INDEX

LIST OF FIGURES

CAMPUS ABNORMAL BEHAVIOUR RECOGNITION WITH TEMPORAL SEGMENT TRANSFORMERS

ABSTRACT

The intelligent campus surveillance system is beneficial to improve safety in school. Abnormal behavior recognition, a field of action recognition in computer vision, plays an essential role in intelligent surveillance systems. Computer vision has been actively applied to action recognition systems based on Convolutional Neural Networks (CNNs). However, capturing sufficient motion sequence features from videos remains a significant challenge in action recognition. This work explores the challenges of video-based abnormal behavior recognition on campus. In addition, a novel framework is established on long-range temporal video structure modeling and a global sparse uniform sampling strategy that divides a video into three segments of identical durations and uniformly samples each snippet. The proposed method incorporates a consensus of three temporal segment transformers (TST) that globally connects patches and computes self attention with joint spatiotemporal factorization. The proposed model is developed on the newly created campus abnormal behavior recognition (CABR50) dataset, which contains 50 human abnormal action classes with an average of over 700 clips per class. Experiments show that it is feasible to implement abnormal behavior recognition on campus and that the proposed method is competitive with other peer video recognition in terms of Top-1 and Top-5 recognition accuracy. The results suggest that TST-L+ can improve campus abnormal behavior recognition, corresponding to Top-1 and Top-5 accuracy results of 83.57% and 97.16%, respectively.

CHAPTER-1
INTRODUCTION

Campus abnormal behavior recognition refers to using surveillance devices and artificial intelligence to identify unusual or potentially threatening behavior on campus. Video understanding is a core technology [1], [2], [3], [4] in many scenarios of surveillance systems. Over the years, unexpected actions, such as fighting, accidents, falling, and suicides, have occurred frequently in schools, causing general concern. Recognizing abnormal behavior can achieve real-time and efficient warning, positively affecting school safety management. Researchers focus on directly exploring abnormal behaviors instead of relying heavily on pre-processing to classify video behaviors [5], [6]. Researchers have focused on applications in specific scenarios on campus, such as classrooms [45] and laboratories [46], [47]. However, there is little research on campus abnormal behavior recognition. Essentially, abnormal behavior is a wide range of applications of video understanding. Motivated by video understanding, this study aims to provide an effective solution for recognizing video-based abnormal behavior on campus.

Deep learning has become increasingly popular for image recognition [7], [8], [9], [10], [11]. Depending on deep learning, video understanding has emerged in an endless stream to push action recognition to a climax. However, there are challenges in representing temporal video features, mainly focusing on three popular categories: (1) two-dimensional (2D) networks, (2) three-dimensional (3D) networks, and (3) transformers. In the first category, 2D network success represented by two-stream networks [12] pushed video understanding into the deep learning era. The following versions [13], [14], [15] related to two-stream networks emerged within a year, which have a similar network structure. One spatial network branch learns spatial information; the other is an optical flow network representing temporal information. Two-stream networks exhibit superior performance in learning spatial and temporal features separately. Because of the complex optical flow calculation and high storage requirements for pre-processing [12], previous studies are unsuitable for large-scale training and real-time deployment.

Meanwhile, for the second category: 3D networks, video understanding is a 3D tensor composed of two spatial dimensions and one temporal dimension to extract spatial and temporal features. However, optimizing the 3D model requires more work and relies heavily on diverse data than 2D networks [12], [16]. This situation changes when an inflated 3D model (I3D) [17] develops. The I3D operation can inflate Image Net's pre-trained 2D model to the corresponding 3D model, accelerating optimization. Research related to 3D convolutional neural networks (CNNs) followed the emergence of I3D [18], [19], [20], [21], [22]. The 3D network with natural temporal properties [16] and inflated operation [17], [18], [19] have a competitive effect on video recognition. Consequently, it has long-dominated action recognition.

In the third category, transformers [23], [24], [25], [26] challenge the dominance of CNNs in deep learning and break the barriers of computer vision and natural language processing models. Because of its excellent capabilities in capturing distant information, especially for medium-range and long-range video modeling . Therefore, researchers are interested in applying transformers from the image field to video understanding [27], [28], [29]. However, if an element of self-attention consists of each image pixel, self-attention cannot be directly calculated in a transformer model with relatively high complexity. Hence, a fundamental problem to be solved is reducing the sequence length and designing a self-attention method. Times former [27] applied this sequence of frame-level patches with a size of 16 × 16 pixels instead of every pixel in an image and explored five structures of self-attention. This demonstrates that the divided space time attention method is faster to train than the 3D CNNs. Applying a transformer in video understanding on the kinetics 400 dataset achieved the best performance compared to CNNs [22] for the first time. However, the transformer has a common issue [23]: it is challenging to learn inductive bias owing to the lack of a large amount of pre-training data, like prior knowledge of the locality and translation equivariance in CNNs. Therefore, researchers have attempted to solve the inductive bias problem of pure transformers [28], [31], [32].

We aim to solve the problem of abnormal campus behavior recognition. Comparing different approaches with the results of the original paper above is challenging because a generic suite is needed to test these types of solutions using the new campus anomalous behavior dataset.

Therefore, the proposed models are adequately compared with three relevant proposals: TSN [13], Slowfast [22], and Swin-B [31]. In addition, this work attempts to innovate abnormal behavior identification on campus. First, the backbone network consists of video shifted windows transformer [31], which effectively overcomes the inductive bias problem of the transformer: locality and translation equivariance. It also dramatically resolves the transformer sequence length issue and improves the global modeling ability of models by using a multi-scale shift window to calculate self-attention.

However, the problem with current models is their inability to model an entire video [13], [14]. Since they operate only on a single frame or a stack of frames within a short segment, they have limited access to the temporal context. The complex actions are illustrated in Fig. 1. Abnormal campus actions contain multiple segments with similar redundancies between the consecutive frames of one segment. Failure to use a long-range temporal structure for network training loses the ability to model the entire behavior. Motivated by this early work on video segmentation fusion TSN [13], we designed a campus abnormal behavior recognition framework called temporal segment transformer (TST) to exploit temporal action features and achieve video-level global modeling. Therefore, instead of working on a single frame or stacked frames, TST processes a sequence of snippets globally and is sparsely sampled from the entire video. Each snippet produces its initial class prediction, and a consensus function between the snippets is exported as the final prediction to enable global video dynamic modeling. It can remove redundant information and increase the difference between the behavior classes. Moreover, extensive experiments and discussions support a comparative study of these three methods. Overall, the main contributions of this study are summarized as follows:

- We propose a consensus of three temporal segment transformers (TST) based on the video Swin transformer for the new campus abnormal behavior recognition (CABR50) dataset. It enhances the ability to capture motion sequences and model long-range abnormal behavior on campus.
- We perform extensive comparative experiments with state-of-the-art methods for recognizing abnormal campus behaviors. The results show that it is feasible and can improve the accuracy of abnormal campus behavior recognition. In addition, we demonstrate the performance comparison

of the TST and previous methods onthe UCF-101 dataset. It indicates that our proposed TST model has acceptable generalization performance.

• Our research provides essential technical support for the identification and early warning of abnormal behavior on campus, which plays an essential role in intelligent campus surveillance systems.

In the rest of the paper, Section II summarizes the research methods related to our work. Section III outlines how the TST is employed for campus abnormal behavior recognition and introduces related components. Section IV presents the experimental results and discussion. Section V concludes the work and presents perspectives.

CHAPTER-2
LITERATURE SURVEY

TITLE: Campus Abnormal Behavior Recognition With Temporal Segment Transformers

AUTHOR: Hai Chuan Liu; Joon Huang Chuah; Anis Salwa Mohd Khairuddin; Xian Min Zhao; Xiao Dan

ABSTRACT: The intelligent campus surveillance system is beneficial to improve safety in school. Abnormal behavior recognition, a field of action recognition in computer vision, plays an essential role in intelligent surveillance systems. Computer vision has been actively applied to action recognition systems based on Convolutional Neural Networks (CNNs). However, capturing sufficient motion sequence features from videos remains a significant challenge in action recognition. This work explores the challenges of video-based abnormal behavior recognition on campus. In addition, a novel framework is established on long-range temporal video structure modeling and a global sparse uniform sampling strategy that divides a video into three segments of identical durations and uniformly samples each snippet. The proposed method incorporates a consensus of three temporal segment transformers (TST) that globally connects patches and computes self-attention with joint spatiotemporal factorization. The proposed model is developed on the newly created campus abnormal behavior recognition (CABR50) dataset, which contains 50 human abnormal action classes with an average of over 700 clips per class. Experiments show that it is feasible to implement abnormal behavior recognition on campus and that the proposed method is competitive with other peer video recognition in terms of Top-1 and Top-5 recognition accuracy. The results suggest that TST-L+ can improve campus abnormal behavior recognition, corresponding to Top-1 and Top-5 accuracy results of 83.57% and 97.16%, respectively.

TITLE: Two Birds With One Stone: Knowledge-Embedded Temporal Convolutional Transformer for Depression Detection and Emotion Recognition

AUTHOR: Wenbo Zheng, Lan Yan ,Fei-Yue Wang.

ABSTRACT: Depression is a critical problem in modern society that affects an estimated 350 million people worldwide, causing feelings of sadness and a lack of interest and pleasure.

Emotional disorders are gaining interest and are closely entwined with depression, because one contributes to an understanding of the other. Despite the achievements in the two separate tasks of emotion recognition and depression detection, there has not been much prior effort to build a unified model that can connect these two tasks with different modalities, including multimedia (text, audio, and video) and unobtrusive physiological signals (e.g., electroencephalography). We propose a novel temporal convolutional transformer with knowledge embedding to address the joint task of depression detection and emotion recognition. This approach not only learns multimodal embeddings across domains via the temporal convolutional transformer but also exploits special-domain knowledge from medical knowledge graphs to improve the performance of detection and recognition. It is essential that the features learned by our method can be perceived as a priori and are suitable for increasing the performance of other related tasks. Our method illustrates the case of "two birds with one stone" in the sense that two or more tasks can be efficiently handled with our unique model, which captures effective features. Experimental results on ten real-world datasets show that the proposed approach significantly outperforms other state-of-the-art approaches. On the other hand, experiments in which our methodology is applied to other reasoning tasks show that our approach effectively supports model reasoning related to emotion and improves its performance.

