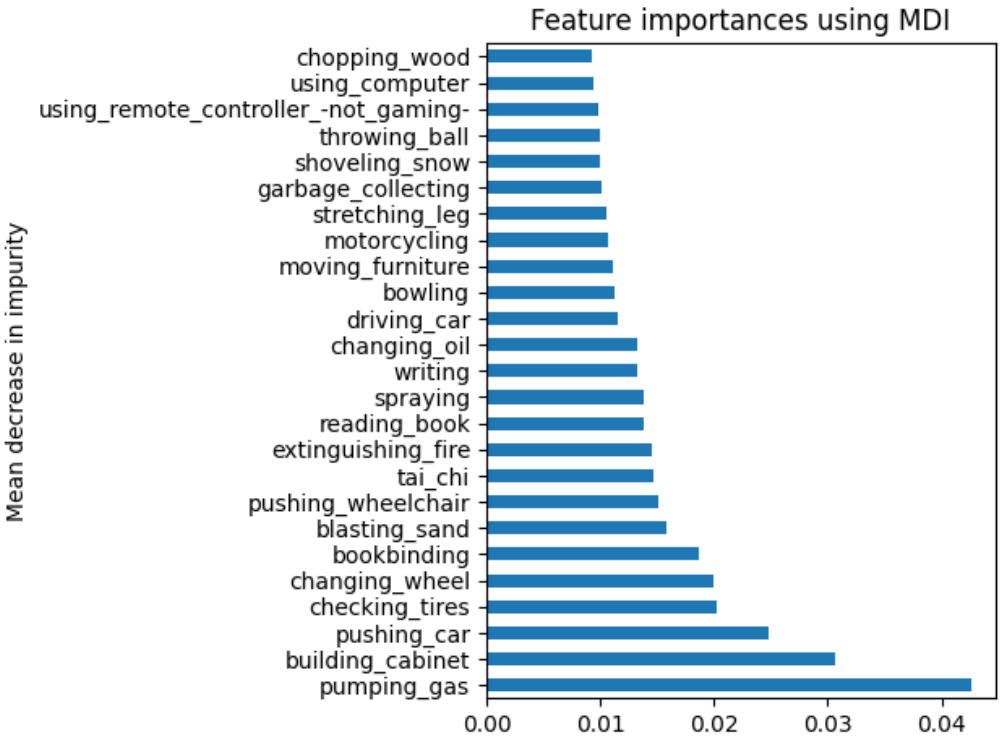

Figure 3.3. NS-i3d-resnet50-v1-kinetics400 dataset feature importance plot

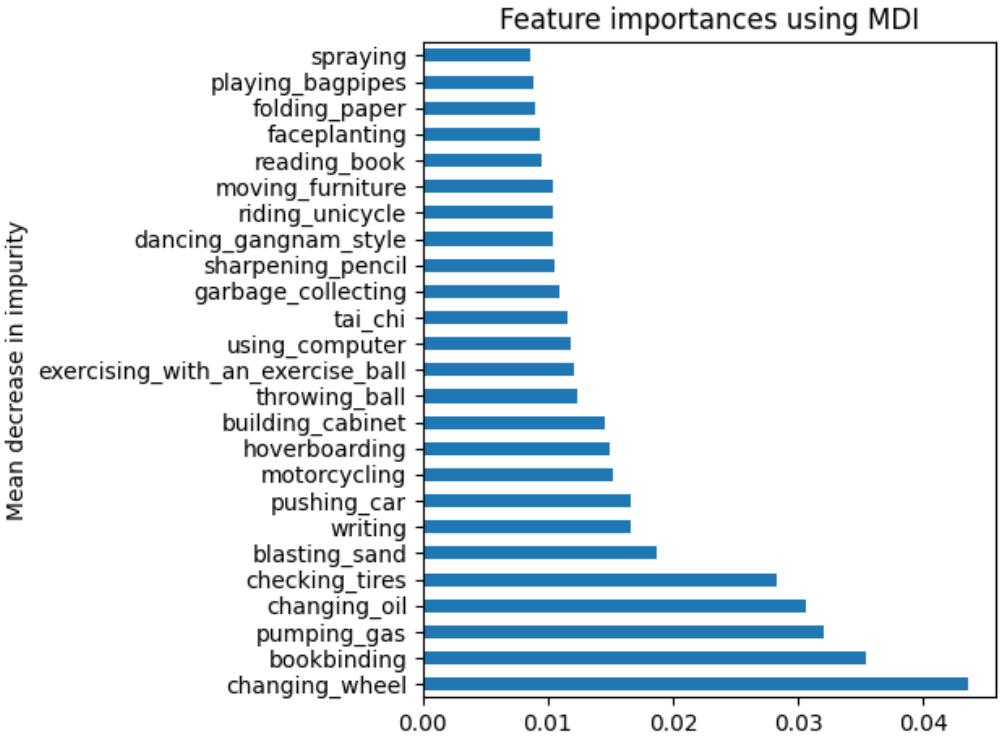

Figure 3.4. NS-slowfast-4x16-resnet50-kinetics400 dataset feature importance plot

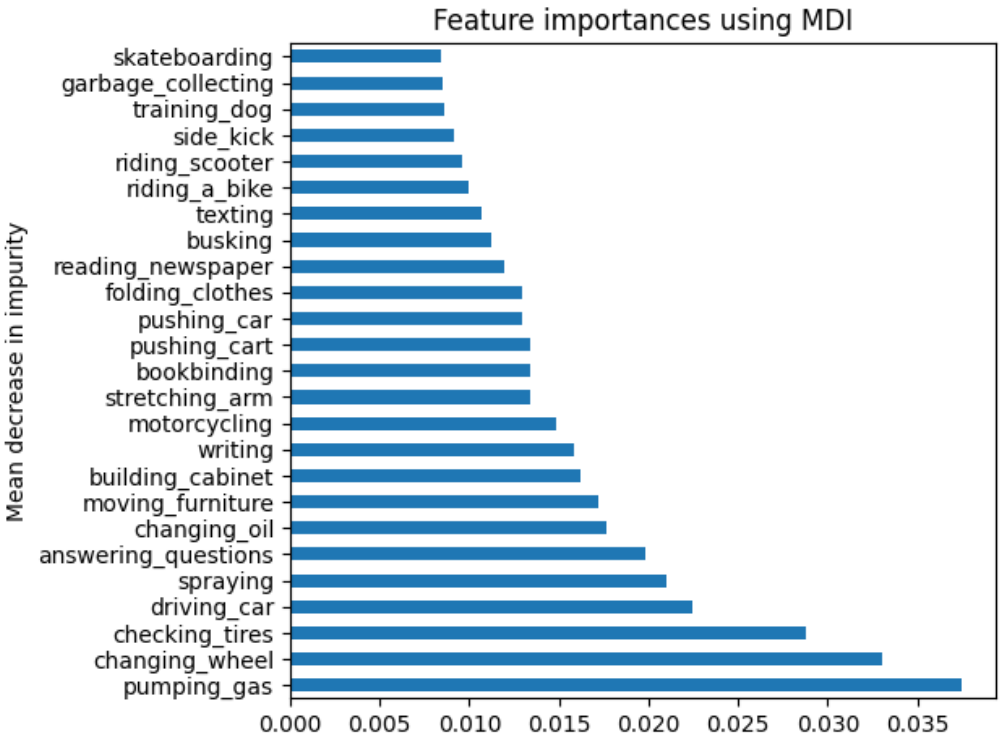

Figure 3.5. NS-slowfast-8x8-resnet101-kinetics400 dataset feature importance plot

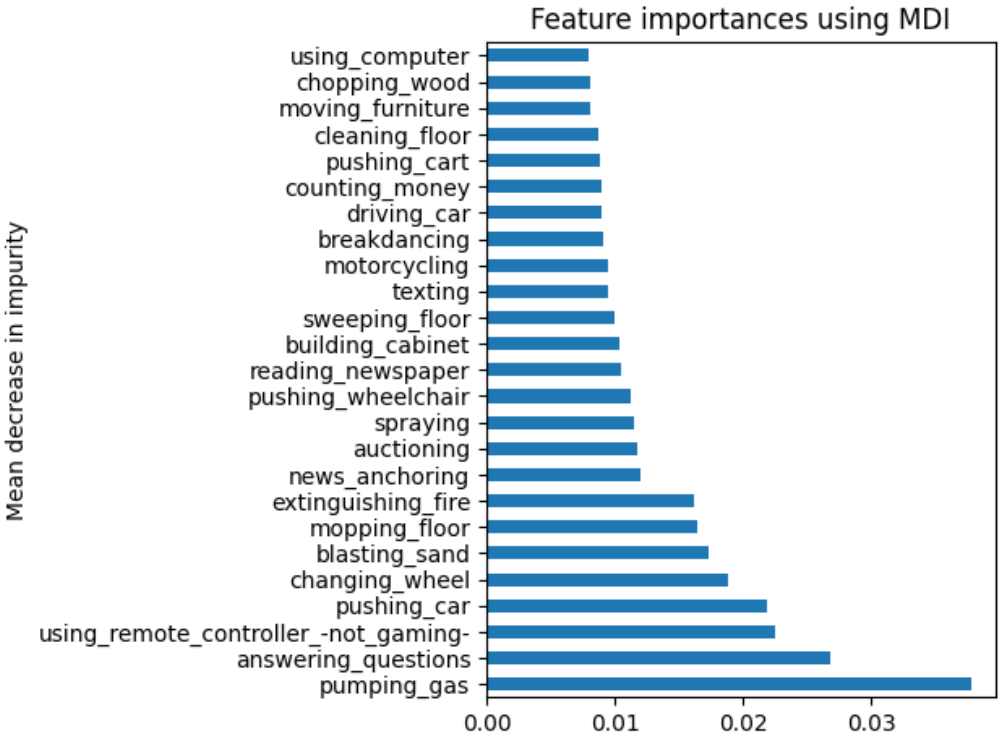

Figure 3.6. NSS-i3d-nl10-resnet101-v1-kinetics400 dataset feature importance plot

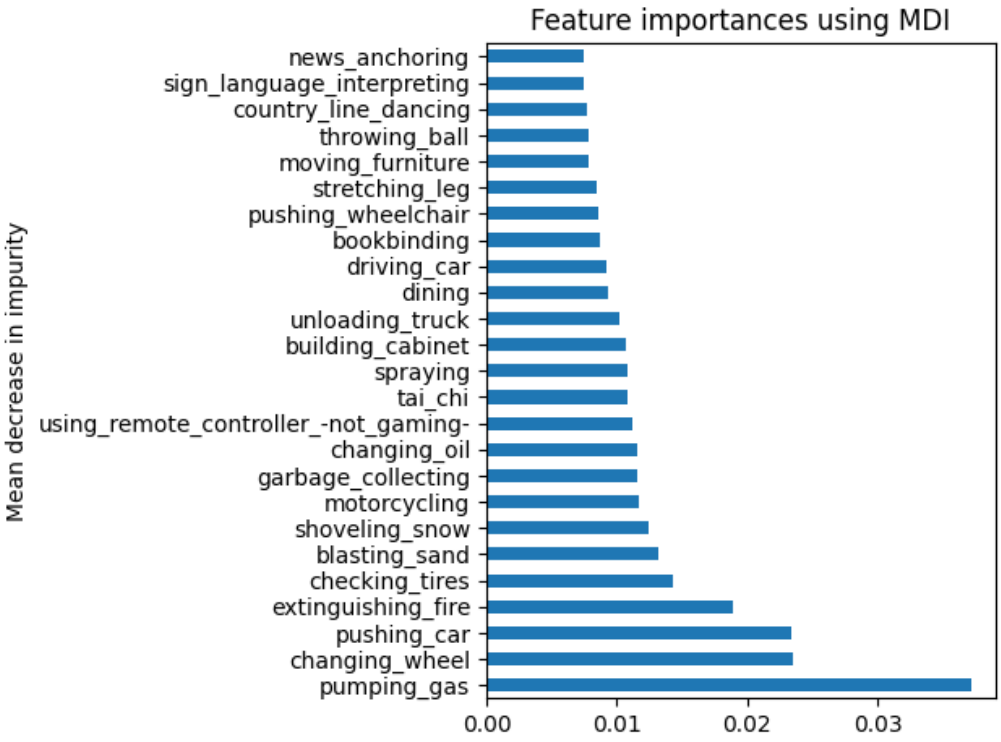

Figure 3.7. NSS-i3d-resnet50-v1-kinetics400 dataset feature importance plot

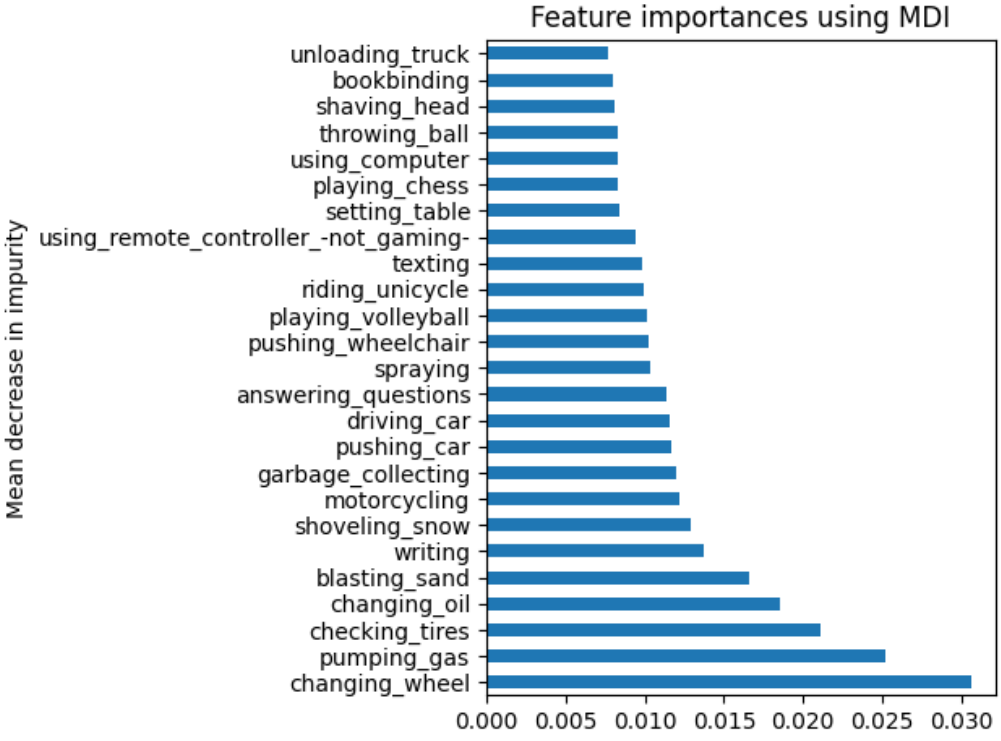

Figure 3.8. NSS-slowfast-4x16-resnet50-kinetics400 dataset feature importance plot

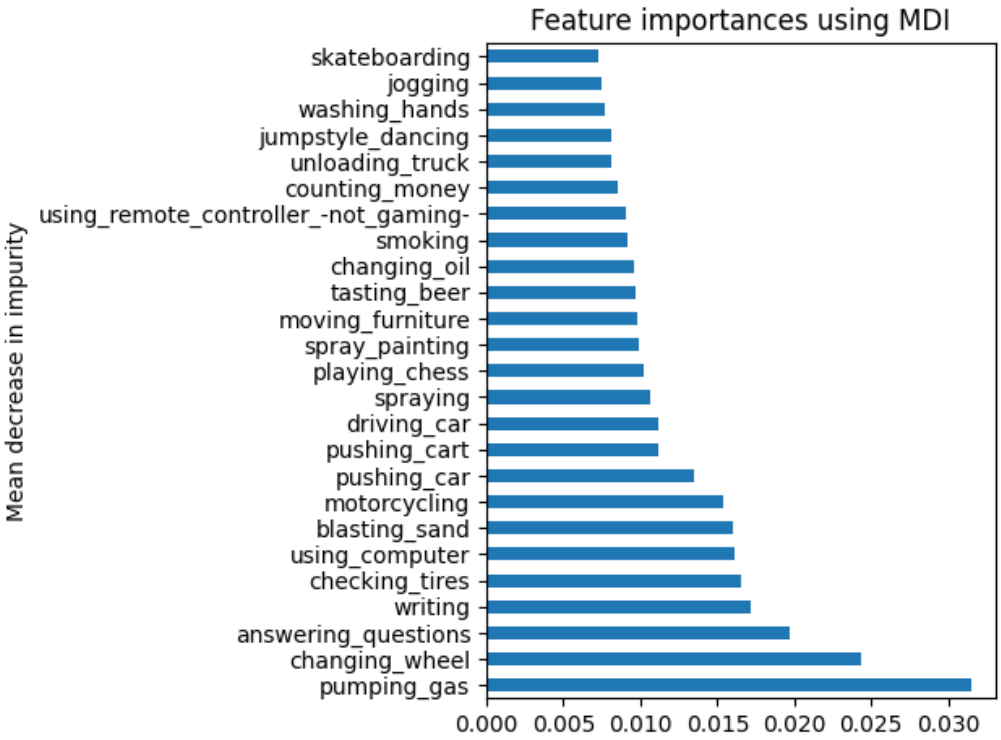

Figure 3.9. NSS-slowfast-8x8-resnet101-kinetics400 dataset feature importance plot