TITLE: Abnormal Human Behaviour Detection in Videos: A Review

AUTHOR: Huiyu Mu, Ruizhi Sun, Gang Yuan

ABSTRACT: Modeling human behavior patterns for detecting the abnormal event has become an important domain in recentyears. A lot of efforts have been made for building smart video surveillance systems with the purpose ofscene analysis and making correct semantic inference from the video moving target. Current approaches havetransferred from rule-based to statistical-based methods with the need of efficient recognition of high-levelactivities. This paper presented not only an update expanding previous related researches, but also a study coveredthe behavior representation and the event modeling. Especially, we provided a new perspective for eventmodeling which divided the methods into the following subcategories: modeling normal event, predictionmodel, query model and deep hybrid model. Finally, we exhibited the available datasets and popular evaluationschemes used for abnormal behavior detection in intelligent video surveillance. More researches will promotethe development of abnormal human behavior detection, e.g. deep generative network, weakly-supervised. It isobviously encouraged and dictated by applications of supervising and monitoring in private and public space.The main purpose of this paper is to widely recognize recent available methods and represent the literature ina way of that brings key challenges into notice.

TITLE: A review of state-of-the-art techniques for abnormal human activity recognition
AUTHOR: Chhavi Dhiman [a], Dinesh Kumar Vishwakarma [b]
ABSTRACT: The concept of intelligent visual identification of abnormal human activity has raised the standards of surveillance systems, situation cognizance, homeland safety and smart environments. However, abnormal human activity is highly diverse in itself due to the aspects such as (a) the fundamental definition of anomaly (b) feature representation of an anomaly, (c) its application, and henceforth (d) the dataset. This paper aims to summarize various existing abnormal human activity recognition (AbHAR) handcrafted and deep approaches with the variation of the type of information available such as two-dimensional or three-dimensional data. Features play a vital role in an excellent performance of an AbHAR system. The proposed literature provides feature designs of abnormal human activity recognition in a video with respect to the context or application such as fall detection, Ambient Assistive Living (AAL), homeland security, surveillance or crowd analysis using RGB, depth and skeletal evidence. The key contributions and limitations of every feature design technique, under each category: 2D and 3D AbHAR, in respective contexts are tabulated that will provide insight of various abnormal action detection approaches. Finally, the paper outlines newly added datasets for AbHAR by the researchers with added complexities for method validations.

TITLE: Temporal Action Localization in the Deep Learning Era

AUTHOR :BingluWang,YongqiangZhao ,LeYang, Teng Long, Xuelong Li

ABSTRACT: The temporal action localization research aims to discover action instances from untrimmed videos, representing a fundamental step in the field of intelligent video understanding. With the advent of deep learning, backbone networks have been instrumental in providing representative spatiotemporal features, while the end-to-end learning paradigm has enabled the development of high-quality models through data-driven training. Both supervised and weakly supervised learning approaches have contributed to the rapid progress of temporal action localization, resulting in a multitude of methods and a large body of literature, making a comprehensive survey a pressing necessity. This paper presents a thorough analysis of existing action localization works, offering a well-organized taxonomy that highlights the strengths and weaknesses of each strategy. In the realm of supervised learning, in addition to the anchor mechanism, we introduce a novel classification mechanism to categorize and summarize existing works. Similarly, for weakly supervised learning, we extend the traditional pre-classification and post-classification mechanisms by providing a fresh perspective on enhancement strategies. Furthermore, we shed light on the bottleneck of confidence estimation, a critical yet overlooked aspect of current works. By conducting detailed analyses, this survey serves as a valuable resource for researchers, providing beneficial guidance to newcomers and inspiring seasoned researchers alike.

CHAPTER-3
SYSTEM ANALSIS

3.1 EXISTING SYSTEM

Xie et al. [45] used spatiotemporal representations to learn the posture estimation of college students to identify abnormal behavior. They analyzed the behavior of sleeping and using mobile phones in the classroom. Other researchers [46], [47] have explicitly looked at abnormal behavior in the laboratory. Rashmi et al. [46] apply YOLOv3 to locate and recognize student actions in still images from surveillance video in school laboratories. Unlike Rashmi's image recognition-based analysis of students' abnormal behavior in the laboratory, Banerjee et al. [47] used video. They propose a deep convolutional network architecture to detect and classify the behavioral patterns of students and teachers in computer-enabled laboratories. The above works can significantly demonstrate recognition of abnormal behavior in specific scenarios on campus. However, do not aim to reveal the feasibility of numerous abnormal behaviors in multiple scenarios on campus. Although there is limited research on campus aberrant behavior recognition, it is a form of video understanding. The following is a review of video understanding research relevant to our work.

Although self-attention is added as a submodule to CNNs to improve this temporal modeling video understanding, the ability of remote modeling can be made more robust by applying a pure transformer. These algorithms [27], [29], [30] are related to decomposing spatiotemporal self-attention using different factorized methods. They achieved better results than previous pure CNN and methods for adding self-attention units to videos. However, depending on the global attention modeling, these methods lead to a geometric increase in complexity.

Subsequently, MVTs [28] and the video Swin transformer [31] presented the idea of multi-scale hierarchical modeling by calculating the self-attention of multi-scale windows, which is much lower than the computational complexity of global self-attention. Moreover, it exceeds the previous decomposition space-time modeling methods in terms of accuracy and efficiency. The latest research [32] proposed a multi-view transformer consisting of multiple independent encoders to represent different dimensional input views, fusing information across views through horizontal connections. Although they achieve state-of-the-art performance, their method relies on many unpublic datasets and trained views. It may limit their generalization performance when applied to new videos or tasks beyond their original training scope. Therefore, a fairer comparison is required. In the case of employing ImageNet-21K as pre-training data to initialize network weights, the transformer method established on multi-scale hierarchical modeling [31] has more competitive advantages in video understanding.

DISADVANTAGES

- In the existing work, the system did not find Sensors and wireless data transmission for measuring food safety.
- This system is less performance due to lack of Real-time Prediction Campus Abnormal Behaviour.

3.2 PROPOSED SYSTEM

• We propose a consensus of three temporal segment transformers (TST) based on the video Swin transformer for the new campus abnormal behavior recognition (CABR50) dataset. It enhances the ability to capture motion sequences and model long-range abnormal behavior on campus.

• We perform extensive comparative experiments with state-of-the-art methods for recognizing abnormal campus behaviors. The results show that it is feasible and can improve the accuracy of abnormal campus behavior recognition. In addition, we demonstrate the performance comparison of the TST and previous methods on the UCF-101 dataset. It indicates that our proposed TST model has acceptable generalization performance.

• Our research provides essential technical support for the identification and early warning of abnormal behavior on campus, which plays an essential role in intelligent campus surveillance systems.

ADVANTAGES

❖ The system is more effective since it involves Convolutional Neural Networks (CNNs) method.

- The system finds more ADVANTAGES OF THE system which designed a campus abnormal behavior recognition framework called temporal segment transformer (TST) to exploit temporal action features and achieve video-level global modeling.

CHAPTER-4
SYSTEM REQUIREMENTS

4.1 FUNCTIONAL REUIREMENTS

ASSISTOR: The Assistor Module is a pivotal component within the framework of the Campus Abnormal Behaviour Recognition system, designed to enhance the efficiency and accuracy of anomaly detection processes.

NONLOCAL USER: The Non-local User Module is an essential component integrated into the overarching framework of the Campus Abnormal Behaviour Recognition system, specifically tailored to cater to the needs and interactions of users who are not physically present within the campus premises.

4.2.1 HARDWARE REQUIREMENTS

➢ Processor	- Pentium –IV
➢ RAM	- 4 GB (min)
➢ Hard Disk	- 20 GB
➢ Key Board	- Standard Windows Keyboard
➢ Mouse	- Two or Three Button Mouse
➢ Monitor	- SVGA

4.2.2 SOFTWARE REQUIREMENTS:

Operating system	**:** Windows 7 Ultimate.
Coding Language	**:** Python.
Front-End	**:** Python.
Back-End	**:** Django-ORM
Designing	**:** Html, css, javascript.
Data Base	**:** MyS

CHAPTER-5
SYSTEM STUDY

5.1 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.
Three key considerations involved in the feasibility analysis are