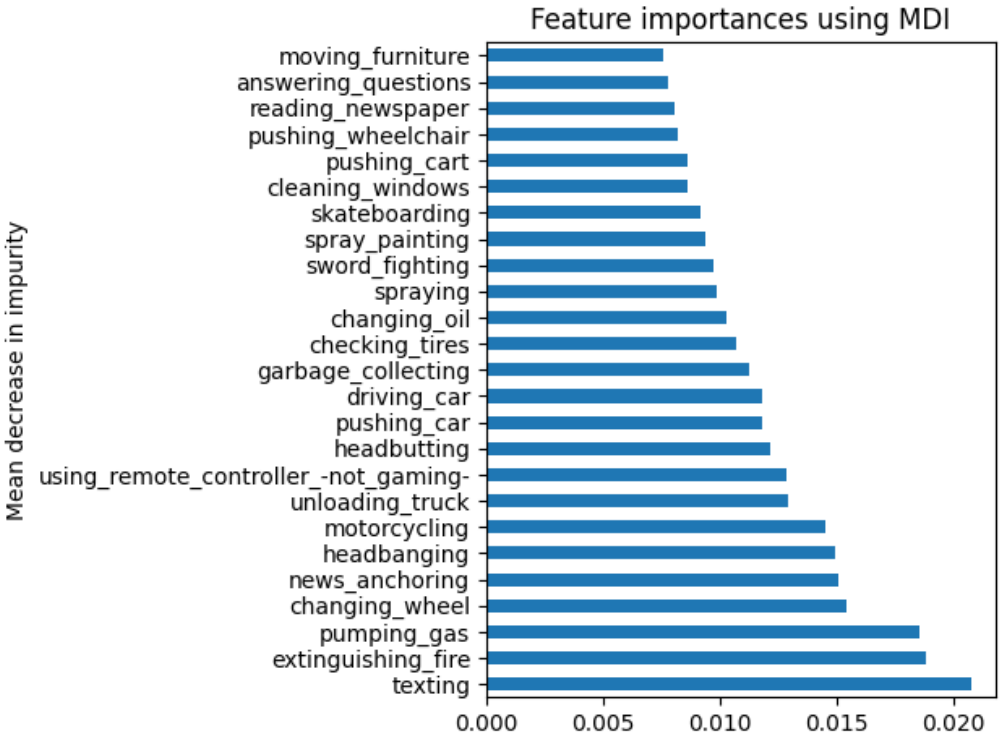

Figure 3.10. NSSR-i3d-nl10-resnet101-v1-kinetics400 dataset feature importance plot

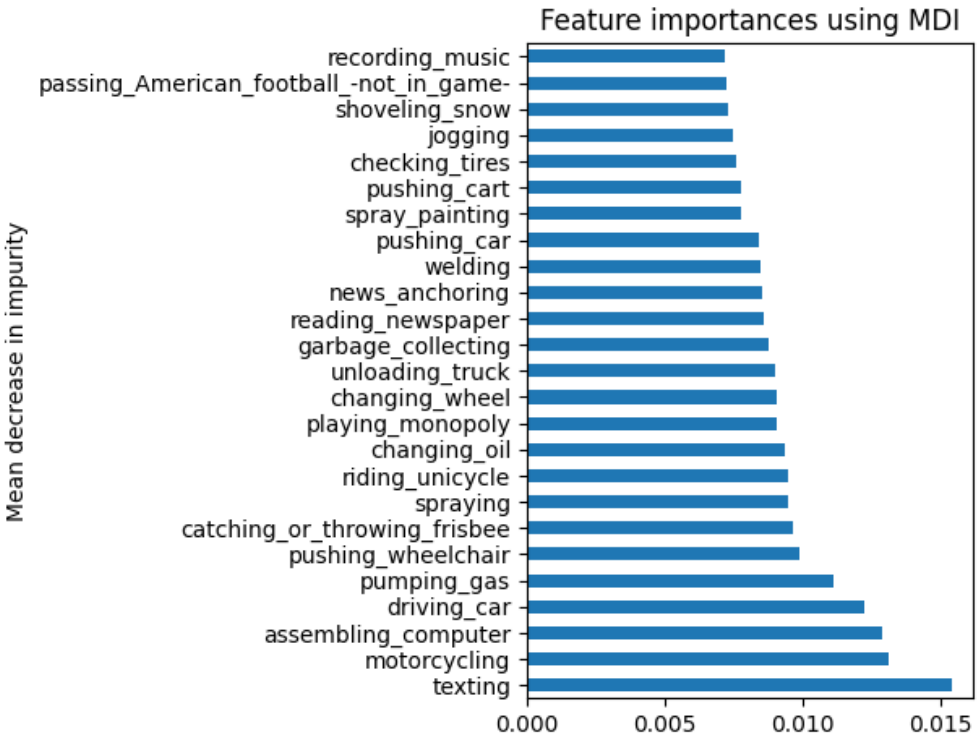

Figure 3.11. NSSR-i3d-resnet50-v1-kinetics400 dataset feature importance plot

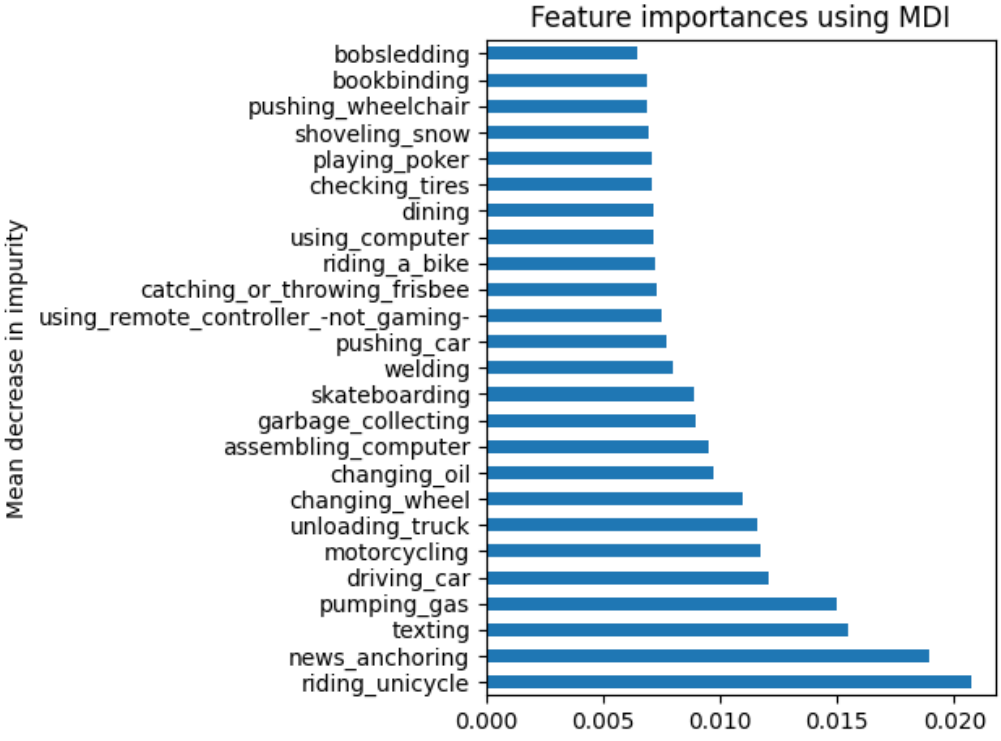

Figure 3.12. The NSSR-slowfast-4x16-resnet50-kinetics400 dataset feature importance plot

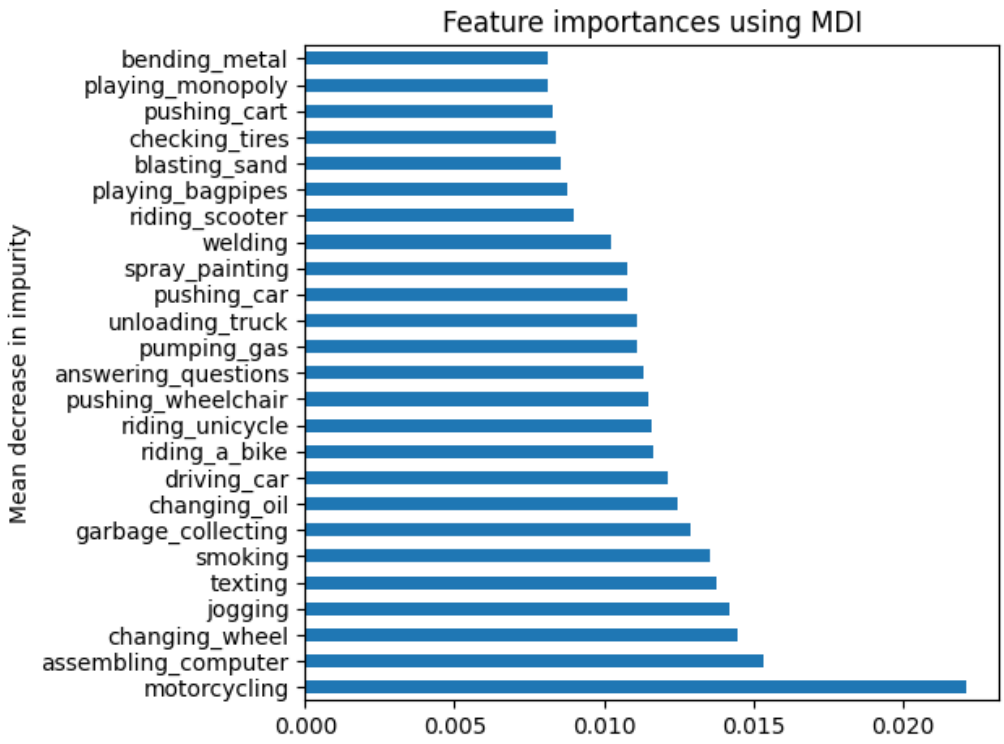

Figure 3.13. NSSR-slowfast-8x8-resnet101-kinetics400 dataset feature importance plot

PCA was applied to all datasets separately, with component numbers 5, 10, 15, and 20 before the training. By the meaning of this, the dimensions of the datasets were reduced. The performance of the models was evaluated both before and after applying PCA. The results revealed that, in some cases, the models performed better following the application of PCA. After the implementation of PCA according to the results, the obtained best models and the datasets on which these models were applied are explained. The best-performing model was the KNeighbors with an AUC of 0.90, accuracy of 0.90, precision of 0.90, recall of 0.91, and F1 score of 0.90 on NS-i3d-nl10- resnet10-v-kinetics400 dataset with used 5 components. The best was the Random Forest with an AUC of 0.85, accuracy of 0.85, precision of 0.85, recall of 0.86, and F1 score of 0.85 on NS-i3d-resnet50-v1-kinetics400 with used 20 components. On the NS-slowfast-4x16-resnet50-kinetics400 with 20 components, the KNeighbors performed the best, with an F1 score of 0.83, an AUC of 0.83, accuracy, precision, and recall of 0.83. With 20 components, Gradient Boosting performed the best, achieving an AUC of 0.84, accuracy, precision, recall, and F1 score of 0.85 on the NS-slowfast-8x8-resnet101-kinetics400.

With an AUC of 0.68, accuracy of 0.68, precision of 0.68, recall of 0.69, and F1 score of 0.68 on NSS-i3d-nl10-resnet101-v1-kinetics400 using 5 components, the Random Forest method performed the best. On the NSS-i3d-resnet50-v1-kinetics400 using 20 components, the KNeighbors approach was received the best, with an F1, accuracy, precision, recall, and AUC of 0.65. With an AUC of 0.73, accuracy, precision, recall, and F1 score of 0.73 on NSS-slowfast-4x16-resnet50-kinetics400 using 5 components, Gradient Boosting performed the best. With an F1 score of 0.72, recall of 0.72, accuracy, precision, and AUC of 0.72 on NSS-slowfast-8x8-resnet101-kinetics400 using 20 components, KNeighbors performed the best. With 20 components and an AUC of 0.66, accuracy of 0.66, precision of 0.66, recall of 0.67, and F1 score of 0.66 on the NSSR-i3d-nl10-resnet101-v1-kinetics400, the Decision Tree model performed the best. The best was the Decision Tree with an AUC of 0.66, accuracy of 0.66, precision of 0.66, recall of 0.66, and F1 score of 0.66 on NSSR-i3d-resnet50-v1-kinetics400 with used 15 components. The best was the Neural Network with an AUC of 0.72, accuracy of 0.72, precision of 0.72, recall of 0.72, and F1 score of 0.72 on NSSR-slowfast-4x16-resnet50-kinetics400 with the same component size. The best was the Neural Network with an AUC of 0.68, accuracy of 0.68, precision of 0.68, recall of 0.68, and F1 score of 0.67 on NSSR-slowfast-8x8- resnet101-kinetics400 with used same component size. It is seen that the best results are obtained when the component number is selected as 20. Furthermore, the results indicate that selecting 10 components led to worse performance, while using 5 and 15 components produced the best outcomes. After applying PCA, the best model is KNeighbors. Additionally, Neural Network and Random Forest, Gradient Boosting and Decision Tree are the other best-performing models.

4.3. Performance Results

In this study, each preprocessing step, PCA, and then feature selection with three different parameters were applied to all datasets separately. All training operations were performed by considering the raw datasets that were not preprocessed and feature selection. As mentioned in the previous sections, the results were evaluated with AUC, recall, precision, accuracy, and F1 score and performance metrics.

Table 2.1 displays the outcomes. The table consists of the dataset name, applied preprocessing process, feature selection process, best machine learning algorithm used, the best parameter set of the algorithm, training performance metrics, and test performance metrics. The best machine learning algorithm and the best parameter set are included in this section, while all the results, including all the machine learning algorithms and all the parameter sets used, are included in the Appendix A section.

Table 4.1. Performance results

Dataset	Preprocess	Feature selection	Machine learning algorithm	Parameter set	Train metrics	Test metrics
NS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 10000)	No	Neural Network	{'hidden_layer_sizes': (50, 50), 'activation': 'relu', 'max_iter': 50}	AUC : 0.86 (+/-0.08) Accuracy : 0.81 (+/-0.13) Recall : 0.81 (+/-0.13) Precision : 0.81 (+/-0.12) F1-Score : 0.80 (+/-0.13)	AUC : 0.90 Accuracy : 0.90 Recall : 0.91 Precision : 0.90 F1-Score : 0.90
NS-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 100000)	No	Gradient Boosting	{'n_estimators': 300, 'max_features': 'log2', 'max_depth': 3, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.87 (+/-0.13) Accuracy : 0.79 (+/-0.14) Recall : 0.79 (+/-0.15) Precision : 0.79 (+/-0.14) F1-Score : 0.79 (+/-0.14)	AUC : 0.90 Accuracy : 0.90 Recall : 0.91 Precision : 0.90 F1-Score : 0.90
NS-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 10000)	No	KNeighbors	{'leaf_size': 30, 'n_neighbors': 5, 'p': 2, 'algorithm': 'auto'}	AUC : 0.86 (+/-0.12) Accuracy : 0.77 (+/-0.13) Recall : 0.77 (+/-0.13) Precision : 0.78 (+/-0.14) F1-Score : 0.77 (+/-0.14)	AUC : 0.86 Accuracy : 0.85 Recall : 0.87 Precision : 0.85 F1-Score : 0.85
NS-slowfast-8x8-resnet101-kinetics400	Binarization (Threshold = 1 / 10000)	No	Random Forest	{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.87 (+/-0.16) Accuracy : 0.78 (+/-0.20) Recall : 0.78 (+/-0.20) Precision : 0.79 (+/-0.21) F1-Score : 0.78 (+/-0.20)	AUC : 0.87 Accuracy : 0.88 Recall : 0.88 Precision : 0.88 F1-Score : 0.87
NSS-i3d-nl10-resnet101-v-kinetics400	Binarization (Threshold = 1 / 1000)	No	Neural Network	{'hidden_layer_sizes': (20,), 'activation': 'relu', 'max_iter': 100}	AUC : 0.78 (+/-0.16) Accuracy : 0.73 (+/-0.15) Recall : 0.73 (+/-0.15) Precision : 0.74 (+/-0.15) F1-Score : 0.73 (+/-0.15)	AUC : 0.77 Accuracy : 0.77 Recall : 0.79 Precision : 0.77 F1-Score : 0.76
NSS-i3d-resnet50-v-kinetics400	Z-score Normalization	No	Gradient Boosting	{'n_estimators': 200, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 2, 'min_samples_leaf': 1, 'subsample': 0.8}	AUC : 0.84 (+/-0.18) Accuracy : 0.75 (+/-0.16) Recall : 0.76 (+/-0.16) Precision : 0.75 (+/-0.16) F1-Score : 0.75 (+/-0.16)	AUC : 0.80 Accuracy : 0.80 Recall : 0.80 Precision : 0.80 F1-Score : 0.80

Table 4.1. (contd)