- ♦ ECONOMICAL FEASIBILITY
- ♦ TECHNICAL FEASIBILITY
- ♦ SOCIAL FEASIBILITY

5.2 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

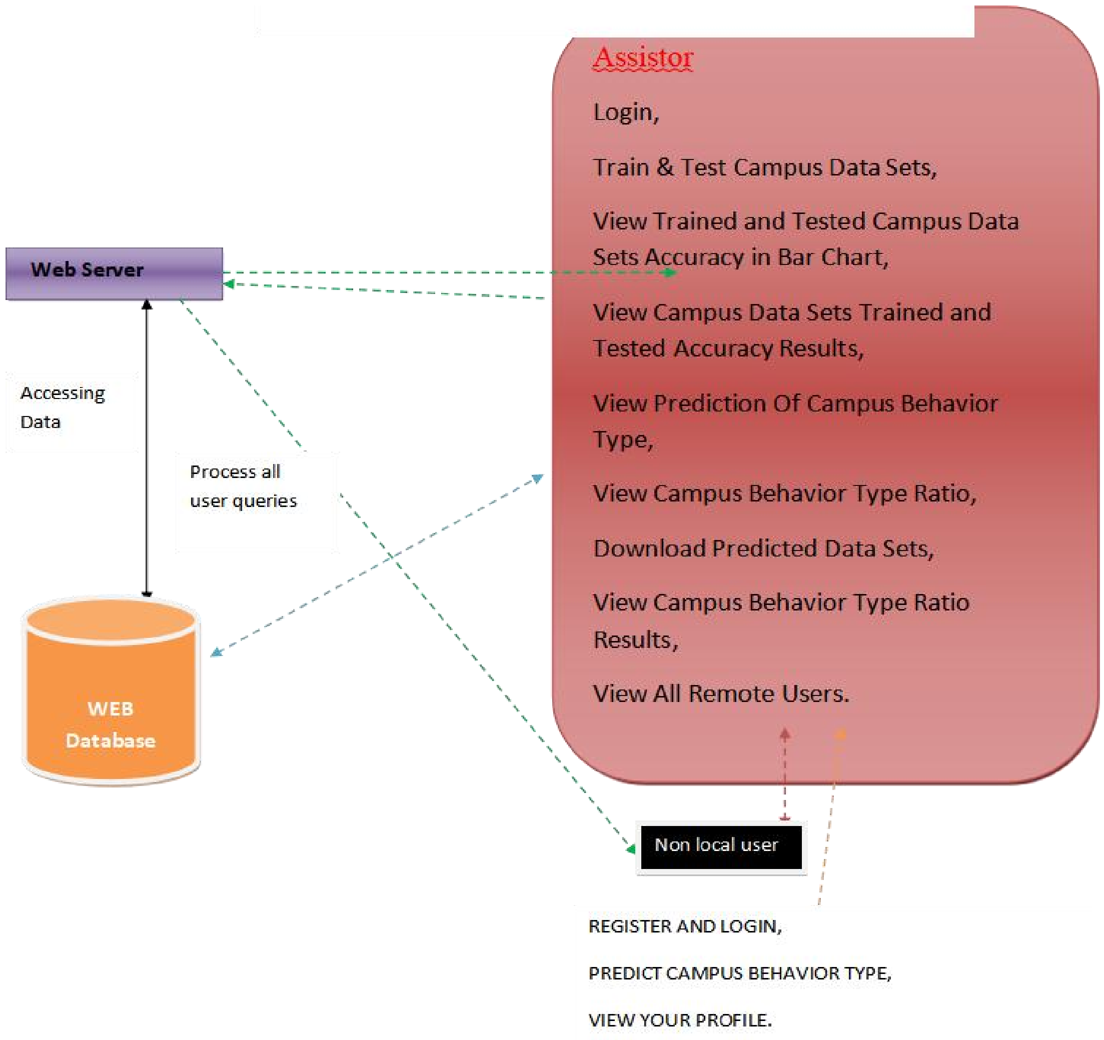
This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

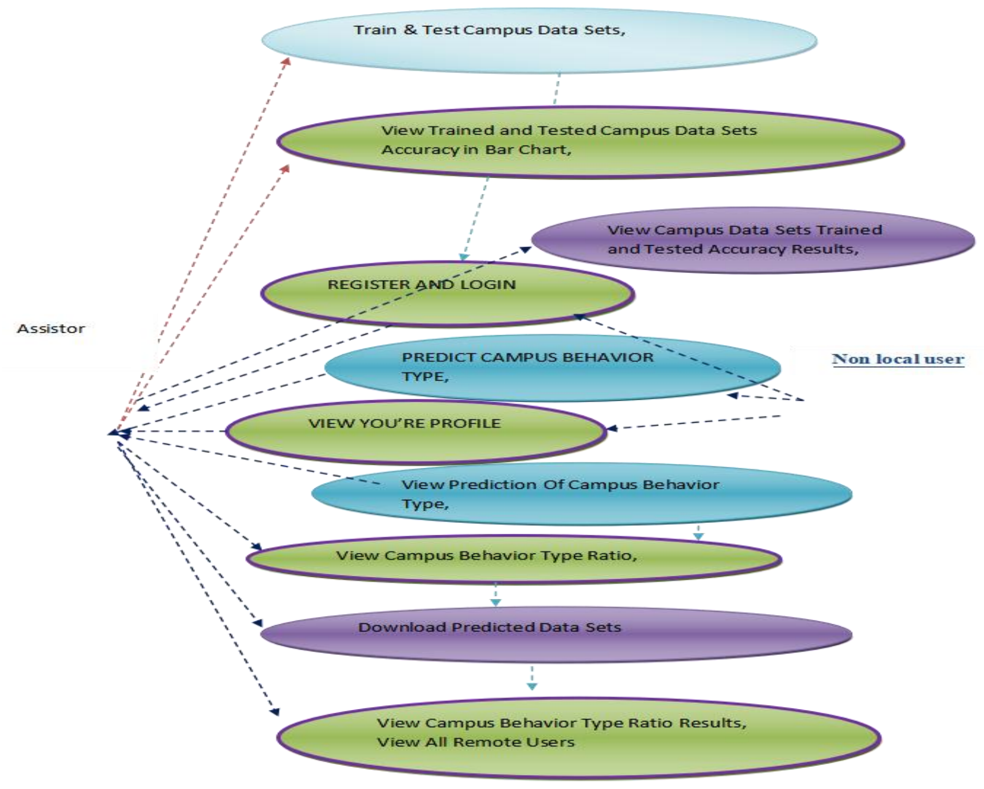
CHAPTER-6
SYSTEM DESIGN

6.1 SYSTEM ARCHITECTURE

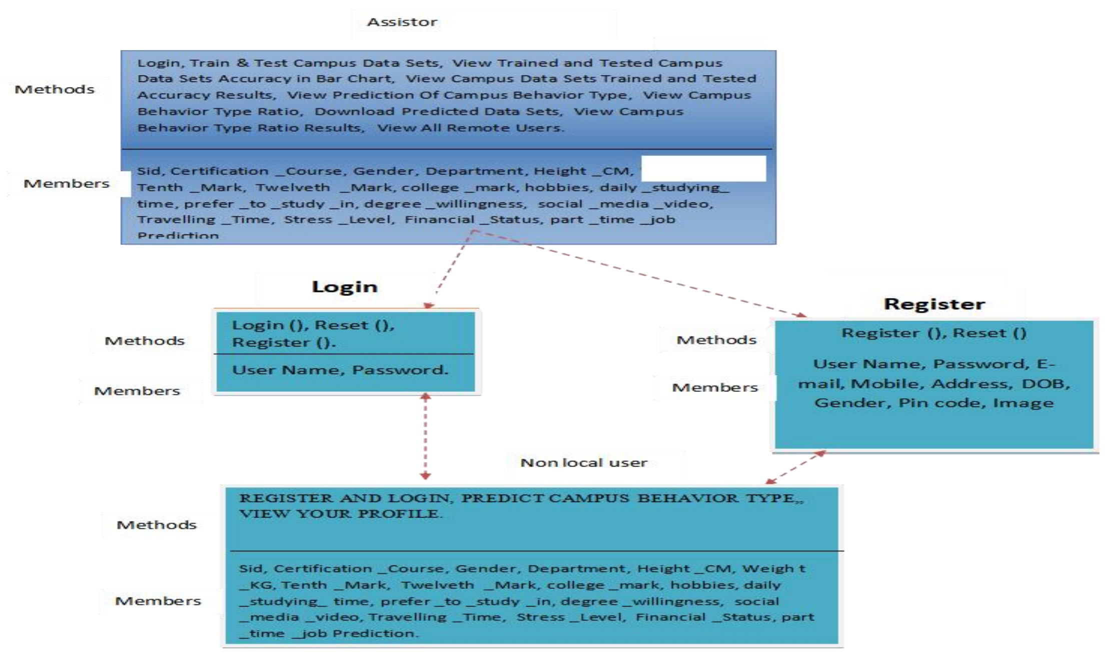

Assistor
Login,
Train & Test Campus Data Sets,
View Trained and Tested Campus Data Sets Accuracy in Bar Chart,
View Campus Data Sets Trained and Tested Accuracy Results,
View Prediction Of Campus Behavior Type,
View Campus Behavior Type Ratio,
Download Predicted Data Sets,
View Campus Behavior Type Ratio Results,
View All Remote Users.
Web Server
Accessing Data
Process all user queries
WEB Database
Non local user
REGISTER AND LOGIN,
PREDICT CAMPUS BEHAVIOR TYPE,
VIEW YOUR PROFILE.

6.2 UML DIAGRAMS

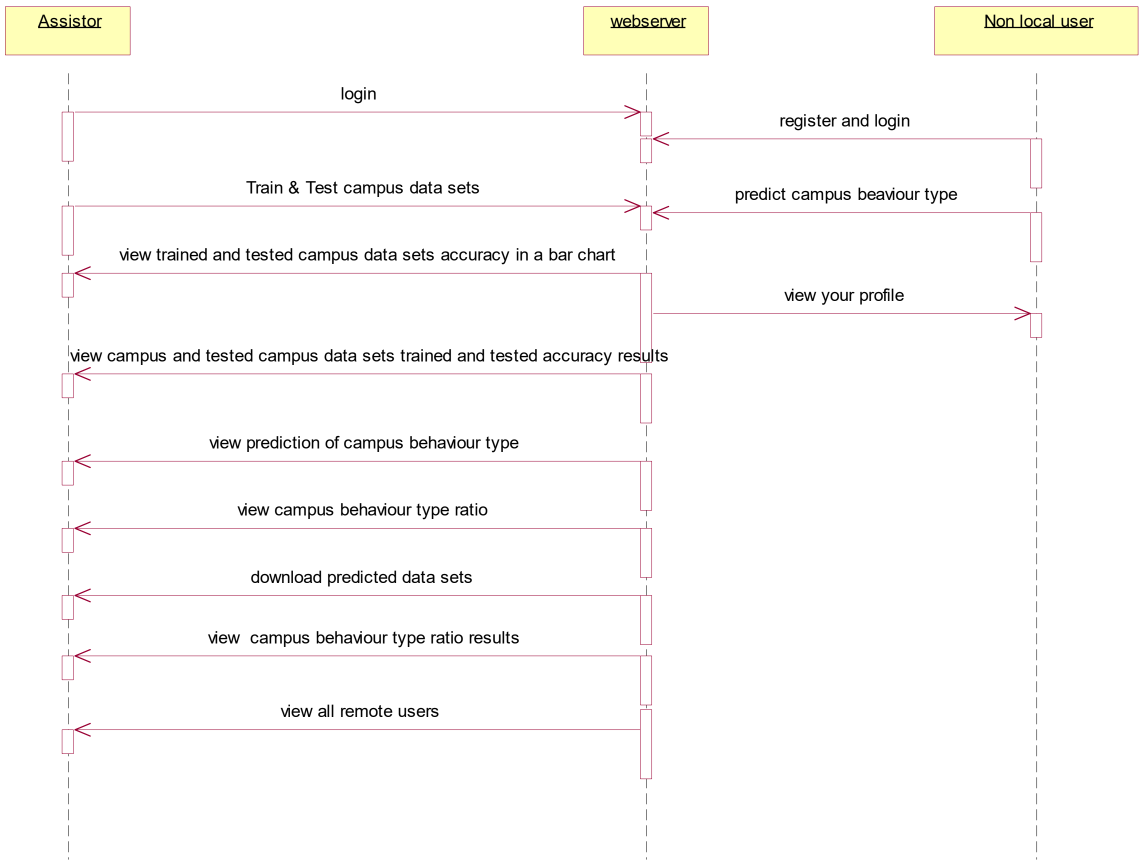
6.2.1 USECASE DIAGRAM


Train & Test Campus Data Sets,
View Trained and Tested Campus Data Sets Accuracy in Bar Chart,
View Campus Data Sets Trained and Tested Accuracy Results,
REGISTER AND LOGIN
Assistor
PREDICT CAMPUS BEHAVIOR TYPE,
Non local user
VIEW YOU'RE PROFILE
View Prediction Of Campus Behavior Type,
View Campus Behavior Type Ratio,
Download Predicted Data Sets
View Campus Behavior Type Ratio Results, View All Remote Users

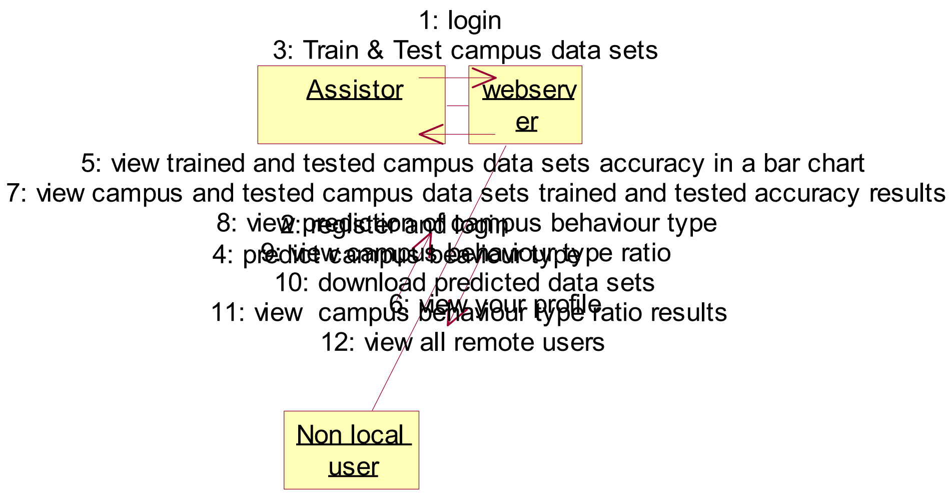
6.2.2 CLASS DIAGRAM


Assistor
Methods
Login, Train & Test Campus Data Sets, View Trained and Tested Campus Data Sets Accuracy in Bar Chart, View Campus Data Sets Trained and Tested Accuracy Results, View Prediction Of Campus Behavior Type, View Campus Behavior Type Ratio, Download Predicted Data Sets, View Campus Behavior Type Ratio Results, View All Remote Users.
Members
Sid, Certification _Course, Gender, Department, Height _CM, Tenth _Mark, Twelveth _Mark, college _mark, hobbies, daily _studying_ time, prefer _to _study _in, degree _willingness, social _media _video, Travelling _Time, Stress _Level, Financial _Status, part _time _job Prediction
Login
Methods
Login (), Reset (), Register ().
Members
User Name, Password.
Register
Methods
Register (), Reset ()
Members
User Name, Password, E-mail, Mobile, Address, DOB, Gender, Pin code, Image
Non local user
Methods
REGISTER AND LOGIN, PREDICT CAMPUS BEHAVIOR TYPE,, VIEW YOUR PROFILE.
Members
Sid, Certification _Course, Gender, Department, Height _CM, Weigh t _KG, Tenth _Mark, Twelveth _Mark, college _mark, hobbies, daily _studying_ time, prefer _to _study _in, degree _willingness, social _media _video, Travelling _Time, Stress _Level, Financial _Status, part _time _job Prediction.

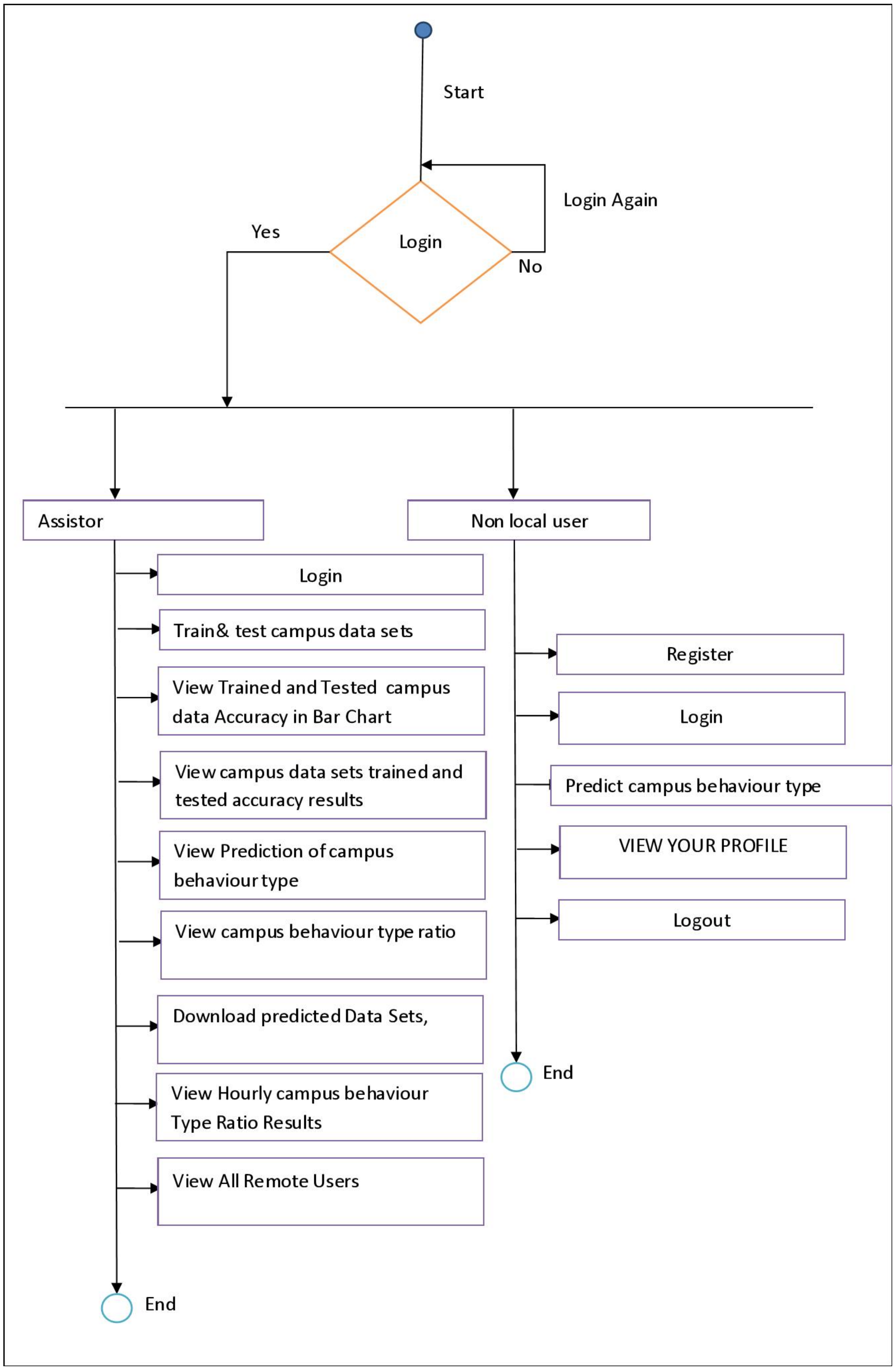
6.2.3 SEQUENCE DIAGRAM


Assistor
webserver
Non local user
login
register and login
Train & Test campus data sets
predict campus beaviour type
view trained and tested campus data sets accuracy in a bar chart
view your profile
view campus and tested campus data sets trained and tested accuracy results
view prediction of campus behaviour type
view campus behaviour type ratio
download predicted data sets
view campus behaviour type ratio results
view all remote users