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Train Metrics	Test Metrics
NSS-slowfast-4x16-resnet50-kinetics400	Max Absolute Scaling	SelectKBest (K = 50)	Random Forest	{'criterion': 'entropy', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 1}	AUC : 0.83 (+/-0.16) Accuracy : 0.74 (+/-0.17) Recall : 0.74 (+/-0.17) Precision : 0.74 (+/-0.16) F1-Score : 0.74 (+/-0.17)	AUC : 0.85 Accuracy : 0.85 Recall : 0.85 Precision : 0.85 F1-Score : 0.85
NSS-slowfast-8x8-resnet101-kinetics400	Binarization (Threshold = 1 / 1000000)	No	Neural Network	{'hidden_layer_sizes': (100,), 'activation': 'identity', 'max_iter': 100}	AUC : 0.76 (+/-0.08) Accuracy : 0.68 (+/-0.12) Recall : 0.69 (+/-0.14) Precision : 0.68 (+/-0.12) F1-Score : 0.67 (+/-0.12)	AUC : 0.80 Accuracy : 0.80 Recall : 0.80 Precision : 0.80 F1-Score : 0.80
NSSR-i3d-nl10-resnet101-v1-kinetics400	Max Absolute Scaling	SelectKBest (K = 50)	Gradient Boosting	{'n_estimators': 200, 'max_features': 'sqrt', 'max_depth': 5, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.77 (+/-0.12) Accuracy : 0.70 (+/-0.08) Recall : 0.70 (+/-0.08) Precision : 0.70 (+/-0.08) F1-Score : 0.70 (+/-0.08)	AUC : 0.76 Accuracy : 0.76 Recall : 0.76 Precision : 0.76 F1-Score : 0.76
NSSR-i3d-resnet50-v1-kinetics400	Binarization (Threshold = 1 / 10000)	No	Gradient Boosting	{'n_estimators': 100, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 2, 'min_samples_leaf': 1, 'subsample': 0.8}	AUC : 0.78 (+/-0.11) Accuracy : 0.69 (+/-0.12) Recall : 0.69 (+/-0.12) Precision : 0.70 (+/-0.12) F1-Score : 0.69 (+/-0.11)	AUC : 0.76 Accuracy : 0.77 Recall : 0.77 Precision : 0.77 F1-Score : 0.77
NSSR-slowfast-4x16-resnet50-kinetics400	Max Absolute Scaling	SelectKBest (K = 20)	Gradient Boosting	{'n_estimators': 100, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.75 (+/-0.08) Accuracy : 0.69 (+/-0.13) Recall : 0.69 (+/-0.13) Precision : 0.69 (+/-0.13) F1-Score : 0.68 (+/-0.13)	AUC : 0.80 Accuracy : 0.80 Recall : 0.81 Precision : 0.80 F1-Score : 0.80
NSSR-slowfast-8x8-resnet101-kinetics400	Binarization (Threshold = 1 / 100000)	No	Gradient Boosting	{'n_estimators': 300, 'max_features': 'log2', 'max_depth': 3, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.76 (+/-0.05) Accuracy : 0.68 (+/-0.05) Recall : 0.68 (+/-0.05) Precision : 0.69 (+/-0.05) F1-Score : 0.68 (+/-0.06)	AUC : 0.73 Accuracy : 0.73 Recall : 0.73 Precision : 0.73 F1-Score : 0.73

4.4. Inspection of False Negatives

In this study, predictions were made using all models to classify videos as either theft-related or normal. The results revealed instances where normal videos were misclassified as theft, or theft-related videos were mistakenly identified as normal. To further investigate these discrepancies, I randomly selected 10 videos and reviewed 3 of them in detail. One of the selected videos depicted a shoplifting incident.As shown in Figure 3.1, I included three frames: one just before the theft, one at the moment of the theft, and one after the event. Because the video quality was low and the frames before and after the theft were similar, it was challenging to identify the precise moment of the theft. This made it challenging for the models to accurately detect the theft, and as a result, the video was misclassified as normal. This, revealed the difficulties of detecting theft in low-quality images where there are consecutive similar frames, which caused to errors in estimation.

Figure 14.1. False negative shoplifting video frames

The second video, which was labeled as "Stealing," was incorrectly predicted as "Normal." In contrast, the third video, labeled as "Normal," was predicted as "Theft." Figure 3.2 shows three frames from the video containing theft, while Figure 3.3 shows three frames from the normal video. Upon comparing these frames, it becomes evident that distinguishing between the two videos is challenging, as the frames appear strikingly similar. This similarity made it difficult for the model to accurately classify the videos, leading to incorrect predictions. The close resemblance between the frames highlights the complexity of accurately detecting theft in such footage.

Figure 15 False negative stealing video frames

Figure 16 False negative normal video frames

5. CONCLUSION

We developed a machine learning-based approach for automatic theft detection utilizing pre-trained human action recognition models in this study. The most suitable models for theft detection in real surveillance datasets were found. Inferences were obtained using action recognition models in theft anomaly video datasets. These inferences were used as features of each video, and this study was guided in this way. By applying machine learning classification models to datasets that contained features obtained from inferences, we aimed to make a difference. In order to assess whether a video is a theft or a normal video, we performed binary classification on different datasets that included theft incidents divided into three separate categories. The best results were achieved with the NS-i3d-nl10-resnet101-v1-kinetics400 and NS-i3d-resnet50-v-kinetics400 datasets, which were generated using the i3d_nl10_resnet101_v1_kinetics400 and i3d-resnet50-v-kinetics400 pre-trained models. These datasets contain normal and theft data. These results were obtained using Neural Network and Gradient Boosting performance models. These results were achieved by applying binarization to the datasets. In the datasets containing stealing, shoplifting, and normal data, the best result was obtained from the NSS-slowfast-4x16-resnet50-kinetics400 dataset. On the other hand, the best results from datasets containing all four categories were achieved from the NSSR-slowfast-4x16-resnet50-kinetics400. These result were obtained by applying max absolute scaling and feature selection techniques to the datasets. When comparing the SlowFast pre-trained video input sampling structures, 8x8 and 4x16, the highest results were obtained with 4x16 according to the dataset categories. While the fast path processes frames at 16 frames per second, the slow path processes frames at four frames per second, allowing us to obtain higher results. The 8x8 result is higher in the dataset containing only normal stealing videos. The machine learning algorithms that delivered the highest performance were Neural Network, Gaussian Naive Bayes, Gradient Boosting, KNeighbors and Random Forest.

In contrast, the Decision Tree and Support Vector Machine models produced the lowest performance and are not recommended for this problem. The results from the preprocessed datasets outperformed those from the raw datasets. The most effective preprocessing techniques were binarization and max absolute scaling. Furthermore, the results achieved by applying feature selection to the datasets generated with the slowfast-4x16-resnet50-kinetics400 pre-trained model are significantly better than those obtained without applying feature selection. Meanwhile, in other datasets, similar performance results were observed with fewer features after applying the feature selection method. The test metrics typically fall within the range obtained by adding or subtracting the standard deviation of the training results. This situation and the test metrics are generally to be higher, as has been expected. However, there are some exceptional cases The short quantity of data in the test set as a result of the little number of theft data is determined to be the cause of this situation.

This approach can be applied to real video surveillance data in theft prevention and increase human safety. Here, by lowering human involvement, there will be the potential to increase accuracy, support scalability, and maintain consistency in video surveillance processing. The ability to accurately detect theft automatically with data received from surveillance video systems increases the security level in indoor and outdoor areas. It enables individuals or institutions to prevent economic or trust losses caused by malicious criminals. Despite its promise, this study still has drawbacks, including the possibility of false positives in complicated situations and its reliance on the standard of surveillance videos. These problems might be resolved by further improving the model. In conclusion, this study shows how machine learning may improve surveillance systems, setting the platform for more secure, more effective ways of preventing crime and theft.

REFERENCES

Adam, A., Rivlin, E., Shimshoni, I., & Reinitz, D. (2008). Robust real-time unusual event detection using multiple fixed-location monitors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *30*(3), 555–560. https://doi.org/10.1109/TPAMI.2007.70825

Amin, S. U., Ullah, M., Sajjad, M., Cheikh, F. A., Hijji, M., Hijji, A., & Muhammad, K. (2022). EADN: An Efficient Deep Learning Model for Anomaly Detection in Videos. *Mathematics 2022, Vol. 10, Page 1555*, *10*(9), 1555. https://doi.org/10.3390/MATH10091555

Anomaly Detection and Localization in Crowded Scenes. (n.d.). Retrieved December 14, 2023, from http://www.svcl.ucsd.edu/projects/anomaly/

Boekhoudt, K., Matei, A., Aghaei, M., & Talavera, E. (n.d.). *HR-Crime: Human-Related Anomaly Detection in Surveillance Videos*. https://doi.org/10.34894/IRRDJE

Chalapathy, R., Menon, A. K., & Chawla, S. (2018). *Anomaly Detection using One-Class Neural Networks*. http://arxiv.org/abs/1802.06360

Chandrakala, S., Deepak, K., & Revathy, G. (2023). Anomaly detection in surveillance videos: a thematic taxonomy of deep models, review and performance analysis. *Artificial Intelligence Review*, *56*(4), 3319–3368. https://doi.org/10.1007/S10462-022-10258-6/FIGURES/10

Danesh Pazho, A., Alinezhad Noghre, G., Rahimi Ardabili, B., Neff, C., & Tabkhi, H. (2022). CHAD: Charlotte Anomaly Dataset. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *13885 LNCS*, 50–66. https://doi.org/10.1007/978-3-031-31435-3_4

Duong, H. T., Le, V. T., & Hoang, V. T. (2023). Deep Learning-Based Anomaly Detection in Video Surveillance: A Survey. *Sensors 2023, Vol. 23, Page 5024*, *23*(11), 5024. https://doi.org/10.3390/S23115024

Karim, M., Khalid, S., Aleryani, A., Khan, J., Ullah, I., & Ali, Z. (2024). Human Action Recognition Systems: A Review of the Trends and State-of-the-Art. *IEEE Access*, *12*, 36372–36390. https://doi.org/10.1109/ACCESS.2024.3373199

Liu, W., Cao, J., Zhu, Y., Liu, B., & Zhu, X. (2023). Real-time anomaly detection on surveillance video with two-stream spatio-temporal generative model. *Multimedia Systems*, *29*(1), 59–71. https://doi.org/10.1007/S00530-022-00979-7/FIGURES/10

Liu, W., Luo, W., Lian, D., & Gao, S. (n.d.). *Future Frame Prediction for Anomaly Detection-A New Baseline*. Retrieved December 14, 2023, from https://github.

Lu, C., Shi, J., & Jia, J. (2013). *Abnormal Event Detection at 150 FPS in MATLAB*. https://doi.org/10.1109/ICCV.2013.338

Mehran, R., Oyama, A., & Shah, M. (n.d.). *Abnormal Crowd Behavior Detection using Social Force Model*.

Muneer, I., Saddique, M., Habib, Z., & Mohamed, H. G. (2023). Shoplifting Detection Using Hybrid Neural Network CNN-BiLSMT and Development of Benchmark Dataset. *Applied Sciences (Switzerland)*, *13*(14). https://doi.org/10.3390/app13148341

Nasaruddin, N., Muchtar, K., Afdhal, A., & Dwiyantoro, A. P. J. (2020). Deep anomaly detection through visual attention in surveillance videos. *Journal of Big Data*, *7*(1), 1–17. https://doi.org/10.1186/S40537-020-00365-Y/TABLES/5

Nayak, R., Pati, U. C., & Das, S. K. (2021). A comprehensive review on deep learning-based methods for video anomaly detection. *Image and Vision Computing*, *106*, 104078. https://doi.org/10.1016/J.IMAVIS.2020.104078

Ramachandra, B., & Jones, M. J. (n.d.). *Street Scene: A new dataset and evaluation protocol for video anomaly detection*.

Şengönül, E., Samet, R., Abu Al-Haija, Q., Alqahtani, A., Alturki, B., & Alsulami, A. A. (2023). An Analysis of Artificial Intelligence Techniques in Surveillance Video Anomaly Detection: A Comprehensive Survey. *Applied Sciences 2023, Vol. 13, Page 4956*, *13*(8), 4956. https://doi.org/10.3390/APP13084956

Shafizadegan, F., Naghsh-Nilchi, A. R., & Shabaninia, E. (2024). Multimodal vision-based human action recognition using deep learning: a review. *Artificial Intelligence Review 2024 57:7*, *57*(7), 1–85. https://doi.org/10.1007/S10462-024-10730-5

Sultani, W., Chen, C., & Shah, M. (n.d.). *Real-world Anomaly Detection in Surveillance Videos*. Retrieved December 14, 2023, from http://crcv.ucf.edu/projects/real-world/

Tran, T. M., Vu, T. N., Vo, N. D., Nguyen, T. V., & Nguyen, K. (2022). Anomaly Analysis in Images and Videos: A Comprehensive Review. *ACM Computing Surveys*, *55*(7). https://doi.org/10.1145/3544014

Ullah, W., Ullah, A., Hussain, T., Khan, Z. A., & Baik, S. W. (2021). An Efficient Anomaly Recognition Framework Using an Attention Residual LSTM in Surveillance Videos. *Sensors 2021, Vol. 21, Page 2811*, *21*(8), 2811. https://doi.org/10.3390/S21082811

Vu, H., Nguyen, T. D., Le, T., Luo, W., & Phung, D. (2019). Robust Anomaly Detection in Videos Using Multilevel Representations. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(01), 5216–5223. https://doi.org/10.1609/AAAI.V33I01.33015216

Waddenkery, N., & Soma, S. (2024). An efficient convolutional neural network for detecting the crime of stealing in videos. *Entertainment Computing*, *51*, 100723. https://doi.org/10.1016/J.ENTCOM.2024.100723

Wan, B., Jiang, W., Fang, Y., Luo, Z., & Ding, G. (2021). Anomaly detection in video sequences: A benchmark and computational model. *IET Image Processing*, *15*(14), 3454–3465. https://doi.org/10.1049/IPR2.12258

Zaigham Zaheer, M., Mahmood, A., Haris Khan, M., Segu, M., Yu, F., & Lee, S.-I. (n.d.). *Generative Cooperative Learning for Unsupervised Video Anomaly Detection*.

APPENDICES

Appendix A. Performance Results of the Best Three Results

In the Appendix A section, for all machine learning algorithms, the performance results of the best three models among all the results obtained by applying all preprocessing and feature selection steps separately are shown in the tables below.

Table 1.1. The best three results after applying selectkbest-20 feature selection and binarization on NS

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 1000)	SelectKBest (K = 20)	Random Forest	{'criterion': 'gini', 'n_estimators': 200, 'max_features': 'sqrt', 'min_samples_leaf': 2}	AUC : 0.88 Accuracy : 0.88 Recall : 0.88 Precision : 0.88 F1-Score : 0.88
NS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 1000)	SelectKBest (K = 20)	Random Forest	{'criterion': 'gini', 'n_estimators': 300, 'max_features': 'log2', 'min_samples_leaf': 2}	AUC : 0.88 Accuracy : 0.88 Recall : 0.88 Precision : 0.88 F1-Score : 0.88
NS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 1000)	SelectKBest (K = 20)	Neural Network	{'hidden_layer_sizes': (100,), 'activation': 'identity', 'max_iter': 100}	AUC : 0.86 Accuracy : 0.85 Recall : 0.87 Precision : 0.85 F1-Score : 0.85
NS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 10000)	SelectKBest (K = 20)	Random Forest	{'criterion': 'gini', 'n_estimators': 200, 'max_features': 'sqrt', 'min_samples_leaf': 2}	AUC : 0.88 Accuracy : 0.88 Recall : 0.88 Precision : 0.88 F1-Score : 0.88
NS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 10000)	SelectKBest (K = 20)	KNeighbors	{'leaf_size': 30, 'n_neighbors': 5, 'p': 2, 'algorithm': 'kd_tree'}	AUC : 0.88 Accuracy : 0.88 Recall : 0.88 Precision : 0.88 F1-Score : 0.88
NS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 10000)	SelectKBest (K = 20)	Neural Network	{'hidden_layer_sizes': (20,), 'activation': 'relu', 'max_iter': 100}	AUC : 0.86 Accuracy : 0.85 Recall : 0.89 Precision : 0.85 F1-Score : 0.85

Table 1.2. The best results after applying selectkbest-20 feature selection and binarization on NS

Dataset	Preprocess	Feature Selection	Best Machine Learning Algorithm	Best Parameter Set	Test Metrics
NS-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 100000)	SelectKBest (K = 20)	Neural Network	{'hidden_layer_sizes': (50, 50), 'activation': 'relu', 'max_iter': 50}	AUC : 0.83 Accuracy : 0.83 Recall : 0.83 Precision : 0.83 F1-Score : 0.83
NS-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 100000)	SelectKBest (K = 20)	Neural Network	{'hidden_layer_sizes': (20,), 'activation': 'relu', 'max_iter': 100}	AUC : 0.82 Accuracy : 0.83 Recall : 0.82 Precision : 0.83 F1-Score : 0.82
NS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 100000)	SelectKBest (K = 20)	Random Forest	{'criterion': 'entropy', 'n_estimators': 200, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.81 Accuracy : 0.80 Recall : 0.83 Precision : 0.80 F1-Score : 0.80
NS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 1000000)	SelectKBest (K = 20)	Random Forest	{'criterion': 'gini', 'n_estimators': 300, 'max_features': 'log2', 'min_samples_leaf': 2}	AUC : 0.83 Accuracy : 0.83 Recall : 0.83 Precision : 0.83 F1-Score : 0.83
NS-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 1000000)	SelectKBest (K = 20)	Gradient Boosting	{'n_estimators': 100, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 2, 'min_samples_leaf': 1, 'subsample': 0.8}	AUC : 0.81 Accuracy : 0.83 Recall : 0.84 Precision : 0.83 F1-Score : 0.82
NS-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 1000000)	SelectKBest (K = 20)	Gradient Boosting	{'n_estimators': 200, 'max_features': 'sqrt', 'max_depth': 5, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.81 Accuracy : 0.83 Recall : 0.84 Precision : 0.83 F1-Score : 0.82

Table 1.3. The best three results after applying selectkbest-20 feature selection and binarization on NSS

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NSS-slowfast-8x8-resnet101-kinetics400	Binarization (Threshold = 1 / 1000)	SelectKBest (K = 20)	Neural Network	{'hidden_layer_sizes': (100,), 'activation': 'relu', 'max_iter': 50}	AUC : 0.75 Accuracy : 0.75 Recall : 0.77 Precision : 0.75 F1-Score : 0.75
NSS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 1000)	SelectKBest (K = 20)	Gradient Boosting	{'n_estimators': 100, 'max_features': 10, 'max_depth': 5, 'min_samples_split': 10, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.74 Accuracy : 0.73 Recall : 0.74 Precision : 0.73 F1-Score : 0.73
NSS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 1000)	SelectKBest (K = 20)	Gradient Boosting	{'n_estimators': 300, 'max_features': 'log2', 'max_depth': 3, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.72 Accuracy : 0.72 Recall : 0.72 Precision : 0.72 F1-Score : 0.72
NSS-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 10000)	SelectKBest (K = 20)	Neural Network	{'hidden_layer_sizes': (20,), 'activation': 'relu', 'max_iter': 100}	AUC : 0.75 Accuracy : 0.75 Recall : 0.75 Precision : 0.75 F1-Score : 0.75
NSS-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 10000)	SelectKBest (K = 20)	Neural Network	{'hidden_layer_sizes': (10,), 'activation': 'relu', 'max_iter': 50}	AUC : 0.75 Accuracy : 0.75 Recall : 0.75 Precision : 0.75 F1-Score : 0.75
NSS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 10000)	SelectKBest (K = 20)	Neural Network	{'hidden_layer_sizes': (20,), 'activation': 'relu', 'max_iter': 100}	AUC : 0.74 Accuracy : 0.73 Recall : 0.80 Precision : 0.73 F1-Score : 0.72

Table 1.4. The best three results after applying selectkbest-20 feature selection and binarization on NSS

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NSS-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 100000)	SelectKBest (K = 20)	Neural Network	{'hidden_layer_sizes': (10,), 'activation': 'relu', 'max_iter': 50}	AUC : 0.72 Accuracy : 0.72 Recall : 0.73 Precision : 0.72 F1-Score : 0.71
NSS-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 100000)	SelectKBest (K = 20)	KNeighbors	{'leaf_size': 30, 'n_neighbors': 11, 'p': 2, 'algorithm': 'auto'}	AUC : 0.70 Accuracy : 0.70 Recall : 0.72 Precision : 0.70 F1-Score : 0.70
NSS-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 100000)	SelectKBest (K = 20)	Random Forest	{'criterion': 'gini', 'n_estimators': 500, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.70 Accuracy : 0.70 Recall : 0.70 Precision : 0.70 F1-Score : 0.70
NSS-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 1000000)	SelectKBest (K = 20)	Support Vector Machine	{'kernel': 'rbf', 'gamma': 1, 'C': 1.0}	AUC : 0.68 Accuracy : 0.68 Recall : 0.68 Precision : 0.68 F1-Score : 0.68
NSS-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 1000000)	SelectKBest (K = 20)	Gaussian Naive Bayes	{'var_smoothing': 0.01}	AUC : 0.68 Accuracy : 0.68 Recall : 0.72 Precision : 0.68 F1-Score : 0.67
NSS-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 1000000)	SelectKBest (K = 20)	Gaussian Naive Bayes	{'var_smoothing': 0.001}	AUC : 0.68 Accuracy : 0.68 Recall : 0.72 Precision : 0.68 F1-Score : 0.67

Table 1.5. The best three results after applying selectkbest-20 feature selection and binarization on NSSR

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NSSR-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 1000)	SelectKBest (K = 20)	Neural Network	{'hidden_layer_sizes': (50, 50), 'activation': 'relu', 'max_iter': 50}	AUC : 0.71 Accuracy : 0.71 Recall : 0.71 Precision : 0.71 F1-Score : 0.71
NSSR-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 1000)	SelectKBest (K = 20)	Gradient Boosting	{'n_estimators': 100, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 2, 'min_samples_leaf': 1, 'subsample': 0.8}	AUC : 0.71 Accuracy : 0.71 Recall : 0.71 Precision : 0.71 F1-Score : 0.71
NSSR-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 1000)	SelectKBest (K = 20)	Neural Network	{'hidden_layer_sizes': (100,), 'activation': 'relu', 'max_iter': 50}	AUC : 0.71 Accuracy : 0.71 Recall : 0.71 Precision : 0.71 F1-Score : 0.71
NSSR-slowfast-8x8-resnet101-kinetics400	Binarization (Threshold = 1 / 10000)	SelectKBest (K = 20)	Neural Network	{'hidden_layer_sizes': (100,), 'activation': 'relu', 'max_iter': 50}	AUC : 0.69 Accuracy : 0.68 Recall : 0.71 Precision : 0.68 F1-Score : 0.68
NSSR-slowfast-8x8-resnet101-kinetics400	Binarization (Threshold = 1 / 10000)	SelectKBest (K = 20)	Gradient Boosting	{'n_estimators': 100, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.69 Accuracy : 0.68 Recall : 0.71 Precision : 0.68 F1-Score : 0.68
NSSR-slowfast-8x8-resnet101-kinetics400	Binarization (Threshold = 1 / 10000)	SelectKBest (K = 20)	Random Forest	{'criterion': 'gini', 'n_estimators': 500, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.68 Accuracy : 0.68 Recall : 0.69 Precision : 0.68 F1-Score : 0.67

Table 1.6. The best three results after applying selectkbest-20 feature selection and binarization on NSSR

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NSSR-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 100000)	SelectKBest (K = 20)	Random Forest	{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 2}	AUC : 0.71 Accuracy : 0.72 Recall : 0.72 Precision : 0.72 F1-Score : 0.71
NSSR-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 100000)	SelectKBest (K = 20)	Neural Network	{'hidden_layer_sizes': (50, 50), 'activation': 'relu', 'max_iter': 50}	AUC : 0.71 Accuracy : 0.72 Recall : 0.72 Precision : 0.72 F1-Score : 0.71
NSSR-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 100000)	SelectKBest (K = 20)	Neural Network	{'hidden_layer_sizes': (100,), 'activation': 'identity', 'max_iter': 100}	AUC : 0.71 Accuracy : 0.72 Recall : 0.72 Precision : 0.72 F1-Score : 0.71
NSSR-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 1000000)	SelectKBest (K = 20)	KNeighbors	{'leaf_size': 30, 'n_neighbors': 5, 'p': 2, 'algorithm': 'auto'}	AUC : 0.64 Accuracy : 0.63 Recall : 0.65 Precision : 0.63 F1-Score : 0.63
NSSR-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 1000000)	SelectKBest (K = 20)	KNeighbors	{'leaf_size': 40, 'n_neighbors': 5, 'p': 2, 'algorithm': 'brute'}	AUC : 0.64 Accuracy : 0.63 Recall : 0.65 Precision : 0.63 F1-Score : 0.63
NSSR-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 1000000)	SelectKBest (K = 20)	Random Forest	{'criterion': 'entropy', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 1	AUC : 0.64 Accuracy : 0.65 Recall : 0.66 Precision : 0.65 F1-Score : 0.64

Table 1.7. The best results after applying selectkbest-20 feature selection and max absolute scaling, z-score normalization on NS

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NS-i3d-nl10-resnet101-v1-kinetics400	Max Absolute Scaling	SelectKBest (K = 20)	Random Forest	{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 1}	AUC : 0.85 Accuracy : 0.85 Recall : 0.86 Precision : 0.85 F1-Score : 0.85
NS-i3d-nl10-resnet101-v1-kinetics400	Max Absolute Scaling	SelectKBest (K = 20)	Random Forest	{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 2}	AUC : 0.83 Accuracy : 0.83 Recall : 0.83 Precision : 0.83 F1-Score : 0.83
NS-i3d-nl10-resnet101-v1-kinetics400	Max Absolute Scaling	SelectKBest (K = 20)	Random Forest	{'criterion': 'gini', 'n_estimators': 200, 'max_features': 'sqrt', 'min_samples_leaf': 2}	AUC : 0.83 Accuracy : 0.83 Recall : 0.83 Precision : 0.83 F1-Score : 0.83
NS-i3d-nl10-resnet101-v1-kinetics400	Z-Score Normalization	SelectKBest (K = 20)	Gradient Boosting	{'n_estimators': 200, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 2, 'min_samples_leaf': 1, 'subsample': 0.8}	AUC : 0.88 Accuracy : 0.88 Recall : 0.88 Precision : 0.88 F1-Score : 0.88
NS-i3d-nl10-resnet101-v1-kinetics400	Z-Score Normalization	SelectKBest (K = 20)	Random Forest	{'criterion': 'entropy', 'n_estimators': 200, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.83 Accuracy : 0.83 Recall : 0.83 Precision : 0.83 F1-Score : 0.83
NS-i3d-nl10-resnet101-v1-kinetics400	Z-Score Normalization	SelectKBest (K = 20)	Random Forest	{'criterion': 'gini', 'n_estimators': 500, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.83 Accuracy : 0.83 Recall : 0.83 Precision : 0.83 F1-Score : 0.83

Table 1.8. The best results after applying selectkbest-20 feature selection and max absolute scaling, z-score normalization on NSS

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NSS-slowfast-4x16-resnet50-kinetics400	Max Absolute Scaling	SelectKBest (K = 20)	Random Forest	{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 1}	AUC : 0.79 Accuracy : 0.78 Recall : 0.80 Precision : 0.78 F1-Score : 0.78
NSS-slowfast-4x16-resnet50-kinetics400	Max Absolute Scaling	SelectKBest (K = 20)	Random Forest	{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 2}	AUC : 0.79 Accuracy : 0.78 Recall : 0.79 Precision : 0.78 F1-Score : 0.78
NSS-slowfast-4x16-resnet50-kinetics400	Max Absolute Scaling	SelectKBest (K = 20)	Random Forest	{'criterion': 'gini', 'n_estimators': 300, 'max_features': 'log2', 'min_samples_leaf': 2}	AUC : 0.77 Accuracy : 0.77 Recall : 0.78 Precision : 0.77 F1-Score : 0.77
NSS-slowfast-4x16-resnet50-kinetics400	Z-Score Normalization	SelectKBest (K = 20)	Gradient Boosting	{'n_estimators': 300, 'max_features': 'log2', 'max_depth': 5, 'min_samples_split': 2, 'min_samples_leaf': 5, 'subsample': 1.0}	AUC : 0.79 Accuracy : 0.78 Recall : 0.79 Precision : 0.78 F1-Score : 0.78
NSS-slowfast-4x16-resnet50-kinetics400	Z-Score Normalization	SelectKBest (K = 20)	Gradient Boosting	{'n_estimators': 200, 'max_features': 'sqrt', 'max_depth': 5, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.77 Accuracy : 0.77 Recall : 0.79 Precision : 0.77 F1-Score : 0.76
NSS-slowfast-4x16-resnet50-kinetics400	Z-Score Normalization	SelectKBest (K = 20)	Random Forest	{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 1}	AUC : 0.77 Accuracy : 0.77 Recall : 0.78 Precision : 0.77 F1-Score : 0.77

Table 1.9. The best results after applying selectkbest-20 feature selection and max absolute scaling, z-score normalization on NSSR

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NSSR-slowfast-4x16-resnet50-kinetics400	Max Absolute Scaling	SelectKBest (K = 20)	Gradient Boosting	{'n_estimators': 100, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.80 Accuracy : 0.80 Recall : 0.81 Precision : 0.80 F1-Score : 0.80
NSSR-slowfast-4x16-resnet50-kinetics400	Max Absolute Scaling	SelectKBest (K = 20)	Gradient Boosting	{'n_estimators': 300, 'max_features': 'log2', 'max_depth': 5, 'min_samples_split': 2, 'min_samples_leaf': 5, 'subsample': 1.0}	AUC : 0.77 Accuracy : 0.77 Recall : 0.78 Precision : 0.77 F1-Score : 0.77
NSSR-slowfast-4x16-resnet50-kinetics400	Max Absolute Scaling	SelectKBest (K = 20)	Random Forest	{'criterion': 'gini', 'n_estimators': 500, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.77 Accuracy : 0.77 Recall : 0.77 Precision : 0.77 F1-Score : 0.77
NSSR-slowfast-4x16-resnet50-kinetics400	Z-Score Normalization	SelectKBest (K = 20)	Gradient Boosting	{'n_estimators': 100, 'max_features': 10, 'max_depth': 5, 'min_samples_split': 10, 'min_samples_leaf': 5, 'subsample': 1.0}	AUC : 0.80 Accuracy : 0.79 Recall : 0.80 Precision : 0.79 F1-Score : 0.79
NSSR-slowfast-4x16-resnet50-kinetics400	Z-Score Normalization	SelectKBest (K = 20)	Gradient Boosting	{'n_estimators': 100, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 2, 'min_samples_leaf': 1, 'subsample': 0.8}	AUC : 0.78 Accuracy : 0.78 Recall : 0.78 Precision : 0.78 F1-Score : 0.77
NSSR-slowfast-4x16-resnet50-kinetics400	Z-Score Normalization	SelectKBest (K = 20)	Gradient Boosting	{'n_estimators': 100, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.78 Accuracy : 0.78 Recall : 0.78 Precision : 0.78 F1-Score : 0.78

Table 1.10. The best three results after applying selectkbest-20 feature selection and no-preprocess on NS and NSS

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NS-i3d-nl10-resnet101-v1-kinetics400	No	SelectKBest (K = 20)	Random Forest	{'criterion': 'entropy', 'n_estimators': 200, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.88 Accuracy : 0.88 Recall : 0.88 Precision : 0.88 F1-Score : 0.88
NS-i3d-nl10-resnet101-v1-kinetics400	No	SelectKBest (K = 20)	Random Forest	{'criterion': 'entropy', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 1}	AUC : 0.86 Accuracy : 0.85 Recall : 0.87 Precision : 0.85 F1-Score : 0.85
NS-i3d-nl10-resnet101-v1-kinetics400	No	SelectKBest (K = 20)	Random Forest	{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.85 Accuracy : 0.85 Recall : 0.86 Precision : 0.85 F1-Score : 0.85
NSS-slowfast-4x16-resnet50-kinetics400	No	SelectKBest (K = 20)	Gradient Boosting	{'n_estimators': 100, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 2, 'min_samples_leaf': 1, 'subsample': 0.8}	AUC : 0.78 Accuracy : 0.78 Recall : 0.79 Precision : 0.78 F1-Score : 0.78
NSS-slowfast-4x16-resnet50-kinetics400	No	SelectKBest (K = 20)	Random Forest	{'criterion': 'entropy', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 1}	AUC : 0.77 Accuracy : 0.77 Recall : 0.77 Precision : 0.77 F1-Score : 0.77
NSS-slowfast-4x16-resnet50-kinetics400	No	SelectKBest (K = 20)	Random Forest	{'criterion': 'gini', 'n_estimators': 200, 'max_features': 'sqrt', 'min_samples_leaf': 2}	AUC : 0.77 Accuracy : 0.77 Recall : 0.77 Precision : 0.77 F1-Score : 0.77

Table 1.11. The best three results after applying selectkbest-20 feature selection and no-preprocess on NSSR

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NSSR-slowfast-4x16-resnet50-kinetics400	No	SelectKBest (K = 20)	Gradient Boosting	{'n_estimators': 100, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 2, 'min_samples_leaf': 1, 'subsample': 0.8}	AUC : 0.80 Accuracy : 0.79 Recall : 0.80 Precision : 0.79 F1-Score : 0.79
NSSR-slowfast-4x16-resnet50-kinetics400	No	SelectKBest (K = 20)	Random Forest	{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 1}	AUC : 0.78 Accuracy : 0.78 Recall : 0.79 Precision : 0.78 F1-Score : 0.78
NSSR-slowfast-4x16-resnet50-kinetics400	No	SelectKBest (K = 20)	Random Forest	{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 2}	AUC : 0.78 Accuracy : 0.78 Recall : 0.78 Precision : 0.78 F1-Score : 0.78

Table 1.12. The best three results after applying selectkbest-30 feature selection and binarization on NS

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 1000)	SelectKBest (K = 30)	KNeighbors	{'leaf_size': 30, 'n_neighbors': 5, 'p': 2, 'algorithm': 'kd_tree'}	AUC : 0.86 Accuracy : 0.85 Recall : 0.89 Precision : 0.85 F1-Score : 0.85
NS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 1000)	SelectKBest (K = 30)	Gradient Boosting	{'n_estimators': 300, 'max_features': 'log2', 'max_depth': 5, 'min_samples_split': 2, 'min_samples_leaf': 5, 'subsample': 1.0}	AUC : 0.85 Accuracy : 0.85 Recall : 0.86 Precision : 0.85 F1-Score : 0.85
NS-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 1000)	SelectKBest (K = 30)	KNeighbors	{'leaf_size': 50, 'n_neighbors': 5, 'p': 2, 'algorithm': 'ball_tree'}	AUC : 0.85 Accuracy : 0.85 Recall : 0.86 Precision : 0.85 F1-Score : 0.85
NS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 10000)	SelectKBest (K = 30)	Random Forest	{'criterion': 'gini', 'n_estimators': 200, 'max_features': 'sqrt', 'min_samples_leaf': 2}	AUC : 0.88 Accuracy : 0.88 Recall : 0.88 Precision : 0.88 F1-Score : 0.88
NS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 10000)	SelectKBest (K = 30)	Random Forest	{'criterion': 'gini', 'n_estimators': 300, 'max_features': 'log2', 'min_samples_leaf': 2}	AUC : 0.88 Accuracy : 0.88 Recall : 0.88 Precision : 0.88 F1-Score : 0.88
NS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 10000)	SelectKBest (K = 30)	Random Forest	{'criterion': 'entropy', 'n_estimators': 200, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.86 Accuracy : 0.85 Recall : 0.87 Precision : 0.85 F1-Score : 0.85