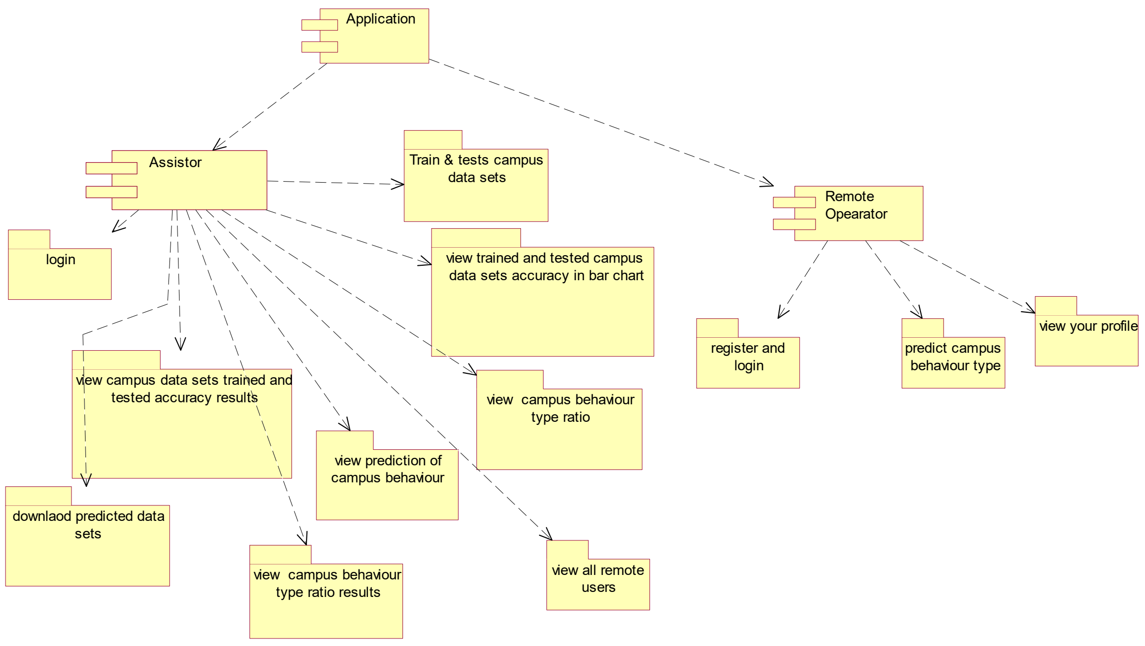
6.2.4 COLLABRATION DIAGRAM


1: login
3: Train & Test campus data sets
Assistor
webserver
5: view trained and tested campus data sets accuracy in a bar chart
7: view campus and tested campus data sets trained and tested accuracy results
8: view prediction of campus behaviour type
2: register and login
4: predict campus beaviour type
9: view campus behaviour type ratio
10: download predicted data sets
6: view your profile
11: view campus behaviour type ratio results
12: view all remote users
Non local user

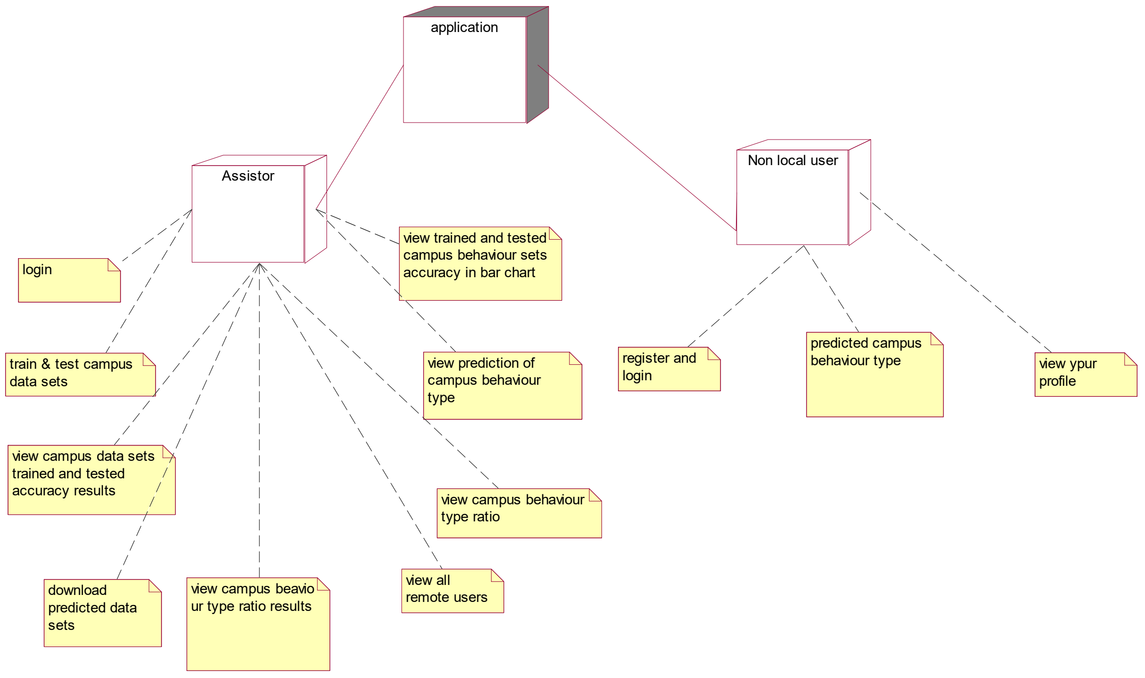
6.2.5 ACTIVITY DIAGRAM


Start
Login Again
Yes
Login
No
Assistor
Non local user
Login
Train& test campus data sets
View Trained and Tested campus data Accuracy in Bar Chart
View campus data sets trained and tested accuracy results
View Prediction of campus behaviour type
View campus behaviour type ratio
Download predicted Data Sets,
View Hourly campus behaviour Type Ratio Results
View All Remote Users
End
Register
Login
Predict campus behaviour type
VIEW YOUR PROFILE
Logout
End

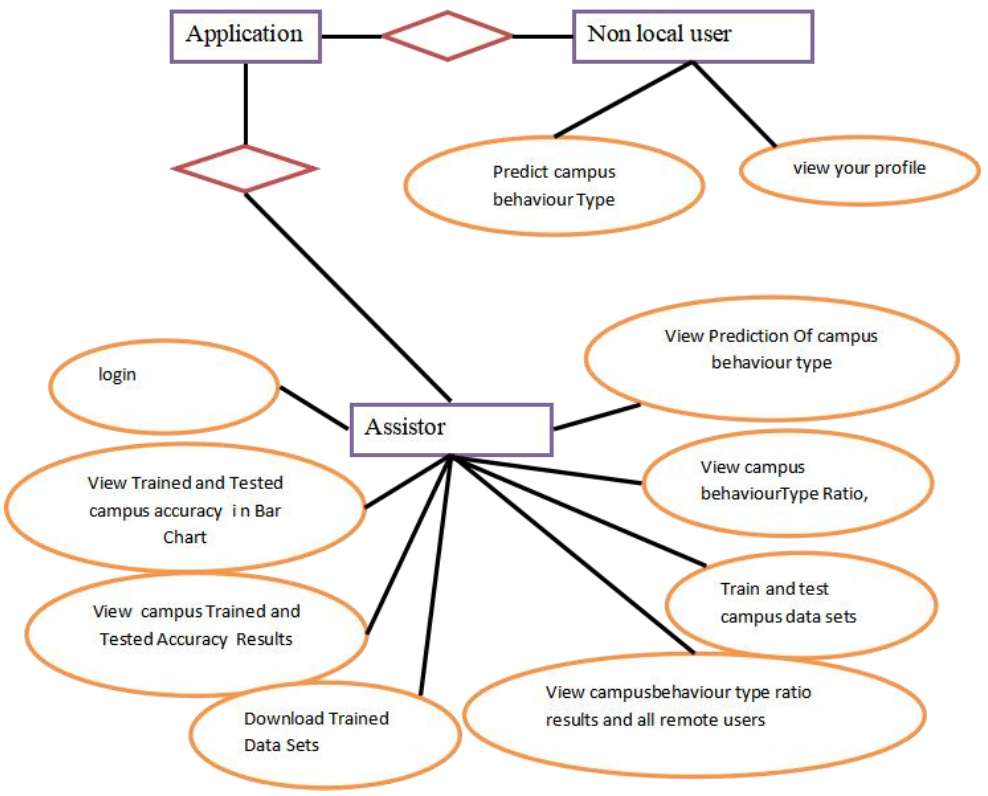
6..2.6 COMPONENT DIAGRAM


Application
Assistor
Train & tests campus data sets
Remote Opearator
login
view trained and tested campus data sets accuracy in bar chart
register and login
predict campus behaviour type
view your profile
view campus data sets trained and tested accuracy results
view campus behaviour type ratio
view prediction of campus behaviour
downlaod predicted data sets
view campus behaviour type ratio results
view all remote users

6.2.7 DEPLOYMENT DIAGRAM


application
Assistor
Non local user
login
view trained and tested campus behaviour sets accuracy in bar chart
train & test campus data sets
view prediction of campus behaviour type
register and login
predicted campus behaviour type
view ypur profile
view campus data sets trained and tested accuracy results
view campus behaviour type ratio
download predicted data sets
view campus beavio ur type ratio results
view all remote users

6.2.8 ER DIAGRAM


Application
Non local user
Predict campus behaviour Type
view your profile
View Prediction Of campus behaviour type
login
Assistor
View Trained and Tested campus accuracy in Bar Chart
View campus behaviourType Ratio,
Train and test campus data sets
View campus Trained and Tested Accuracy Results
View campusbehaviour type ratio results and all remote users
Download Trained Data Sets

6.2.9 DATA DICTIONARY

Database: Campus Abnormal Behaviour Recognition With Temporal Segment Transformers _methods

Table name: auth_group

Column	Data Type	Constraints	Description
id	Int(11)	Primary key	Unique Identifier
name	Varchar(1000)	Not null	name

Column	Data Type	Constraints	Description
id	Int(11)	Primary key	Unique Identifier
Group id	Int(11)	Primary Key	Unique Identifier

Permission_id	Int(11)	Primary Key	Unique Identifier

Table Name: Auth_group_Permission

Table Name: auth_permission

Column	Data Type	Constraints	Description
id	Int(11)	Primary key	Unique Identifier
Name	Int(255)	Not Null	name
Content_type_id	Int(11)	Primary Key	Unique Identifier
Code name	Int(100)	Not Null	name

Table Name: auth_User

Column	Data Type	Constraints	Description
id	Int(11)	Primary key	Unique Identifier
Password	varchar(128)	Not Null	password
Last_Login	Datetime(6)	Not Null	Last login
Is_superuser	tinyint(1)	Not Null	Name
username	Varchar(150)	Not Null	Username
lastname	Varchar(30)	Not Null	Lastname
email	Varchar(150)	Not Null	Email id
Is_staff	Tinyint(1)	Not Null	Staff
Is_active	Tinyint(1)	Not Null	Active
Date_joined	Datetime(6)	Not Null	Date and time

Table Name: auth_user_groups

Column	Data Type	Constraints	Description
id	Int(11)	Primary key	Unique Identifier

User_id	Int(11)	Primary Key	Unique Identifier
Group_id	Int(11)	Primary Key	Unique Identifier

CHAPTER-7
INPUT AND OUTPUT DESIGN

7.1 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

7.1.1 OBJECTIVES

1.Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3.When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow.