Table 1.13. The best three results after applying selectkbest-30 feature selection and binarization on NS

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NS-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 100000)	SelectKBest (K = 30)	Random Forest	{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.83 Accuracy : 0.83 Recall : 0.83 Precision : 0.83 F1-Score : 0.83
NS-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 100000)	SelectKBest (K = 30)	KNeighbors	{'leaf_size': 50, 'n_neighbors': 15, 'p': 2, 'algorithm': 'auto'}	AUC : 0.83 Accuracy : 0.83 Recall : 0.83 Precision : 0.83 F1-Score : 0.83
NS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 100000)	SelectKBest (K = 30)	Gaussian Naive Bayes	{'var_smoothing': 0.01}	AUC : 0.81 Accuracy : 0.80 Recall : 0.82 Precision : 0.80 F1-Score : 0.80
NS-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 1000000)	SelectKBest (K = 30)	Gradient Boosting	{'n_estimators': 200, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 2, 'min_samples_leaf': 1, 'subsample': 0.8}	AUC : 0.76 Accuracy : 0.78 Recall : 0.79 Precision : 0.78 F1-Score : 0.77
NS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 1000000)	SelectKBest (K = 30)	KNeighbors	{'leaf_size': 50, 'n_neighbors': 15, 'p': 2, 'algorithm': 'auto'}	AUC : 0.75 Accuracy : 0.75 Recall : 0.75 Precision : 0.75 F1-Score : 0.75
NS-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 1000000)	SelectKBest (K = 30)	Gradient Boosting	{'n_estimators': 300, 'max_features': 'log2', 'max_depth': 3, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.74 Accuracy : 0.75 Recall : 0.75 Precision : 0.75 F1-Score : 0.75

Table 1.14. The best three results after applying selectkbest-30 feature selection and binarization on NSS

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NSS-slowfast-8x8-resnet101-kinetics400	Binarization (Threshold = 1 / 1000)	SelectKBest (K = 30)	Random Forest	{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 2}	AUC : 0.74 Accuracy : 0.73 Recall : 0.75 Precision : 0.73 F1-Score : 0.73
NSS-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 1000)	SelectKBest (K = 30)	Neural Network	{'hidden_layer_sizes': (50, 50), 'activation': 'relu', 'max_iter': 50}	AUC : 0.73 Accuracy : 0.73 Recall : 0.73 Precision : 0.73 F1-Score : 0.73
NSS-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 1000)	SelectKBest (K = 30)	Neural Network	{'hidden_layer_sizes': (10,), 'activation': 'relu', 'max_iter': 50}	AUC : 0.72 Accuracy : 0.72 Recall : 0.75 Precision : 0.72 F1-Score : 0.71
NSS-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 10000)	SelectKBest (K = 30)	Neural Network	{'hidden_layer_sizes': (100,), 'activation': 'identity', 'max_iter': 100}	AUC : 0.77 Accuracy : 0.77 Recall : 0.77 Precision : 0.77 F1-Score : 0.77
NSS-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 10000)	SelectKBest (K = 30)	KNeighbors	{'leaf_size': 40, 'n_neighbors': 7, 'p': 2, 'algorithm': 'auto'}	AUC : 0.77 Accuracy : 0.77 Recall : 0.77 Precision : 0.77 F1-Score : 0.77
NSS-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 10000)	SelectKBest (K = 30)	KNeighbors	{'leaf_size': 30, 'n_neighbors': 5, 'p': 2, 'algorithm': 'auto'}	AUC : 0.77 Accuracy : 0.77 Recall : 0.77 Precision : 0.77 F1-Score : 0.77

Table 1.15. The best three results after applying selectkbest-30 feature selection and binarization on NSS

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NSS-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 100000)	SelectKBest (K = 30)	KNeighbors	{'leaf_size': 30, 'n_neighbors': 11, 'p': 2, 'algorithm': 'auto'}	AUC : 0.74 Accuracy : 0.73 Recall : 0.74 Precision : 0.73 F1-Score : 0.73
NSS-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 100000)	SelectKBest (K = 30)	KNeighbors	{'leaf_size': 50, 'n_neighbors': 15, 'p': 2, 'algorithm': 'auto'}	AUC : 0.74 Accuracy : 0.73 Recall : 0.74 Precision : 0.73 F1-Score : 0.73
NSS-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 100000)	SelectKBest (K = 30)	Neural Network	{'hidden_layer_sizes': (50, 50), 'activation': 'relu', 'max_iter': 50}	AUC : 0.72 Accuracy : 0.72 Recall : 0.74 Precision : 0.72 F1-Score : 0.71
NSS-slowfast-8x8-resnet101-kinetics400	Binarization (Threshold = 1 / 1000000)	SelectKBest (K = 30)	Neural Network	{'hidden_layer_sizes': (10,), 'activation': 'relu', 'max_iter': 50}	AUC : 0.68 Accuracy : 0.68 Recall : 0.70 Precision : 0.68 F1-Score : 0.67
NSS-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 1000000)	SelectKBest (K = 30)	Neural Network	{'hidden_layer_sizes': (10,), 'activation': 'relu', 'max_iter': 50}	AUC : 0.66 Accuracy : 0.67 Recall : 0.68 Precision : 0.67 F1-Score : 0.66
NSS-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 1000000)	SelectKBest (K = 30)	Decision Tree	{'criterion': 'gini', 'random_state': 0, 'max_depth': 3, 'min_samples_leaf': 2, 'splitter': 'best'}	AUC : 0.66 Accuracy : 0.67 Recall : 0.73 Precision : 0.67 F1-Score : 0.64

Table 1.16. The best three results after applying selectkbest-30 feature selection and binarization on NSSR

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NSSR-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 1000)	SelectKBest (K = 30)	Neural Network	'hidden_layer_sizes': (50, 50), 'activation': 'relu', 'max_iter': 50}	AUC : 0.73 Accuracy : 0.73 Recall : 0.73 Precision : 0.73 F1-Score : 0.73
NSSR-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 1000)	SelectKBest (K = 30)	Neural Network	{'hidden_layer_sizes': (20,), 'activation': 'relu', 'max_iter': 100}	AUC : 0.72 Accuracy : 0.73 Recall : 0.73 Precision : 0.73 F1-Score : 0.73
NSSR-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 1000)	SelectKBest (K = 30)	Neural Network	{'hidden_layer_sizes': (100,), 'activation': 'identity', 'max_iter': 100}	AUC : 0.72 Accuracy : 0.73 Recall : 0.73 Precision : 0.73 F1-Score : 0.73
NSSR-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 10000)	SelectKBest (K = 30)	Neural Network	{'hidden_layer_sizes': (100,), 'activation': 'identity', 'max_iter': 100}	AUC : 0.72 Accuracy : 0.71 Recall : 0.74 Precision : 0.71 F1-Score : 0.70
NSSR-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 10000)	SelectKBest (K = 30)	Neural Network	{'hidden_layer_sizes': (100,), 'activation': 'relu', 'max_iter': 50}	AUC : 0.71 Accuracy : 0.71 Recall : 0.72 Precision : 0.71 F1-Score : 0.71
NSSR-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 10000)	SelectKBest (K = 30)	Neural Network	{'hidden_layer_sizes': (50, 50), 'activation': 'relu', 'max_iter': 50}	AUC : 0.71 Accuracy : 0.71 Recall : 0.72 Precision : 0.71 F1-Score : 0.71

Table 1.17. The best three results after applying selectkbest-30 feature selection and binarization on NSSR

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NSSR-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 100000)	SelectKBest (K = 30)	Neural Network	{'hidden_layer_sizes': (100,), 'activation': 'identity', 'max_iter': 100}	AUC : 0.72 Accuracy : 0.73 Recall : 0.73 Precision : 0.73 F1-Score : 0.72
NSSR-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 100000)	SelectKBest (K = 30)	Neural Network	{'hidden_layer_sizes': (100,), 'activation': 'relu', 'max_iter': 50}	AUC : 0.71 Accuracy : 0.72 Recall : 0.72 Precision : 0.72 F1-Score : 0.71
NSSR-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 100000)	SelectKBest (K = 30)	Neural Network	{'hidden_layer_sizes': (100,), 'activation': 'identity', 'max_iter': 100}	AUC : 0.71 Accuracy : 0.72 Recall : 0.72 Precision : 0.72 F1-Score : 0.71
NSSR-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 1000000)	SelectKBest (K = 30)	KNeighbors	{'leaf_size': 30, 'n_neighbors': 3, 'p': 2, 'algorithm': 'auto'}	AUC : 0.66 Accuracy : 0.68 Recall : 0.68 Precision : 0.68 F1-Score : 0.66
NSSR-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 1000000)	SelectKBest (K = 30)	Random Forest	{'criterion': 'entropy', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 1}	AUC : 0.66 Accuracy : 0.68 Recall : 0.69 Precision : 0.68 F1-Score : 0.66
NSSR-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 1000000)	SelectKBest (K = 30)	Random Forest	{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 2}	AUC : 0.66 Accuracy : 0.68 Recall : 0.69 Precision : 0.68 F1-Score : 0.66

Table 1.18. The best three results after applying selectkbest-30 feature selection and max absolute scaling, z-score normalization on NS

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NS-i3d-nl10-resnet101-v1-kinetics400	Max Absolute Scaling	SelectKBest (K = 30)	Random Forest	{'criterion': 'gini', 'n_estimators': 500, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.86 Accuracy : 0.85 Recall : 0.87 Precision : 0.85 F1-Score : 0.85
NS-i3d-resnet50-v-kinetics400	Max Absolute Scaling	SelectKBest (K = 30)	Random Forest	{'criterion': 'entropy', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 1}	AUC : 0.86 Accuracy : 0.85 Recall : 0.87 Precision : 0.85 F1-Score : 0.85
NS-i3d-resnet50-v-kinetics400	Max Absolute Scaling	SelectKBest (K = 30)	Random Forest	{'criterion': 'gini', 'n_estimators': 500, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.85 Accuracy : 0.85 Recall : 0.86 Precision : 0.85 F1-Score : 0.85
NS-i3d-nl10-resnet101-v1-kinetics400	Z-Score Normalization	SelectKBest (K = 30)	Gradient Boosting	{'n_estimators': 300, 'max_features': 'log2', 'max_depth': 3, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.85 Accuracy : 0.85 Recall : 0.86 Precision : 0.85 F1-Score : 0.85
NS-i3d-resnet50-v-kinetics400	Z-Score Normalization	SelectKBest (K = 30)	Random Forest	{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.85 Accuracy : 0.85 Recall : 0.86 Precision : 0.85 F1-Score : 0.85
NS-i3d-resnet50-v-kinetics400	Z-Score Normalization	SelectKBest (K = 30)	Random Forest	{'criterion': 'gini', 'n_estimators': 500, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.85 Accuracy : 0.85 Recall : 0.86 Precision : 0.85 F1-Score : 0.85

Table 1.19. The best three results after applying selectkbest-30 feature selection and max absolute scaling, z-score normalization on NSS

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NSS-slowfast-4x16-resnet50-kinetics400	Max Absolute Scaling	SelectKBest (K = 30)	Random Forest	{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 1}	AUC : 0.77 Accuracy : 0.77 Recall : 0.78 Precision : 0.77 F1-Score : 0.77
NSS-slowfast-4x16-resnet50-kinetics400	Max Absolute Scaling	SelectKBest (K = 30)	Random Forest	{'criterion': 'entropy', 'n_estimators': 200, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.77 Accuracy : 0.77 Recall : 0.78 Precision : 0.77 F1-Score : 0.77
NSS-slowfast-4x16-resnet50-kinetics400	Max Absolute Scaling	SelectKBest (K = 30)	Random Forest	{'criterion': 'gini', 'n_estimators': 500, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.77 Accuracy : 0.77 Recall : 0.77 Precision : 0.77 F1-Score : 0.77
NSS-slowfast-4x16-resnet50-kinetics400	Z-Score Normalization	SelectKBest (K = 30)	Gradient Boosting	{'n_estimators': 300, 'max_features': 'log2', 'max_depth': 3, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.79 Accuracy : 0.78 Recall : 0.80 Precision : 0.78 F1-Score : 0.78
NSS-slowfast-4x16-resnet50-kinetics400	Z-Score Normalization	SelectKBest (K = 30)	Random Forest	{'criterion': 'gini', 'n_estimators': 200, 'max_features': 'sqrt', 'min_samples_leaf': 2}	AUC : 0.79 Accuracy : 0.78 Recall : 0.79 Precision : 0.78 F1-Score : 0.78
NSS-slowfast-4x16-resnet50-kinetics400	Z-Score Normalization	SelectKBest (K = 30)	Random Forest	{'criterion': 'gini', 'n_estimators': 300, 'max_features': 'log2', 'min_samples_leaf': 2}	AUC : 0.79 Accuracy : 0.78 Recall : 0.79 Precision : 0.78 F1-Score : 0.78

Table 1.20. The best three results after applying selectkbest-30 feature selection and max absolute scaling, z-score normalization on NSSR

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NSSR-slowfast-4x16-resnet50-kinetics400	Max Absolute Scaling	SelectKBest (K = 30)	Gradient Boosting	{'n_estimators': 100, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 2, 'min_samples_leaf': 1, 'subsample': 0.8}	AUC : 0.75 Accuracy : 0.75 Recall : 0.76 Precision : 0.75 F1-Score : 0.75
NSSR-slowfast-4x16-resnet50-kinetics400	Max Absolute Scaling	SelectKBest (K = 30)	Random Forest	{'criterion': 'gini', 'n_estimators': 200, 'max_features': 'sqrt', 'min_samples_leaf': 2}	AUC : 0.75 Accuracy : 0.75 Recall : 0.75 Precision : 0.75 F1-Score : 0.75
NSSR-slowfast-4x16-resnet50-kinetics400	Max Absolute Scaling	SelectKBest (K = 30)	Random Forest	{'criterion': 'entropy', 'n_estimators': 200, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.75 Accuracy : 0.74 Recall : 0.75 Precision : 0.74 F1-Score : 0.74
NSSR-slowfast-4x16-resnet50-kinetics400	Z-Score Normalization	SelectKBest (K = 30)	Gradient Boosting	{'n_estimators': 300, 'max_features': 'log2', 'max_depth': 5, 'min_samples_split': 2, 'min_samples_leaf': 5, 'subsample': 1.0}	AUC : 0.74 Accuracy : 0.74 Recall : 0.74 Precision : 0.74 F1-Score : 0.74
NSSR-slowfast-4x16-resnet50-kinetics400	Z-Score Normalization	SelectKBest (K = 30)	Gradient Boosting	{'n_estimators': 200, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 2, 'min_samples_leaf': 1, 'subsample': 0.8}	AUC : 0.74 Accuracy : 0.73 Recall : 0.74 Precision : 0.73 F1-Score : 0.73
NSSR-slowfast-4x16-resnet50-kinetics400	Z-Score Normalization	SelectKBest (K = 30)	Gradient Boosting	{'n_estimators': 300, 'max_features': 'log2', 'max_depth': 3, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.74 Accuracy : 0.73 Recall : 0.74 Precision : 0.73 F1-Score : 0.73

Table 1.21. The best three results after applying selectkbest-30 feature selection and no-preprocess on NS and NSS

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NS-i3d-nl10-resnet101-v1-kinetics400	No	SelectKBest (K = 30)	Gradient Boosting	{'n_estimators': 100, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 2, 'min_samples_leaf': 1, 'subsample': 0.8}	AUC : 0.86 Accuracy : 0.85 Recall : 0.89 Precision : 0.85 F1-Score : 0.85
NS-i3d-nl10-resnet101-v1-kinetics400	No	SelectKBest (K = 30)	Random Forest	{'criterion': 'entropy', 'n_estimators': 200, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.86 Accuracy : 0.85 Recall : 0.87 Precision : 0.85 F1-Score : 0.85
NS-i3d-nl10-resnet101-v1-kinetics400	No	SelectKBest (K = 30)	KNeighbors	{'leaf_size': 40, 'n_neighbors': 7, 'p': 2, 'algorithm': 'auto'}	AUC : 0.86 Accuracy : 0.85 Recall : 0.87 Precision : 0.85 F1-Score : 0.85
NSS-slowfast-4x16-resnet50-kinetics400	No	SelectKBest (K = 30)	Gradient Boosting	{'n_estimators': 200, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 2, 'min_samples_leaf': 1, 'subsample': 0.8}	AUC : 0.80 Accuracy : 0.80 Recall : 0.81 Precision : 0.80 F1-Score : 0.80
NSS-slowfast-4x16-resnet50-kinetics400	No	SelectKBest (K = 30)	Random Forest	{'criterion': 'entropy', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 1}	AUC : 0.78 Accuracy : 0.78 Recall : 0.79 Precision : 0.78 F1-Score : 0.78
NSS-slowfast-4x16-resnet50-kinetics400	No	SelectKBest (K = 30)	Gradient Boosting	{'n_estimators': 200, 'max_features': 'sqrt', 'max_depth': 5, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.78 Accuracy : 0.78 Recall : 0.79 Precision : 0.78 F1-Score : 0.78

Table 1.22. The best three results after applying selectkbest-30 feature selection and no-preprocess on NSSR