7.2 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2.Select methods for presenting information.

3.Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

CHAPTER-8
IMPLEMENTATION

8.1 MODULES

- Assistor
- Non local user

8.1.1 MODULE DESCRIPTION: This module acts as a facilitator, assisting in the analysi of complex temporal data streams generated by various sensors and surveillance devices deployed across the campus infrastructure. By utilizing machine learning algorithms and pattern recognition methodologies, the Assistor Module dynamically adapts to evolving patterns of normalcy, enabling it to effectively discern deviations indicative of potential security threats, safety hazards, or abnormal activities.

NONLOCAL USER: The Non-local User Module is an essential component integrated into the overarching framework of the Campus Abnormal Behaviour Recognition system, specifically tailored to cater to the needs and interactions of users who are not physically present within the campus premises. As a critical element within the system, the Non-local User Module facilitates seamless engagement and collaboration between remote stakeholders, ensuring comprehensive monitoring and response capabilities across distributed environments.

Functioning as a specialized interface, the Non-local User Module extends the reach of the Campus Abnormal Behaviour Recognition system beyond the confines of the physical campus, enabling remote users to access and interact with the system's functionalities from any location with internet connectivity. Whether they are security personnel overseeing multiple campuses, administrative staff working off-site, or law enforcement agencies coordinating responses from remote command centers, the Non-local User Module provides a centralized platform for real-time monitoring, analysis, and decision-making.

CHAPTER-9

SOFTWARE ENVIRONMENT

9.1 PYTHON

Python is a **high-level, interpreted**, **interactive** and **object-oriented scripting language**. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted:** Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive:** You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented:** Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language:** Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

Python's features include:

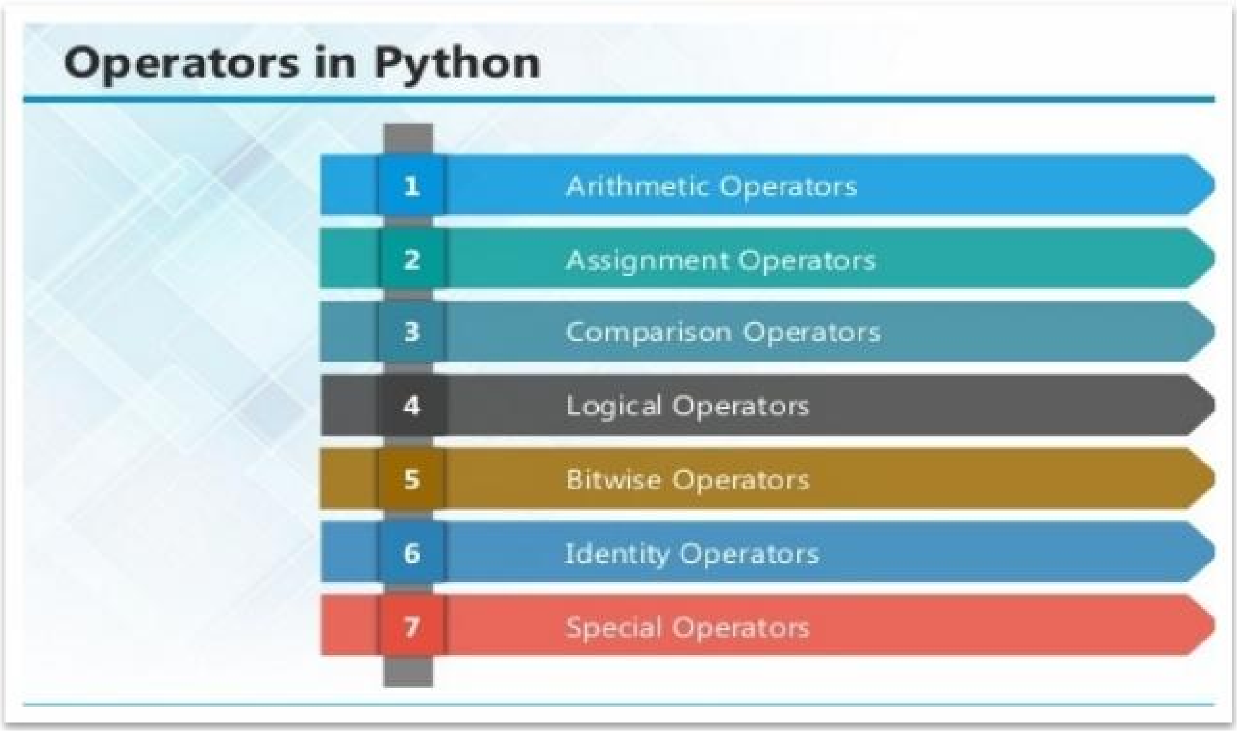
- **Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read:** Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain:** Python's source code is fairly easy-to-maintain.
- **A broad standard library:** Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

- **Interactive Mode:** Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable:** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable:** You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases:** Python provides interfaces to all major commercial databases.
- **GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable:** Python provides a better structure and support for large programs than shell scripting.

Python has a big list of good features:

- It supports functional and structured programming methods as well as OOP.

- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- IT supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.



Operators in Python

1. Arithmetic Operators
2. Assignment Operators
3. Comparison Operators
4. Logical Operators
5. Bitwise Operators
6. Identity Operators
7. Special Operators

ARITHMETIC OPERATORS

Operator	Description	Example
+ Addition	Adds values on either side of the operator.	a + b = 30

- Subtraction	Subtracts right hand operand from left hand operand.	a – b = -10
* Multiplication	Multiplies values on either side of the operator	a * b = 200
/ Division	Divides left hand operand by right hand operand	b / a = 2
% Modulus	Divides left hand operand by right hand operand and returns remainder	b % a = 0
** Exponent	Performs exponential (power) calculation on operators	a**b =10 to the power 20
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity):	9//2 = 4 and 9.0//2.0 = 4.0, -11//3 = -4, -11.0//3 = -4.0

ASSIGNMENT OPERATOR

Operator	Description	Example
=	Assigns values from right side operands to left side operand	c = a + b assigns value of a + b into c
+= Add AND	It adds right operand to the left operand and assign the result to left operand	c += a is equivalent to c = c + a
-= Subtract AND	It subtracts right operand from the left operand and assign the result to left operand	c -= a is equivalent to c = c - a
*= Multiply AND	It multiplies right operand with the left operand and assign the result to left operand	c *= a is equivalent to c = c * a
/= Divide AND	It divides left operand with the right operand and assign the result to left operand	c /= a is equivalent to c = c / ac /= a is equivalent to c = c / a

%= Modulus AND	It takes modulus using two operands and assign the result to left operand	c %= a is equivalent to c = c % a
**= Exponent AND	Performs exponential (power) calculation on operators and assign value to the left operand	c **= a is equivalent to c = c ** a
//= Floor Division	It performs floor division on operators and assign value to the left operand	c //= a is equivalent to c = c // a

IDENTITY OPERATOR

Operator	**Description**	**Example**
is	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.	x is y, here **is** results in 1 if id(x) equals id(y).
is not	Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.	x is not y, here **is not** results in 1 if id(x) is not equal to id(y

COMPARISON OPERATOR

Operator	Description	Example
& Binary AND	Operator copies a bit to the result if it exists in both operands	(a & b) (means 0000 1100)
| Binary OR	It copies a bit if it exists in either operand.	(a | b) = 61 (means 0011 1101)
^ Binary XOR	It copies the bit if it is set in one operand but not both.	(a ^ b) = 49 (means 0011 0001)
~ Binary Ones Complement	It is unary and has the effect of 'flipping' bits.	(~a) = -61 (means 1100 0011 in 2's complement form due to a signed binary number.
<< Binary Left Shift	The left operands value is moved left by the number of bits specified by the right operand.	a << 2 = 240 (means 1111 0000)
>> Binary Right Shift	The left operands value is moved right by the number of bits specified by the right operand.	a >> 2 = 15 (means 0000 1111)

LOGICAL OPERATOR

Operator	Description	Example

and Logical AND	If both the operands are true then condition becomes true.	(a and b) is true.
or Logical OR	If any of the two operands are non-zero then condition becomes true.	(a or b) is true.
not Logical NOT	Used to reverse the logical state of its operand.	Not(a and b) is false.

Membership Operators

Operator	Description	Example
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x in y, here in results in a 1 if x is a member of sequence y.
not in	Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.	x not in y, here not in results in a 1 if x is not a member of sequence y.

Python Operators Precedence

Operator	Description
**	Exponentiation (raise to the power)
~ + -	Complement, unary plus and minus (method names for the last two are +@ and -@)

* / % //	Multiply, divide, modulo and floor division
+ -	Addition and subtraction
>> <<	Right and left bitwise shift
&	Bitwise 'AND'
^ |	Bitwise exclusive `OR' and regular `OR'
<= < > >=	Comparison operators
<> == !=	Equality operators
= %= /= //= -= += *= **=	Assignment operators
is is not	Identity operators
in not in	Membership operators
not or and	Logical operators

LIST

The list is a most versatile data type available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example −

```
list1 = ['physics', 'chemistry', 1997, 2000];

list2 = [1, 2, 3, 4, 5 ];

list3 = ["a", "b", "c", "d"]
```

Basic List Operations

Lists respond to the + and * operators much like strings; they mean concatenation and repetition here too, except that the result is a new list, not a string.