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NSSR-slowfast-4x16-resnet50-kinetics400	No	SelectKBest (K = 30)	Random Forest	{'criterion': 'entropy', 'n_estimators': 200, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.77 Accuracy : 0.77 Recall : 0.77 Precision : 0.77 F1-Score : 0.77
NSSR-slowfast-4x16-resnet50-kinetics400	No	SelectKBest (K = 30)	Gradient Boosting	{'n_estimators': 200, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 2, 'min_samples_leaf': 1, 'subsample': 0.8}	AUC : 0.77 Accuracy : 0.77 Recall : 0.77 Precision : 0.77 F1-Score : 0.77
NSSR-slowfast-4x16-resnet50-kinetics400	No	SelectKBest (K = 30)	Random Forest	{'criterion': 'gini', 'n_estimators': 200, 'max_features': 'sqrt', 'min_samples_leaf': 2}	AUC : 0.76 Accuracy : 0.76 Recall : 0.77 Precision : 0.76 F1-Score : 0.76

Table 1.23. The best three results after applying selectkbest-50 feature selection and binarization on NS

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 1000)	SelectKBest (K = 50)	Random Forest	{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 1}	AUC : 0.88 Accuracy : 0.88 Recall : 0.89 Precision : 0.88 F1-Score : 0.88
NS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 1000)	SelectKBest (K = 50)	Random Forest	{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.86 Accuracy : 0.85 Recall : 0.87 Precision : 0.85 F1-Score : 0.85
NS-slowfast-8x8-resnet101-kinetics400	Binarization (Threshold = 1 / 1000)	SelectKBest (K = 50)	Neural Network	{'hidden_layer_sizes': (10,), 'activation': 'relu', 'max_iter': 50}	AUC : 0.86 Accuracy : 0.85 Recall : 0.87 Precision : 0.85 F1-Score : 0.85
NS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 10000)	SelectKBest (K = 50)	Gradient Boosting	{'n_estimators': 200, 'max_features': 'sqrt', 'max_depth': 5, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.88 Accuracy : 0.88 Recall : 0.89 Precision : 0.88 F1-Score : 0.88
NS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 10000)	SelectKBest (K = 50)	Neural Network	{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 2}	AUC : 0.88 Accuracy : 0.88 Recall : 0.88 Precision : 0.88 F1-Score : 0.88
NS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 10000)	SelectKBest (K = 50)	Random Forest	{'criterion': 'gini', 'n_estimators': 300, 'max_features': 'log2', 'min_samples_leaf': 2}	AUC : 0.88 Accuracy : 0.88 Recall : 0.88 Precision : 0.88 F1-Score : 0.88

Table 1.24. The best three results after applying selectkbest-50 feature selection and binarization on NS

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NS-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 100000)	SelectKBest (K = 50)	Random Forest	{'criterion': 'entropy', 'n_estimators': 200, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.88 Accuracy : 0.88 Recall : 0.88 Precision : 0.88 F1-Score : 0.88
NS-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 100000)	SelectKBest (K = 50)	Random Forest	{'criterion': 'gini', 'n_estimators': 500, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.83 Accuracy : 0.83 Recall : 0.83 Precision : 0.83 F1-Score : 0.83
NS-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 100000)	SelectKBest (K = 50)	Neural Network	{'hidden_layer_sizes': (100,), 'activation': 'relu', 'max_iter': 50}	AUC : 0.83 Accuracy : 0.83 Recall : 0.83 Precision : 0.83 F1-Score : 0.83
NS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 1000000)	SelectKBest (K = 50)	Support Vector Machine	{'kernel': 'rbf', 'gamma': 1, 'C': 1.0}	AUC : 0.76 Accuracy : 0.75 Recall : 0.78 Precision : 0.75 F1-Score : 0.75
NS-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 1000000)	SelectKBest (K = 50)	Random Forest	{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 1}	AUC : 0.76 Accuracy : 0.78 Recall : 0.79 Precision : 0.78 F1-Score : 0.77
NS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 1000000)	SelectKBest (K = 50)	KNeighbors	{'leaf_size': 50, 'n_neighbors': 5, 'p': 2, 'algorithm': 'ball_tree'}	AUC : 0.75 Accuracy : 0.75 Recall : 0.75 Precision : 0.75 F1-Score : 0.75

Table 1.25. The best three results after applying selectkbest-50 feature selection and binarization on NSS

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NSS-slowfast-8x8-resnet101-kinetics400	Binarization (Threshold = 1 / 1000)	SelectKBest (K = 50)	Random Forest	{'criterion': 'gini', 'n_estimators': 300, 'max_features': 'log2', 'min_samples_leaf': 2}	AUC : 0.77 Accuracy : 0.77 Recall : 0.77 Precision : 0.77 F1-Score : 0.77
NSS-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 1000)	SelectKBest (K = 50)	KNeighbors	{'leaf_size': 50, 'n_neighbors': 15, 'p': 2, 'algorithm': 'auto'}	AUC : 0.73 Accuracy : 0.72 Recall : 0.82 Precision : 0.72 F1-Score : 0.70
NSS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 1000)	SelectKBest (K = 50)	Neural Network	{'hidden_layer_sizes': (20,), 'activation': 'relu', 'max_iter': 100}	AUC : 0.72 Accuracy : 0.72 Recall : 0.77 Precision : 0.72 F1-Score : 0.71
NSS-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 10000)	SelectKBest (K = 50)	Random Forest	{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 2}	AUC : 0.80 Accuracy : 0.80 Recall : 0.80 Precision : 0.80 F1-Score : 0.80
NSS-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 10000)	SelectKBest (K = 50)	Random Forest	{'criterion': 'gini', 'n_estimators': 200, 'max_features': 'sqrt', 'min_samples_leaf': 2}	AUC : 0.78 Accuracy : 0.78 Recall : 0.79 Precision : 0.78 F1-Score : 0.78
NSS-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 10000)	SelectKBest (K = 50)	KNeighbors	{'leaf_size': 30, 'n_neighbors': 5, 'p': 2, 'algorithm': 'auto'}	AUC : 0.78 Accuracy : 0.78 Recall : 0.78 Precision : 0.78 F1-Score : 0.78

Table 1.26. The best three results after applying selectkbest-50 feature selection and binarization on NSS

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NSS-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 100000)	SelectKBest (K = 50)	KNeighbors	{'leaf_size': 40, 'n_neighbors': 7, 'p': 2, 'algorithm': 'auto'}	AUC : 0.73 Accuracy : 0.73 Recall : 0.73 Precision : 0.73 F1-Score : 0.73
NSS-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 100000)	SelectKBest (K = 50)	Random Forest	{'criterion': 'entropy', 'n_estimators': 200, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.72 Accuracy : 0.72 Recall : 0.72 Precision : 0.72 F1-Score : 0.72
NSS-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 100000)	SelectKBest (K = 50)	Gradient Boosting	{'n_estimators': 300, 'max_features': 'log2', 'max_depth': 3, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.71 Accuracy : 0.72 Recall : 0.72 Precision : 0.72 F1-Score : 0.72
NSS-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 1000000)	SelectKBest (K = 50)	Decision Tree	{'criterion': 'gini', 'random_state': 0, 'max_depth': None, 'min_samples_leaf': 1, 'splitter': 'best'}	AUC : 0.70 Accuracy : 0.70 Recall : 0.71 Precision : 0.70 F1-Score : 0.70
NSS-slowfast-8x8-resnet101-kinetics400	Binarization (Threshold = 1 / 1000000)	SelectKBest (K = 50)	Neural Network	{'hidden_layer_sizes': (50, 50), 'activation': 'relu', 'max_iter': 50}	AUC : 0.68 Accuracy : 0.68 Recall : 0.69 Precision : 0.68 F1-Score : 0.68
NSS-slowfast-8x8-resnet101-kinetics400	Binarization (Threshold = 1 / 1000000)	SelectKBest (K = 50)	KNeighbors	{'leaf_size': 50, 'n_neighbors': 15, 'p': 2, 'algorithm': 'auto'}	AUC : 0.66 Accuracy : 0.67 Recall : 0.68 Precision : 0.67 F1-Score : 0.66

Table 1.27. The best three results after applying selectkbest-50 feature selection and binarization on NSSR

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NSSR-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 1000)	SelectKBest (K = 50)	Neural Network	{'hidden_layer_sizes': (50, 50), 'activation': 'relu', 'max_iter': 50}	AUC : 0.73 Accuracy : 0.73 Recall : 0.73 Precision : 0.73 F1-Score : 0.73
NSSR-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 1000)	SelectKBest (K = 50)	Random Forest	{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.73 Accuracy : 0.73 Recall : 0.73 Precision : 0.73 F1-Score : 0.73
NSSR-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 1000)	SelectKBest (K = 50)	Neural Network	{'hidden_layer_sizes': (100,), 'activation': 'relu', 'max_iter': 50}	AUC : 0.73 Accuracy : 0.73 Recall : 0.73 Precision : 0.73 F1-Score : 0.73
NSSR-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 10000)	SelectKBest (K = 50)	Neural Network	{'hidden_layer_sizes': (100,), 'activation': 'relu', 'max_iter': 50}	AUC : 0.76 Accuracy : 0.76 Recall : 0.76 Precision : 0.76 F1-Score : 0.76
NSSR-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 10000)	SelectKBest (K = 50)	Gradient Boosting	{'n_estimators': 200, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 2, 'min_samples_leaf': 1, 'subsample': 0.8}	AUC : 0.75 Accuracy : 0.75 Recall : 0.75 Precision : 0.75 F1-Score : 0.75
NSSR-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 10000)	SelectKBest (K = 50)	Neural Network	{'hidden_layer_sizes': (50, 50), 'activation': 'relu', 'max_iter': 50}	AUC : 0.75 Accuracy : 0.75 Recall : 0.75 Precision : 0.75 F1-Score : 0.75

Table 1.28. The best three results after applying selectkbest-50 feature selection and binarization on NSSR

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NSSR-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 100000)	SelectKBest (K = 50)	Gradient Boosting	{'n_estimators': 200, 'max_features': 'sqrt', 'max_depth': 5, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.78 Accuracy : 0.78 Recall : 0.78 Precision : 0.78 F1-Score : 0.78
NSSR-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 100000)	SelectKBest (K = 50)	Gradient Boosting	{'n_estimators': 300, 'max_features': 'log2', 'max_depth': 5, 'min_samples_split': 2, 'min_samples_leaf': 5, 'subsample': 1.0}	AUC : 0.77 Accuracy : 0.78 Recall : 0.77 Precision : 0.78 F1-Score : 0.77
NSSR-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 100000)	SelectKBest (K = 50)	Gradient Boosting	{'n_estimators': 100, 'max_features': 10, 'max_depth': 5, 'min_samples_split': 10, 'min_samples_leaf': 5, 'subsample': 1.0}	AUC : 0.77 Accuracy : 0.78 Recall : 0.77 Precision : 0.78 F1-Score : 0.77
NSSR-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 1000000)	SelectKBest (K = 50)	Gradient Boosting	{'n_estimators': 200, 'max_features': 'sqrt', 'max_depth': 5, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.66 Accuracy : 0.67 Recall : 0.67 Precision : 0.67 F1-Score : 0.66
NSSR-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 1000000)	SelectKBest (K = 50)	Gradient Boosting	{'n_estimators': 300, 'max_features': 'log2', 'max_depth': 5, 'min_samples_split': 2, 'min_samples_leaf': 5, 'subsample': 1.0}	AUC : 0.65 Accuracy : 0.66 Recall : 0.67 Precision : 0.66 F1-Score : 0.65
NSSR-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 1000000)	SelectKBest (K = 50)	Neural Network	{'hidden_layer_sizes': (50, 50), 'activation': 'relu', 'max_iter': 50}	AUC : 0.65 Accuracy : 0.66 Recall : 0.67 Precision : 0.66 F1-Score : 0.65

Table 1.29. The best three results after applying selectkbest-50 feature selection and max absolute scaling, z-score normalization on NS

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NS-i3d-nl10-resnet101-v1-kinetics400	Max Absolute Scaling	SelectKBest (K = 50)	Random Forest	{'criterion': 'gini', 'n_estimators': 300, 'max_features': 'log2', 'min_samples_leaf': 2}	AUC : 0.90 Accuracy : 0.88 Recall : 0.90 Precision : 0.88 F1-Score : 0.87
NS-i3d-nl10-resnet101-v1-kinetics400	Max Absolute Scaling	SelectKBest (K = 50)	Random Forest	{'criterion': 'gini', 'n_estimators': 500, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.86 Accuracy : 0.85 Recall : 0.87 Precision : 0.85 F1-Score : 0.85
NS-i3d-nl10-resnet101-v1-kinetics400	Max Absolute Scaling	SelectKBest (K = 50)	Neural Network	{'hidden_layer_sizes': (20,), 'activation': 'relu', 'max_iter': 100}	AUC : 0.84 Accuracy : 0.83 Recall : 0.85 Precision : 0.83 F1-Score : 0.82
NS-i3d-nl10-resnet101-v1-kinetics400	Z-Score Normalization	SelectKBest (K = 50)	Random Forest	{'criterion': 'gini', 'n_estimators': 300, 'max_features': 'log2', 'min_samples_leaf': 2}	AUC : 0.86 Accuracy : 0.85 Recall : 0.87 Precision : 0.85 F1-Score : 0.85
NS-i3d-resnet50-v-kinetics400	Z-Score Normalization	SelectKBest (K = 50)	Gradient Boosting	{'n_estimators': 100, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 2, 'min_samples_leaf': 1, 'subsample': 0.8}	AUC : 0.85 Accuracy : 0.85 Recall : 0.86 Precision : 0.85 F1-Score : 0.85
NS-i3d-nl10-resnet101-v1-kinetics400	Z-Score Normalization	SelectKBest (K = 50)	Random Forest	{'criterion': 'entropy', 'n_estimators': 200, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.84 Accuracy : 0.83 Recall : 0.85 Precision : 0.83 F1-Score : 0.82

Table 1.30. The best three results after applying selectkbest-50 feature selection and max absolute scaling, z-score normalization on NSS

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NSS-slowfast-4x16-resnet50-kinetics400	Max Absolute Scaling	SelectKBest (K = 50)	Random Forest	{'criterion': 'entropy', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 1}	AUC : 0.85 Accuracy : 0.85 Recall : 0.85 Precision : 0.85 F1-Score : 0.85
NSS-slowfast-4x16-resnet50-kinetics400	Max Absolute Scaling	SelectKBest (K = 50)	Gradient Boosting	{'n_estimators': 100, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 2, 'min_samples_leaf': 1, 'subsample': 0.8}	AUC : 0.83 Accuracy : 0.83 Recall : 0.84 Precision : 0.83 F1-Score : 0.83
NSS-slowfast-4x16-resnet50-kinetics400	Max Absolute Scaling	SelectKBest (K = 50)	Gradient Boosting	{'n_estimators': 200, 'max_features': 'sqrt', 'max_depth': 5, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.82 Accuracy : 0.82 Recall : 0.82 Precision : 0.82 F1-Score : 0.82
NSS-slowfast-4x16-resnet50-kinetics400	Z-Score Normalization	SelectKBest (K = 50)	Gradient Boosting	{'n_estimators': 100, 'max_features': 10, 'max_depth': 5, 'min_samples_split': 10, 'min_samples_leaf': 5, 'subsample': 1.0}	AUC : 0.83 Accuracy : 0.83 Recall : 0.84 Precision : 0.83 F1-Score : 0.83
NSS-slowfast-4x16-resnet50-kinetics400	Z-Score Normalization	SelectKBest (K = 50)	Gradient Boosting	{'n_estimators': 200, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 2, 'min_samples_leaf': 1, 'subsample': 0.8}	AUC : 0.80 Accuracy : 0.80 Recall : 0.81 Precision : 0.80 F1-Score : 0.80
NSS-slowfast-4x16-resnet50-kinetics400	Z-Score Normalization	SelectKBest (K = 50)	Random Forest	{'criterion': 'entropy', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 1}	AUC : 0.80 Accuracy : 0.80 Recall : 0.81 Precision : 0.80 F1-Score : 0.80

Table 1.31. The best three results after applying selectkbest-50 feature selection and max absolute scaling, z-score normalization on NSSR

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NSSR-i3d-nl10-resnet101-v1-kinetics400	Max Absolute Scaling	SelectKBest (K = 50)	Gradient Boosting	{'n_estimators': 200, 'max_features': 'sqrt', 'max_depth': 5, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.76 Accuracy : 0.76 Recall : 0.76 Precision : 0.76 F1-Score : 0.76
NSSR-slowfast-4x16-resnet50-kinetics400	Max Absolute Scaling	SelectKBest (K = 50)	Random Forest	{'criterion': 'gini', 'n_estimators': 300, 'max_features': 'log2', 'min_samples_leaf': 2}	AUC : 0.74 Accuracy : 0.74 Recall : 0.74 Precision : 0.74 F1-Score : 0.74
NSSR-i3d-nl10-resnet101-v1-kinetics400	Max Absolute Scaling	SelectKBest (K = 50)	Gradient Boosting	{'n_estimators': 100, 'max_features': 10, 'max_depth': 5, 'min_samples_split': 10, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.74 Accuracy : 0.74 Recall : 0.74 Precision : 0.74 F1-Score : 0.74
NSSR-slowfast-4x16-resnet50-kinetics400	Z-Score Normalization	SelectKBest (K = 50)	Random Forest	{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 2}	AUC : 0.76 Accuracy : 0.76 Recall : 0.76 Precision : 0.76 F1-Score : 0.76
NSSR-slowfast-4x16-resnet50-kinetics400	Z-Score Normalization	SelectKBest (K = 50)	Random Forest	{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 1}	AUC : 0.74 Accuracy : 0.74 Recall : 0.74 Precision : 0.74 F1-Score : 0.74
NSSR-slowfast-4x16-resnet50-kinetics400	Z-Score Normalization	SelectKBest (K = 50)	Gradient Boosting	{'n_estimators': 100, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.74 Accuracy : 0.73 Recall : 0.74 Precision : 0.73 F1-Score : 0.73