Python Expression	Results	Description
len([1, 2, 3])	3	Length
[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]	Concatenation
['Hi!'] * 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']	Repetition
3 in [1, 2, 3]	True	Membership
for x in [1, 2, 3]: print x,	1 2 3	Iteration

Built-in List Functions & Methods:

Python includes the following list functions −

SN	Function with Description
1	cmp(list1, list2) Compares elements of both lists.
2	len(list) Gives the total length of the list.

3	max(list) Returns item from the list with max value.
4	min(list) Returns item from the list with min value.
5	list(seq) Converts a tuple into list.

Python includes following list methods

SN	Methods with Description
1	list.append(obj) Appends object obj to list
2	list.count(obj) Returns count of how many times obj occurs in list
3	list. extend(seq) Appends the contents of seq to list
4	list.index(obj) Returns the lowest index in list that obj appears
5	list.insert(index, obj) Inserts object obj into list at offset index
6	list.pop(obj=list[-1]) Removes and returns last object or obj from list

7	list.remove(obj) Removes object obj from list
8	list.reverse() Reverses objects of list in place
9	list.sort([func]) Sorts objects of list, use compare function if given

TUPLES

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally we can put these comma-separated values between parentheses also. For example −

```
tup1 = ('physics', 'chemistry', 1997, 2000);

tup2 = (1, 2, 3, 4, 5 );

tup3 = "a", "b", "c", "d";
```

The empty tuple is written as two parentheses containing nothing −

```
tup1 = ();
```

To write a tuple containing a single value you have to include a comma, even though there is only one value −

```
tup1 = (50,);
```

Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.

- **Accessing Values in Tuples:**

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example –

```
tup1 = ('physics', 'chemistry', 1997, 2000);

tup2 = (1, 2, 3, 4, 5, 6, 7 );

print "tup1[0]: ", tup1[0]

print "tup2[1:5]: ", tup2[1:5]
```

When the code is executed, it produces the following result −

```
tup1[0]:  physics

tup2[1:5]:  [2, 3, 4, 5]
```

Updating Tuples:

Tuples are immutable which means you cannot update or change the values of tuple elements. We are able to take portions of existing tuples to create new tuples as the following example demonstrates −

```
tup1 = (12, 34.56);

tup2 = ('abc', 'xyz');

tup3 = tup1 + tup2;

print tup3
```

When the above code is executed, it produces the following result −

```
(12, 34.56, 'abc', 'xyz')
```

Delete Tuple Elements

Removing individual tuple elements is not possible. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded.

To explicitly remove an entire tuple, just use the **del** statement. For example:

```
tup = ('physics', 'chemistry', 1997, 2000);

print tup

del tup;

print "After deleting tup : "

print tup
```

Basic Tuples Operations:

Python Expression	Results	Description
len((1, 2, 3))	3	Length
(1, 2, 3) + (4, 5, 6)	(1, 2, 3, 4, 5, 6)	Concatenation
('Hi!',) * 4	('Hi!', 'Hi!', 'Hi!', 'Hi!')	Repetition
3 in (1, 2, 3)	True	Membership
for x in (1, 2, 3): print x,	1 2 3	Iteration

Built-in Tuple Functions

SN	Function with Description
1	cmp(tuple1, tuple2):Compares elements of both tuples.
2	len(tuple):Gives the total length of the tuple.
3	max(tuple):Returns item from the tuple with max value.
4	min(tuple):Returns item from the tuple with min value.
5	tuple(seq):Converts a list into tuple.

DICTIONARY

Each key is separated from its value by a colon (:), the items are separated by commas, and the whole thing is enclosed in curly braces. An empty dictionary without any items is written with just two curly braces, like this: {}.

Keys are unique within a dictionary while values may not be. The values of a dictionary can be of any type, but the keys must be of an immutable data type such as strings, numbers, or tuples.

Accessing Values in Dictionary:

To access dictionary elements, you can use the familiar square brackets along with the key to obtain its value. Following is a simple example −

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}


print "dict['Name']: ", dict['Name']
print "dict['Age']: ", dict['Age']
```

Result –

```
dict['Name']:  Zara

dict['Age']:  7
```

Updating Dictionary

We can update a dictionary by adding a new entry or a key-value pair, modifying an existing entry, or deleting an existing entry as shown below in the simple example −

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}



dict['Age'] = 8; # update existing entry

dict['School'] = "DPS School"; # Add new entry

print "dict['Age']: ", dict['Age']

print "dict['School']: ", dict['School']
```

Result −

```
dict['Age']:  8

dict['School']:  DPS School
```

Delete Dictionary Elements

We can either remove individual dictionary elements or clear the entire contents of a dictionary. You can also delete entire dictionary in a single operation.

To explicitly remove an entire dictionary, just use the **del** statement. Following is a simple example –

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
```

```
del dict['Name']; # remove entry with key 'Name'

dict.clear();     # remove all entries in dict

del dict ;        # delete entire dictionary


print "dict['Age']: ", dict['Age']

print "dict['School']: ", dict['School']
```

Built-in Dictionary Functions & Methods –

Python includes the following dictionary functions –

SN	Function with Description
1	cmp(dict1, dict2) Compares elements of both dict.
2	len(dict) Gives the total length of the dictionary. This would be equal to the number of items in the dictionary.
3	str(dict) Produces a printable string representation of a dictionary
4	type(variable) Returns the type of the passed variable. If passed variable is dictionary, then it would

	return a dictionary type.

Python includes following dictionary methods −

SN	**Methods with Description**
1	dict.clear():Removes all elements of dictionary *dict*
2	dict. Copy():Returns a shallow copy of dictionary *dict*
3	dict.fromkeys():Create a new dictionary with keys from seq and values *set* to *value*.
4	dict.get(key, default=None):For *key* key, returns value or default if key not in dictionary
5	dict.has_key(key):Returns *true* if key in dictionary *dict*, *false* otherwise
6	dict.items():Returns a list of *dict*'s (key, value) tuple pairs
7	dict.keys():Returns list of dictionary dict's keys
8	dict.setdefault(key, default=None):Similar to get(), but will set dict[key]=default if *key* is not already in dict
9	dict.update(dict2):Adds dictionary *dict2*'s key-values pairs to *dict*
10	dict.values():Returns list of dictionary *dict*'s values

A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing. Python gives you many built-in functions like print(), etc. but you can also create your own functions. These functions are called *user-defined functions.*

Defining a Function

Simple rules to define a function in Python.

- Function blocks begin with the keyword def followed by the function name and parentheses (( )).
- Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.
- The first statement of a function can be an optional statement - the documentation string of the function or *docstring*.
- The code block within every function starts with a colon (:) and is indented.
- The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

```
def functionname( parameters ):
  "function_docstring"
  function_suite
  return [expression]
```

Calling a Function

Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code.Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt. Following is the example to call printme() function −

```
# Function definition is here

def printme( str ):

   "This prints a passed string into this function"

   print str

   return;

# Now you can call printme function

printme("I'm first call to user defined function!")

printme("Again second call to the same function")
```

When the above code is executed, it produces the following result −

```
I'm first call to user defined function!

Again second call to the same function
```

Function Arguments

You can call a function by using the following types of formal arguments:

- Required arguments
- Keyword arguments
- Default arguments
- Variable-length arguments

Scope of Variables

All variables in a program may not be accessible at all locations in that program. This depends on where you have declared a variable.

The scope of a variable determines the portion of the program where you can access a particular identifier. There are two basic scopes of variables in Python −

Global variables   Local variables

Global vs. Local variables

Variables that are defined inside a function body have a local scope, and those defined outside have a global scope.

This means that local variables can be accessed only inside the function in which they are declared, whereas global variables can be accessed throughout the program body by all functions. When you call a function, the variables declared inside it are brought into scope. Following is a simple example −

```
total = 0; # This is global variable.

# Function definition is here

def sum( arg1, arg2 ):

  # Add both the parameters and return them."

  total = arg1 + arg2; # Here total is local variable.

  print "Inside the function local total : ", total

  return total;

sum( 10, 20 );

print "Outside the function global total : ", total
```

Result −

```
Inside the function local total :  30

Outside the function global total :  0
```

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference.Simply, a module is a file consisting of Python code. A module can define functions, classes and variables. A module can also include runnable code.

Example:

The Python code for a module named *aname* normally resides in a file named *aname.py*. Here's an example of a simple module, support.py

```
def print_func( par ):

  print "Hello : ", par

  return
```

The *import* Statement

The *import* has the following syntax:

```
import module1[, module2[,... moduleN]
```

When the interpreter encounters an import statement, it imports the module if the module is present in the search path. A search path is a list of directories that the interpreter searches before importing a module. For example, to import the module support.py, you need to put the following command at the top of the script −

A module is loaded only once, regardless of the number of times it is imported. This prevents the module execution from happening over and over again if multiple imports occur.

Packages in Python

A package is a hierarchical file directory structure that defines a single Python application environment that consists of modules and sub packages and sub-sub packages.

Consider a file *Pots.py* available in *Phone* directory. This file has following line of source code –

```
def Pots():

  print "I'm Pots Phone"
```

Similar way, we have another two files having different functions with the same name as above –

- *Phone/Isdn.py* file having function Isdn()
- *Phone/G3.py* file having function G3()

Now, create one more file __init__.py in *Phone* directory −

- Phone/__init__.py

To make all of your functions available when you've imported Phone,to put explicit import statements in __init__.py as follows −

```
from Pots import Pots

from Isdn import Isdn

from G3 import G3
```

After you add these lines to __init__.py, you have all of these classes available when you import the Phone package.