Table 1.32. The best three results after applying selectkbest-50 feature selection and no-preprocess on NS and NSS

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NS-i3d-nl10-resnet101-v1-kinetics400	No	SelectKBest (K = 50)	Random Forest	{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.86 Accuracy : 0.85 Recall : 0.87 Precision : 0.85 F1-Score : 0.85
NS-i3d-nl10-resnet101-v1-kinetics400	No	SelectKBest (K = 50)	Random Forest	{'criterion': 'entropy', 'n_estimators': 200, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.86 Accuracy : 0.85 Recall : 0.87 Precision : 0.85 F1-Score : 0.85
NS-i3d-resnet50-v-kinetics400	No	SelectKBest (K = 50)	Random Forest	{'criterion': 'gini', 'n_estimators': 500, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.85 Accuracy : 0.85 Recall : 0.85 Precision : 0.85 F1-Score : 0.85
NSS-slowfast-4x16-resnet50-kinetics400	No	SelectKBest (K = 50)	Random Forest	{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 2}	AUC : 0.83 Accuracy : 0.83 Recall : 0.84 Precision : 0.83 F1-Score : 0.83
NSS-slowfast-4x16-resnet50-kinetics400	No	SelectKBest (K = 50)	Random Forest	{'criterion': 'gini', 'n_estimators': 300, 'max_features': 'log2', 'min_samples_leaf': 2}	AUC : 0.83 Accuracy : 0.83 Recall : 0.83 Precision : 0.83 F1-Score : 0.83
NSS-slowfast-4x16-resnet50-kinetics400	No	SelectKBest (K = 50)	Random Forest	{'criterion': 'gini', 'n_estimators': 200, 'max_features': 'sqrt', 'min_samples_leaf': 2}	AUC : 0.82 Accuracy : 0.82 Recall : 0.82 Precision : 0.82 F1-Score : 0.82

Table 1.33. The best three results after applying selectkbest-50 feature selection and no-preprocess on NSSR

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NSSR-slowfast-4x16-resnet50-kinetics400	No	SelectKBest (K = 50)	Gradient Boosting	{'n_estimators': 200, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 2, 'min_samples_leaf': 1, 'subsample': 0.8}	AUC : 0.75 Accuracy : 0.75 Recall : 0.75 Precision : 0.75 F1-Score : 0.75
NSSR-slowfast-4x16-resnet50-kinetics400	No	SelectKBest (K = 50)	Random Forest	{'criterion': 'gini', 'n_estimators': 300, 'max_features': 'log2', 'min_samples_leaf': 2}	AUC : 0.74 Accuracy : 0.73 Recall : 0.74 Precision : 0.73 F1-Score : 0.73
NSSR-i3d-resnet50-v-kinetics400	No	SelectKBest (K = 50)	Gradient Boosting	{'n_estimators': 100, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 2, 'min_samples_leaf': 1, 'subsample': 0.8}	AUC : 0.73 Accuracy : 0.73 Recall : 0.74 Precision : 0.73 F1-Score : 0.73

Table 1.34. The best three results after applying binarizationa and no selection on NS

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 1000)	No	Random Forest	{'criterion': 'entropy', 'n_estimators': 200, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.88 Accuracy : 0.88 Recall : 0.88 Precision : 0.88 F1-Score : 0.88
NS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 1000)	No	Random Forest	{'criterion': 'gini', 'n_estimators': 500, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.88 Accuracy : 0.88 Recall : 0.88 Precision : 0.88 F1-Score : 0.88
NS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 1000)	No	Neural Network	{'hidden_layer_sizes': (20,), 'activation': 'relu', 'max_iter': 100}	AUC : 0.85 Accuracy : 0.85 Recall : 0.85 Precision : 0.85 F1-Score : 0.85
NS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 10000)	No	Neural Network	{'hidden_layer_sizes': (50, 50), 'activation': 'relu', 'max_iter': 50}	AUC : 0.90 Accuracy : 0.90 Recall : 0.91 Precision : 0.90 F1-Score : 0.90
NS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 10000)	No	Neural Network	{'hidden_layer_sizes': (100,), 'activation': 'relu', 'max_iter': 50}	AUC : 0.88 Accuracy : 0.88 Recall : 0.89 Precision : 0.88 F1-Score : 0.88
NS-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 10000)	No	Neural Network	{'hidden_layer_sizes': (20,), 'activation': 'relu', 'max_iter': 100}	AUC : 0.88 Accuracy : 0.88 Recall : 0.88 Precision : 0.88 F1-Score : 0.88

Table 1.35. The best three results after applying binarization and no selection on NS

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NS-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 100000)	No	Gradient Boosting	{'n_estimators': 300, 'max_features': 'log2', 'max_depth': 3, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.90 Accuracy : 0.90 Recall : 0.91 Precision : 0.90 F1-Score : 0.90
NS-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 100000)	No	Random Forest	{'criterion': 'gini', 'n_estimators': 200, 'max_features': 'sqrt', 'min_samples_leaf': 2}	AUC : 0.88 Accuracy : 0.88 Recall : 0.89 Precision : 0.88 F1-Score : 0.88
NS-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 100000)	No	Gradient Boosting	{'n_estimators': 100, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 2, 'min_samples_leaf': 1, 'subsample': 0.8}	AUC : 0.88 Accuracy : 0.88 Recall : 0.89 Precision : 0.88 F1-Score : 0.88
NS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 1000000)	No	Neural Network	{'hidden_layer_sizes': (100,), 'activation': 'relu', 'max_iter': 50}	AUC : 0.83 Accuracy : 0.83 Recall : 0.83 Precision : 0.83 F1-Score : 0.83
NS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 1000000)	No	Neural Network	{'hidden_layer_sizes': (100,), 'activation': 'identity', 'max_iter': 100}	AUC : 0.80 Accuracy : 0.80 Recall : 0.81 Precision : 0.80 F1-Score : 0.80
NS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 1000000)	No	Gradient Boosting	{'n_estimators': 200, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 2, 'min_samples_leaf': 1, 'subsample': 0.8}	AUC : 0.80 Accuracy : 0.80 Recall : 0.81 Precision : 0.80 F1-Score : 0.80

Table 1.36. The best three results after applying binarization and no selection on NSS

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NSS-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 1000)	No	Neural Network	{'hidden_layer_sizes': (100,), 'activation': 'relu', 'max_iter': 50}	AUC : 0.78 Accuracy : 0.78 Recall : 0.79 Precision : 0.78 F1-Score : 0.78
NSS-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 1000)	No	Gradient Boosting	{'n_estimators': 100, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.78 Accuracy : 0.78 Recall : 0.78 Precision : 0.78 F1-Score : 0.78
NSS-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 1000)	No	Neural Network	{'hidden_layer_sizes': (100,), 'activation': 'relu', 'max_iter': 50}	AUC : 0.77 Accuracy : 0.77 Recall : 0.82 Precision : 0.77 F1-Score : 0.76
NSS-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 10000)	No	Random Forest	{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 2}	AUC : 0.82 Accuracy : 0.82 Recall : 0.82 Precision : 0.82 F1-Score : 0.82
NSS-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 10000)	No	Random Forest	{'criterion': 'entropy', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 1}	AUC : 0.80 Accuracy : 0.80 Recall : 0.81 Precision : 0.80 F1-Score : 0.80
NSS-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 10000)	No	Neural Network	{'hidden_layer_sizes': (100,), 'activation': 'relu', 'max_iter': 50}	AUC : 0.79 Accuracy : 0.78 Recall : 0.79 Precision : 0.78 F1-Score : 0.78

Table 1.37. The best three results after applying binarization and no selection on NSS

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NSS-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 100000)	No	Random Forest	{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 2}	AUC : 0.84 Accuracy : 0.83 Recall : 0.84 Precision : 0.83 F1-Score : 0.83
NSS-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 100000)	No	Gradient Boosting	{'n_estimators': 100, 'max_features': 10, 'max_depth': 5, 'min_samples_split': 10, 'min_samples_leaf': 5, 'subsample': 1.0}	AUC : 0.83 Accuracy : 0.83 Recall : 0.84 Precision : 0.83 F1-Score : 0.83
NSS-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 100000)	No	Gradient Boosting	{'n_estimators': 100, 'max_features': 10, 'max_depth': 5, 'min_samples_split': 10, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.82 Accuracy : 0.82 Recall : 0.82 Precision : 0.82 F1-Score : 0.82
NSS-slowfast-8x8-resnet101-kinetics400	Binarization (Threshold = 1 / 1000000)	No	Neural Network	{'hidden_layer_sizes': (100,), 'activation': 'identity', 'max_iter': 100}	AUC : 0.80 Accuracy : 0.80 Recall : 0.80 Precision : 0.80 F1-Score : 0.80
NSS-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 1000000)	No	Gradient Boosting	{'n_estimators': 300, 'max_features': 'log2', 'max_depth': 5, 'min_samples_split': 2, 'min_samples_leaf': 5, 'subsample': 1.0}	AUC : 0.75 Accuracy : 0.75 Recall : 0.75 Precision : 0.75 F1-Score : 0.75
NSS-slowfast-8x8-resnet101-kinetics400	Binarization (Threshold = 1 / 1000000)	No	Gradient Boosting	{'n_estimators': 100, 'max_features': 10, 'max_depth': 5, 'min_samples_split': 10, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.73 Accuracy : 0.73 Recall : 0.73 Precision : 0.73 F1-Score : 0.73

Table 1.38. The best three results after applying binarization and no selection on NSSR

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NSSR-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 1000)	No	Neural Network	{'hidden_layer_sizes': (10,), 'activation': 'relu', 'max_iter': 50}	AUC : 0.74 Accuracy : 0.73 Recall : 0.74 Precision : 0.73 F1-Score : 0.73
NSSR-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 1000)	No	Random Forest	{'criterion': 'gini', 'n_estimators': 200, 'max_features': 'sqrt', 'min_samples_leaf': 2}	AUC : 0.73 Accuracy : 0.73 Recall : 0.73 Precision : 0.73 F1-Score : 0.72
NSSR-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 1000)	No	Random Forest	{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 1}	AUC : 0.73 Accuracy : 0.73 Recall : 0.73 Precision : 0.73 F1-Score : 0.73
NSSR-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 10000)	No	Gradient Boosting	{'n_estimators': 200, 'max_features': 'sqrt', 'max_depth': 5, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.79 Accuracy : 0.78 Recall : 0.79 Precision : 0.78 F1-Score : 0.78
NSSR-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 10000)	No	Neural Network	{'hidden_layer_sizes': (100,), 'activation': 'relu', 'max_iter': 50}	AUC : 0.77 Accuracy : 0.77 Recall : 0.77 Precision : 0.77 F1-Score : 0.77
NSSR-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 10000)	No	Gradient Boosting	{'n_estimators': 200, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 2, 'min_samples_leaf': 1, 'subsample': 0.8}	AUC : 0.77 Accuracy : 0.77 Recall : 0.77 Precision : 0.77 F1-Score : 0.77

Table 1.39 The best three results after applying binarization and no selection on NSSR

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NSSR-i3d-nl10-resnet101-v1-kinetics400	Binarization (Threshold = 1 / 100000)	No	Gradient Boosting	{'n_estimators': 300, 'max_features': 'log2', 'max_depth': 3, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.75 Accuracy : 0.75 Recall : 0.75 Precision : 0.75 F1-Score : 0.75
NSSR-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 100000)	No	Random Forest	{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 5}	AUC : 0.75 Accuracy : 0.75 Recall : 0.75 Precision : 0.75 F1-Score : 0.75
NSSR-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 100000)	No	Gradient Boosting	{'n_estimators': 100, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.73 Accuracy : 0.73 Recall : 0.74 Precision : 0.73 F1-Score : 0.73
NSSR-i3d-resnet50-v-kinetics400	Binarization (Threshold = 1 / 1000000)	No	Random Forest	{'criterion': 'entropy', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 1}	AUC : 0.70 Accuracy : 0.71 Recall : 0.71 Precision : 0.71 F1-Score : 0.71
NSSR-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 1000000)	No	Random Forest	{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 1}	AUC : 0.70 Accuracy : 0.71 Recall : 0.72 Precision : 0.71 F1-Score : 0.70
NSSR-slowfast-4x16-resnet50-kinetics400	Binarization (Threshold = 1 / 1000000)	No	Gradient Boosting	{'n_estimators': 300, 'max_features': 'log2', 'max_depth': 5, 'min_samples_split': 2, 'min_samples_leaf': 5, 'subsample': 1.0}	AUC : 0.70 Accuracy : 0.70 Recall : 0.70 Precision : 0.70 F1-Score : 0.70

Table 1.40. The best three results after applying max absolute scaling, z-score normalization and no selection on NS

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NS-i3d-nl10-resnet101-v1-kinetics400	Max Absolute Scaling	No	Gaussian Naive Bayes	{'var_smoothing': 0.01}	AUC : 0.90 Accuracy : 0.90 Recall : 0.90 Precision : 0.90 F1-Score : 0.90
NS-i3d-nl10-resnet101-v1-kinetics400	Max Absolute Scaling	No	Gaussian Naive Bayes	{'var_smoothing': 0.001}	AUC : 0.90 Accuracy : 0.90 Recall : 0.90 Precision : 0.90 F1-Score : 0.90
NS-i3d-nl10-resnet101-v1-kinetics400	Max Absolute Scaling	No	Neural Network	{'hidden_layer_sizes': (100,), 'activation': 'relu', 'max_iter': 50}	AUC : 0.88 Accuracy : 0.88 Recall : 0.88 Precision : 0.88 F1-Score : 0.88
NS-i3d-nl10-resnet101-v1-kinetics400	Z-Score Normalization	No	Gaussian Naive Bayes	{'var_smoothing': 0.01}	AUC : 0.88 Accuracy : 0.88 Recall : 0.88 Precision : 0.88 F1-Score : 0.88
NS-i3d-nl10-resnet101-v1-kinetics400	Z-Score Normalization	No	Gaussian Naive Bayes	{'var_smoothing': 0.001}	AUC : 0.88 Accuracy : 0.88 Recall : 0.88 Precision : 0.88 F1-Score : 0.88
NS-i3d-nl10-resnet101-v1-kinetics400	Z-Score Normalization	No	Gaussian Naive Bayes	{'var_smoothing': 1e-05}	AUC : 0.88 Accuracy : 0.88 Recall : 0.88 Precision : 0.88 F1-Score : 0.88

Table 1.41. The best three results after applying max absolute scaling, z-score normalization and no selection on NSS

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NSS-slowfast-4x16-resnet50-kinetics400	Max Absolute Scaling	No	Gradient Boosting	{'n_estimators': 200, 'max_features': 'sqrt', 'max_depth': 5, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.82 Accuracy : 0.82 Recall : 0.82 Precision : 0.82 F1-Score : 0.82
NSS-slowfast-4x16-resnet50-kinetics400	Max Absolute Scaling	No	Gradient Boosting	{'n_estimators': 100, 'max_features': 10, 'max_depth': 5, 'min_samples_split': 10, 'min_samples_leaf': 5, 'subsample': 1.0}	AUC : 0.82 Accuracy : 0.82 Recall : 0.82 Precision : 0.82 F1-Score : 0.82
NSS-slowfast-4x16-resnet50-kinetics400	Max Absolute Scaling	No	Gradient Boosting	{'n_estimators': 100, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.80 Accuracy : 0.80 Recall : 0.80 Precision : 0.80 F1-Score : 0.80
NSS-slowfast-4x16-resnet50-kinetics400	Z-Score Normalization	No	Random Forest	{'criterion': 'entropy', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 1}	AUC : 0.85 Accuracy : 0.85 Recall : 0.85 Precision : 0.85 F1-Score : 0.85
NSS-slowfast-4x16-resnet50-kinetics400	Z-Score Normalization	No	Gradient Boosting	{'n_estimators': 100, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 2, 'min_samples_leaf': 1, 'subsample': 0.8}	AUC : 0.82 Accuracy : 0.82 Recall : 0.82 Precision : 0.82 F1-Score : 0.82
NSS-slowfast-4x16-resnet50-kinetics400	Z-Score Normalization	No	Gradient Boosting	{'n_estimators': 200, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 2, 'min_samples_leaf': 1, 'subsample': 0.8}	AUC : 0.82 Accuracy : 0.82 Recall : 0.82 Precision : 0.82 F1-Score : 0.82

Table 1.42. The best three results after applying max absolute scaling, z-score normalization and no selection on NSSR

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NSSR-slowfast-4x16-resnet50-kinetics400	Max Absolute Scaling	No	Random Forest	{'criterion': 'gini', 'n_estimators': 300, 'max_features': 'log2', 'min_samples_leaf': 2}	AUC : 0.76 Accuracy : 0.76 Recall : 0.76 Precision : 0.76 F1-Score : 0.76
NSSR-slowfast-4x16-resnet50-kinetics400	Max Absolute Scaling	No	Gradient Boosting	{'n_estimators': 200, 'max_features': 'sqrt', 'max_depth': 5, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.76 Accuracy : 0.76 Recall : 0.76 Precision : 0.76 F1-Score : 0.76
NSSR-i3d-nl10-resnet101-v1-kinetics400	Max Absolute Scaling	No	Gradient Boosting	{'n_estimators': 200, 'max_features': 'sqrt', 'max_depth': 5, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.76 Accuracy : 0.76 Recall : 0.76 Precision : 0.76 F1-Score : 0.76
NSSR-slowfast-4x16-resnet50-kinetics400	Z-Score Normalization	No	Random Forest	{'criterion': 'gini', 'n_estimators': 300, 'max_features': 'log2', 'min_samples_leaf': 2}	AUC : 0.77 Accuracy : 0.77 Recall : 0.78 Precision : 0.77 F1-Score : 0.77
NSSR-slowfast-4x16-resnet50-kinetics400	Z-Score Normalization	No	Gradient Boosting	{'n_estimators': 300, 'max_features': 'log2', 'max_depth': 5, 'min_samples_split': 2, 'min_samples_leaf': 5, 'subsample': 1.0}	AUC : 0.77 Accuracy : 0.77 Recall : 0.77 Precision : 0.77 F1-Score : 0.77
NSSR-slowfast-4x16-resnet50-kinetics400	Z-Score Normalization	No	Gradient Boosting	{'n_estimators': 100, 'max_features': 10, 'max_depth': 5, 'min_samples_split': 10, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.75 Accuracy : 0.76 Recall : 0.76 Precision : 0.75 F1-Score : 0.75

Table 1.43. The best three results after applying no-preprocess and no selection on NS and NSS

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NS-i3d-nl10-resnet101-v1-kinetics400	No	No	Random Forest	{'criterion': 'gini', 'n_estimators': 200, 'max_features': 'sqrt', 'min_samples_leaf': 2}	AUC : 0.88 Accuracy : 0.88 Recall : 0.88 Precision : 0.88 F1-Score : 0.88
NS-i3d-nl10-resnet101-v1-kinetics400	No	No	Gaussian Naive Bayes	{'var_smoothing': 1e-08}	AUC : 0.88 Accuracy : 0.88 Recall : 0.88 Precision : 0.88 F1-Score : 0.88
NS-i3d-nl10-resnet101-v1-kinetics400	No	No	Gaussian Naive Bayes	{'var_smoothing': 1e-12}	AUC : 0.88 Accuracy : 0.88 Recall : 0.88 Precision : 0.88 F1-Score : 0.88
NSS-slowfast-4x16-resnet50-kinetics400	No	No	Gradient Boosting	{'n_estimators': 100, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 2, 'min_samples_leaf': 1, 'subsample': 0.8}	AUC : 0.82 Accuracy : 0.82 Recall : 0.82 Precision : 0.82 F1-Score : 0.82
NSS-slowfast-4x16-resnet50-kinetics400	No	No	Gradient Boosting	{'n_estimators': 200, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 2, 'min_samples_leaf': 1, 'subsample': 0.8}	AUC : 0.80 Accuracy : 0.80 Recall : 0.81 Precision : 0.80 F1-Score : 0.80
NSS-slowfast-4x16-resnet50-kinetics400	No	No	Gradient Boosting	{'n_estimators': 100, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 1.0}	AUC : 0.80 Accuracy : 0.80 Recall : 0.80 Precision : 0.80 F1-Score : 0.80

Table 1.44. The best three results after applying no-preprocess and no selection on NSSR

Dataset	Preprocess	Feature Selection	Machine Learning Algorithm	Parameter Set	Test Metrics
NSSR-slowfast-4x16-resnet50-kinetics400	No	No	Gradient Boosting	{'n_estimators': 300, 'max_features': 'log2', 'max_depth': 5, 'min_samples_split': 2, 'min_samples_leaf': 5, 'subsample': 1.0}	AUC : 0.77 Accuracy : 0.77 Recall : 0.77 Precision : 0.77 F1-Score : 0.77
NSSR-slowfast-4x16-resnet50-kinetics400	No	No	Random Forest	{'criterion': 'gini', 'n_estimators': 300, 'max_features': 'log2', 'min_samples_leaf': 2}	AUC : 0.76 Accuracy : 0.76 Recall : 0.77 Precision : 0.76 F1-Score : 0.76
NSSR-i3d-resnet50-v-kinetics400	No	No	Gradient Boosting	{'n_estimators': 200, 'max_features': 'sqrt', 'max_depth': 3, 'min_samples_split': 2, 'min_samples_leaf': 1, 'subsample': 0.8}	AUC : 0.76 Accuracy : 0.76 Recall : 0.76 Precision : 0.76 F1-Score : 0.76

Appendix B. 400 Action Classes

In Appendix B section, the 400 action classes used in the data obtained using the pre-train action recognition model are shown in tables.

Table 2.1. 400 action classes

abseiling	braiding_hair	cleaning_toilet	doing_laundry	fixing_hair	hurdling	marching
air_drumming	breading_or_breadcrumbing	cleaning_windows	doing_nails	flipping_pancake	hurling_-sport-	massaging_back
answering_questions	breakdancing	climbing_a_rope	drawing	flying_kite	ice_climbing	massaging_feet
applauding	brush_painting	climbing_ladder	dribbling_basketball	folding_clothes	ice_fishing	massaging_legs
applying_cream	brushing_hair	climbing_tree	drinking	folding_napkins	ice_skating	massaging_
archery	brushing_teeth	contact_juggling	drinking_beer	folding_paper	ironing	person's_head
arm_wrestling	building_cabinet	cooking_chicken	drinking_shots	front_raises	javelin_throw	milking_cow
arranging_flowers	building_shed	cooking_egg	driving_car	frying_vegetables	jetskiing	mopping_floor
assembling_computer	bungee_jumping	cooking_on_campfire	driving_tractor	garbage_collecting	jogging	motorcycling
auctioning	busking	cooking_sausages	drop_kicking	gargling	juggling_balls	moving_furniture
baby_waking_up	canoeing_or_kayaking	counting_money	drumming_fingers	getting_a_haircut	juggling_fire	mowing_lawn
baking_cookies	capoeira	country_line_dancing	dunking_basketball	getting_a_tattoo	juggling_soccer_ball	news_anchoring
balloon_blowing	carrying_baby	cracking_neck	dying_hair	giving_or_receiving_award	jumping_into_pool	opening_bottle
bandaging	cartwheeling	crawling_baby	eating_burger	golf_chipping	jumpstyle_dancing	opening_present
barbequing	carving_pumpkin	crossing_river	eating_cake	golf_driving	kicking_field_goal	paragliding
bartending	catching fish	crying	eating_carrots	golf_putting	kicking_soccer_ball	parasailing
beatboxing	catching_or_throwing_baseball	curling_hair	eating_chips	grinding_meat	kissing	parkour
bee_keeping	catching_or_throwing_frisbee	cutting_nails	eating_doughnuts	grooming_dog	kitesurfing	passing_American
belly_dancing	catching_or_throwing_softball	cutting_pineapple	eating_hotdog	grooming_horse	knitting	_football_in_game-
bench_pressing	celebrating	cutting_watermelon	eating_ice_cream	gymnastics_tumbling	krumping	passing_American_
bending_back	changing_oil	dancing_ballet	eating_spaghetti	hammer_throw	laughing	football_notin_game-
bending_metal	changing_wheel	dancing_charleston	eating_watermelon	headbanging	laying_bricks	peeling_apples
biking_through_snow	checking_tires	dancing_gangnam_style	egg_hunting	headbutting	long_jump	peeling_potatoes
blasting_sand	cheerleading	dancing_macarena	exercising_arm	high_jump	lunge	petting_animal_-
blowing_glass	chopping_wood	deadlifting	exercising_with_an_exercise_ball	high_kick	making_a_cake	not_cat-
blowing_leaves	clapping	decorating_the_christmas_tree	extinguishing_fire	hitting_baseball	making_a_sandwich	petting_cat
blowing_nose	clay_pottery_making	digging	faceplanting	hockey_stop	making_bed	picking_fruit
blowing_out_candles	clean_and_jerk	dining	feeding_birds	holding_snake	making_jewelry	planting_trees
bobsledding	cleaning_floor	disc_golfing	feeding_fish	hopscotch	making_pizza	plastering
bookbinding	cleaning_gutters	diving_cliff	feeding_goats	hoverboarding	making_snowman	playing_accordion
bouncing_on_trampoline	cleaning_pool	dodgeball	filling_eyebrows	hugging	making_sushi	playing_badminton
bowling	cleaning_shoes	doing_aerobics	finger_snapping	hula_hooping	making_tea	playing_bagpipes

Table 2.1. (contd)

playing_basketball	playing_violin	running_on_treadmil	skipping_rope	swinging_legs	using_remote_controller
playing_bass_guitar	playing_volleyball	sailing	slacklining	swinging_on_something	_-not_gaming-
playing_cards	playing_xylophone	salsa_dancing	slapping	sword_fighting	using_segway
playing_cello	pole_vault	sanding_floor	sled_dog_racing	tai_chi	vault
playing_chess	presenting_weather_forecast	scrambling_eggs	smoking	taking_a_shower	waiting_in_line
playing_clarinet	pull_ups	scuba_diving	smoking_hookah	tango_dancing	walking_the_dog
playing_controller	pumping_fist	setting_table	snatch_weight_lifting	tap_dancing	washing_dishes
playing_cricket	pumping_gas	shaking_hands	sneezing	tapping_guitar	washing_feet
playing_cymbals	punching_bag	shaking_head	sniffing	tapping_pen	washing_hair
playing_didgeridoo	punching_person_-boxing-	sharpening_knives	snorkeling	tasting_beer	washing_hands
playing_drums	push_up	sharpening_pencil	snowboarding	tasting_food	water_skiing
playing_flute	pushing_car	shaving_head	snowkiting	testifying	water_sliding
playing_guitar	pushing_cart	shaving_legs	snowmobiling	texting	watering_plants
playing_harmonica	pushing_wheelchair	shearing_sheep	somersaulting	throwing_axe	waxing_back
playing_harp	reading_book	shining_shoes	spinning_poi	throwing_ball	waxing_chest
playing_ice_hockey	reading_newspaper	shooting_basketball	spray_painting	throwing_discus	waxing_eyebrows
playing_keyboard	recording_music	shooting_goal_-soccer-	spraying	tickling	waxing_legs
playing_kickball	riding_a_bike	shot_put	springboard_diving	tobogganing	weaving_basket
playing_monopoly	riding_camel	shoveling_snow	squat	tossing_coin	welding
playing_organ	riding_elephant	shredding_paper	sticking_tongue_out	tossing_salad	whistling
playing_paintball	riding_mechanical_bull	shuffling_cards	stomping_grapes	training_dog	windsurfing
playing_piano	riding_mountain_bike	side_kick	stretching_arm	trapezing	wrapping_present
playing_poker	riding_mule	sign_language_interpreting	stretching_leg	trimming_or_shaving_beard	wrestling
playing_recorder	riding_or_walking_with_horse	singing	strumming_guitar	trimming_trees	writing
playing_saxophone	riding_scooter	situp	surfing_crowd	triple_jump	yawning
playing_squash	riding_unicycle	skateboarding	surfing_water	tying_bow_tie	yoga
_or_racquetball	ripping_paper	ski_jumping	sweeping_floor	tying_knot_-not_on_a_tie-	zumba
playing_tennis	robot_dancing	skiing_not_	swimming_backstroke	tying_tie	
playing_trombone	rock_climbing	slalom_or_crosscountry-	swimming_breast_stroke	unboxing	
playing_trumpet	rock_scissors_paper	skiing_crosscountry	swimming_butterfly_stroke	unloading_truck	
playing_ukulele	roller_skating	skiing_slalom	swing_dancing	using_computer	

Appendix C. Machine Learning Algorithms Parameter Sets

In Appendix C section, All parameter sets used for all machine learning algorithms are shown in tables.

Table 3.1. Neural network parameter sets

{'hidden_layer_sizes': (10,), 'activation': 'relu', 'max_iter' : 50}
{'hidden_layer_sizes': (100,), 'activation': 'relu', 'max_iter' : 50}
{'hidden_layer_sizes': (50,50), 'activation': 'relu', 'max_iter' : 50}
{'hidden_layer_sizes': (20,), 'activation': 'relu', 'max_iter' : 100}
{'hidden_layer_sizes': (100,), 'activation': 'identity', 'max_iter' : 100}

Table 3.2. Support vector machine parameter sets

{'kernel': 'linear', 'C': 1.0}
{'kernel': 'rbf', 'gamma': 1, 'C': 1.0}
{'kernel': 'poly', 'degree': 3, 'gamma': 0.01, 'C': 1.0}
{'kernel': 'sigmoid', 'gamma': 1, 'C': 1.0}

Table 3.3. Decision tree parameter sets

{'criterion': 'gini', 'random_state' : 0, 'max_depth' : None, 'min_samples_leaf' : 1, 'splitter': 'best'}
{'criterion': 'entropy', 'random_state' : 0, 'max_depth' : 5, 'min_samples_leaf' : 1, 'splitter': 'best'}
{'criterion': 'entropy', 'random_state' : 42, 'max_depth' : 3, 'min_samples_leaf' : 1, 'splitter': 'best'}
{'criterion': 'gini', 'random_state' : 0, 'max_depth' : 3, 'min_samples_leaf' : 2, 'splitter': 'best'}
{'criterion': 'gini', 'random_state' : 42, 'max_depth' : 5, 'min_samples_leaf' : 2, 'splitter': 'best'}
{'criterion': 'gini', 'random_state' : 0, 'max_depth' : 3, 'min_samples_leaf' : 2, 'splitter': 'random'}
{'criterion': 'entropy', 'random_state' : 0, 'max_depth' : 3, 'min_samples_leaf' : 2, 'splitter': 'random'}

Table 3.4. Gaussian naive bayes parameter sets

{'var_smoothing': 1e-2}
{'var_smoothing': 1e-3}
{'var_smoothing': 1e-5}
{'var_smoothing': 1e-8}
{'var_smoothing': 1e-12}
{'var_smoothing': 1e-15}

Table 3.5. Random forest parameter sets

{'criterion': 'entropy', 'n_estimators' : 100, 'max_features' : 'sqrt', 'min_samples_leaf' : 1}
{'criterion': 'gini', 'n_estimators' : 100, 'max_features' : 'sqrt', 'min_samples_leaf' : 1}
{'criterion': 'gini', 'n_estimators' : 100, 'max_features' : 'sqrt', 'min_samples_leaf' : 2}
{'criterion': 'gini', 'n_estimators' : 200, 'max_features' : 'sqrt', 'min_samples_leaf' : 2}
{'criterion': 'gini', 'n_estimators' : 300, 'max_features' : 'log2', 'min_samples_leaf' : 2}
{'criterion': 'gini', 'n_estimators': 100, 'max_features': 'sqrt', 'min_samples_leaf': 5}
{'criterion': 'entropy', 'n_estimators': 200, 'max_features': 'sqrt', 'min_samples_leaf': 5}
{'criterion': 'gini', 'n_estimators': 500, 'max_features': 'sqrt', 'min_samples_leaf': 5}

Table 3.6. KNeighbors parameter sets

{'leaf_size': 30, 'n_neighbors': 3, 'p': 2, 'algorithm': 'auto'}
{'leaf_size': 30, 'n_neighbors': 5, 'p': 2, 'algorithm': 'auto'}
{'leaf_size': 40, 'n_neighbors': 7, 'p': 2, 'algorithm': 'auto'}
{'leaf_size': 30, 'n_neighbors': 11, 'p': 2, 'algorithm': 'auto'}
{'leaf_size': 50, 'n_neighbors': 15, 'p': 2, 'algorithm': 'auto'}
{'leaf_size': 50, 'n_neighbors': 5, 'p': 2, 'algorithm': 'ball_tree'}
{'leaf_size': 30, 'n_neighbors': 5, 'p': 2, 'algorithm': 'kd_tree'}
{'leaf_size': 40, 'n_neighbors': 5, 'p': 2, 'algorithm': 'brute'}

Table 3.7. Gradient boosting parameter sets

{'n_estimators': 100, 'max_features' : 'sqrt', 'max_depth' : 3, 'min_samples_split' : 2, 'min_samples_leaf' : 1,
{'n_estimators': 200, 'max_features' : 'sqrt', 'max_depth' : 3, 'min_samples_split' : 2, 'min_samples_leaf' : 1,
{'n_estimators': 100, 'max_features' : 'sqrt', 'max_depth' : 3, 'min_samples_split' : 5, 'min_samples_leaf' : 1,
{'n_estimators': 300, 'max_features' : 'log2', 'max_depth' : 3, 'min_samples_split' : 5, 'min_samples_leaf' : 1,
{'n_estimators': 200, 'max_features' : 'sqrt', 'max_depth' : 5, 'min_samples_split' : 5, 'min_samples_leaf' : 1,
{'n_estimators': 100, 'max_features' : 10, 'max_depth' : 5, 'min_samples_split' : 10, 'min_samples_leaf' : 1,
{'n_estimators': 300, 'max_features' : 'log2', 'max_depth' : 5, 'min_samples_split' : 2, 'min_samples_leaf' : 5,
{'n_estimators': 100, 'max_features' : 10, 'max_depth' : 5, 'min_samples_split' : 10, 'min_samples_leaf' : 5,

CURRICULUM VITAE

Personel Information

Name & Surname : İ*** K***** U******

Nationality : T*****h

Phone : +90 5********6

E-posta : i***************@** *** *r

Education

Degree	Institution	Year of Graduation
High School	Muğla Anatolian High School	2012
BSc	Muğla Sıtkı Koçman University	2017

Work Experience

Software Engineer (2018 - 2022) – Tübitak (İLTAREN)

Research Assistant (2022 - Present) – Muğla Sıtkı Koçman University Department of Computer Engineering

Oral Presentations in International Symposiums

Karaca Uluoğlu İrem, Süzek Barış Ethem, 2024. A Machine Learning Based Approach for Automatic Theft Detection. IGSCONG'24 - 4th International Graduate Studies Congress (IGSCONG)