```
# Now import your Phone Package.

import Phone

Phone.Pots()

Phone.Isdn()

Phone.G3()
```

RESULT:

```
I'm Pots Phone

I'm 3G Phone

I'm ISDN Phone
```

In the above example, we have taken example of a single functions in each file, but you can keep multiple functions in your files. You can also define different Python classes in those files and then you can create your packages out of those classes.

This chapter covers all the basic I/O functions available in Python.

Printing to the Screen

The simplest way to produce output is using the *print* statement where you can pass zero or more expressions separated by commas. This function converts the expressions you pass into a string and

writes the result to standard output as follows −

```
print "Python is really a great language,", "isn't it?"
```

Result:

```
Python is really a great language, isn't it?
```

Reading Keyboard Input

Python provides two built-in functions to read a line of text from standard input, which by default comes from the keyboard. These functions are −

- raw_input
- input

The *raw_input* Function

The *raw_input([prompt])* function reads one line from standard input and returns it as a string (removing the trailing newline).

```
str = raw_input("Enter your input: ");

print "Received input is : ", str
```

This prompts you to enter any string and it would display same string on the screen. When I typed "Hello Python!", its output is like this −

```
Enter your input: Hello Python

Received input is :  Hello Python
```

The *input* Function

The *input([prompt])* function is equivalent to raw_input, except that it assumes the input is a valid Python expression and returns the evaluated result to you.

```
str = input("Enter your input: ");

print "Received input is : ", str
```

This would produce the following result against the entered input −

```
Enter your input: [x*5 for x in range(2,10,2)]

Recieved input is :  [10, 20, 30, 40]
```

Opening and Closing Files

Until now, you have been reading and writing to the standard input and output. Now, we will see how to use actual data files.

Python provides basic functions and methods necessary to manipulate files by default. You can do most of the file manipulation using a **file** object.

The *open* Function

Before you can read or write a file, you have to open it using Python's built-in *open()* function. This function creates a **file** object, which would be utilized to call other support methods associated with it.

Syntax

```
file object = open(file_name [, access_mode][, buffering])
```

Here are parameter details:

- **file_name:** The file_name argument is a string value that contains the name of the file that you want to access.

- **access_mode:** The access_mode determines the mode in which the file has to be opened, i.e., read, write, append, etc. A complete list of possible values is given below in the table. This is optional parameter and the default file access mode is read (r).

- **buffering:** If the buffering value is set to 0, no buffering takes place. If the buffering value is 1, line buffering is performed while accessing a file. If you specify the buffering value as an integer greater than 1, then buffering action is performed with the indicated buffer size. If negative, the buffer size is the system default(default behavior).

Here is a list of the different modes of opening a file −

Modes	Description
r	Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode.
rb	Opens a file for reading only in binary format. The file pointer is placed at the beginning of the file. This is the default mode.

r+	Opens a file for both reading and writing. The file pointer placed at the beginning of the file.
rb+	Opens a file for both reading and writing in binary format. The file pointer placed at the beginning of the file.
w	Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
wb	Opens a file for writing only in binary format. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
w+	Opens a file for both writing and reading. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
wb+	Opens a file for both writing and reading in binary format. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
a	Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
ab	Opens a file for appending in binary format. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
a+	Opens a file for both appending and reading. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.
ab+	Opens a file for both appending and reading in binary format. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.

The *file* Object Attributes

Once a file is opened and you have one *file* object, you can get various information related to that file.

Here is a list of all attributes related to file object:

Attribute	Description
file.closed	Returns true if file is closed, false otherwise.
file.mode	Returns access mode with which file was opened.
file.name	Returns name of the file.
file.softspace	Returns false if space explicitly required with print, true otherwise.

Example

```
# Open a file

fo = open("foo.txt", "wb")

print "Name of the file: ", fo.name

print "Closed or not : ", fo.closed

print "Opening mode : ", fo.mode

print "Softspace flag : ", fo.softspace
```

This produces the following result –

```
Name of the file:  foo.txt

Closed or not :  False
```

```
Opening mode :  wb

Softspace flag :  0
```

The *close()* Method

The close() method of a *file* object flushes any unwritten information and closes the file object, after which no more writing can be done.Python automatically closes a file when the reference object of a file is reassigned to another file. It is a good practice to use the close() method to close a file

Syntax

```
fileObject.close();
```

Example

```
# Open a file

fo = open("foo.txt", "wb")

print "Name of the file: ", fo.name

# Close opend file

fo.close()
```

Result −

```
Name of the file:  foo.txt
```

Reading and Writing Files

The *file* object provides a set of access methods to make our lives easier. We would see how to use *read()* and *write()* methods to read and write files.

The *write()* Method

The *write()* method writes any string to an open file. It is important to note that Python strings can have binary data and not just text.The write() method does not add a newline character ('\n') to the end of the string **Syntax**

```
fileObject.write(string);
```

Here, passed parameter is the content to be written into the opened file. **Example**

```
# Open a file

fo = open("foo.txt", "wb")

fo.write( "Python is a great language.\nYeah its great!!\n");



# Close opend file

fo.close()
```

The above method would create *foo.txt* file and would write given content in that file and finally it would close that file. If you would open this file, it would have following content.

```
Python is a great language.

Yeah its great!!
```

The *read()* Method

The *read()* method reads a string from an open file. It is important to note that Python strings can have binary data. apart from text data.

Syntax

```
fileObject.read([count]);
```

Here, passed parameter is the number of bytes to be read from the opened file. This method starts reading from the beginning of the file and if *count* is missing, then it tries to read as much as possible, maybe until the end of file.

Example

Let's take a file *foo.txt*, which we created above.

```
# Open a file
fo = open("foo.txt", "r+")
str = fo.read(10);
print "Read String is : ", str
# Close opend file
fo.close()
```

This produces the following result −

```
Read String is :  Python is
```

File Positions

The *tell()* method tells you the current position within the file; in other words, the next read or write will occur at that many bytes from the beginning of the file.

32

The *seek(offset[, from])* method changes the current file position. The *offset* argument indicates the number of bytes to be moved. The *from* argument specifies the reference position from where the bytes are to be moved.

If *from* is set to 0, it means use the beginning of the file as the reference position and 1 means use the current position as the reference position and if it is set to 2 then the end of the file would be taken as the reference position.

Example

Let us take a file *foo.txt*, which we created above.

```
# Open a file
fo = open("foo.txt", "r+")
str = fo.read(10);
```

```
print "Read String is : ", str


# Check current position

position = fo.tell();

print "Current file position : ", position


# Reposition pointer at the beginning once again

position = fo.seek(0, 0);

str = fo.read(10);

print "Again read String is : ", str

# Close opend file

fo.close()
```

This produces the following result −

```
Read String is :  Python is

Current file position :  10

Again read String is :  Python is
```

Renaming and Deleting Files

Python **os** module provides methods that help you perform file-processing operations, such as renaming and deleting files.

To use this module you need to import it first and then you can call any related functions.

The rename() Method

The *rename()* method takes two arguments, the current filename and the new filename.

Syntax

```
os.rename(current_file_name, new_file_name)
```

Example

Following is the example to rename an existing file *test1.txt*:

```
import os


# Rename a file from test1.txt to test2.txt

os.rename( "test1.txt", "test2.txt" )
```

The *remove()* Method

You can use the *remove()* method to delete files by supplying the name of the file to be deleted as the argument.

Syntax

```
os.remove(file_name)
```

Example

Following is the example to delete an existing file *test2.txt* –

```
#!/usr/bin/python

import os


# Delete file test2.txt

os.remove("text2.txt")
```

Directories in Python

All files are contained within various directories, and Python has no problem handling these too. The **os** module has several methods that help you create, remove, and change directories.

The *mkdir()* Method

You can use the *mkdir()* method of the **os** module to create directories in the current directory. You need to supply an argument to this method which contains the name of the directory to be created.

Syntax

```
os.mkdir("newdir")
```

Example

Following is the example to create a directory *test* in the current directory −

```
#!/usr/bin/python

import os


# Create a directory "test"

os.mkdir("test")
```

The *chdir()* Method

You can use the *chdir()* method to change the current directory. The chdir() method takes an argument, which is the name of the directory that you want to make the current directory.

Syntax

```
os.chdir("newdir")
```

Example

Following is the example to go into "/home/newdir" directory −

```
#!/usr/bin/python

import os


# Changing a directory to "/home/newdir"

os.chdir("/home/newdir")
```

The *getcwd()* Method

The *getcwd()* method displays the current working directory.

Syntax

```
os.getcwd()
```

Example

Following is the example to give current directory –

```
import os


# This would give location of the current directory

os.getcwd()
```

The *rmdir()* Method

The *rmdir()* method deletes the directory, which is passed as an argument in the method.

Before removing a directory, all the contents in it should be removed.

Syntax:

```
os.rmdir('dirname')
```

Example

Following is the example to remove "/tmp/test" directory. It is required to give fully qualified name of the directory, otherwise it would search for that directory in the current directory.

```
import os

# This would  remove "/tmp/test"  directory.

os.rmdir( "/tmp/test"  )
```

File & Directory Related Methods

There are three important sources, which provide a wide range of utility methods to handle and manipulate files & directories on Windows and Unix operating systems. They are as follows −

- File Object Methods: The *file* object provides functions to manipulate files.

- OS Object Methods: This provides methods to process files as well as directories.

Python provides two very important features to handle any unexpected error in your Python programs and to add debugging capabilities in them −

- **Exception Handling:** This would be covered in this tutorial. Here is a list standard Exceptions available in Python: Standard Exceptions.

- **Assertions:** This would be covered in Assertions in Python

List of Standard Exceptions −

EXCEPTION NAME	DESCRIPTION
Exception	Base class for all exceptions

StopIteration	Raised when the next() method of an iterator does not point to any object.
SystemExit	Raised by the sys.exit() function.
StandardError	Base class for all built-in exceptions except StopIteration and SystemExit.
ArithmeticError	Base class for all errors that occur for numeric calculation.
OverflowError	Raised when a calculation exceeds maximum limit for a numeric type.
FloatingPointError	Raised when a floating point calculation fails.
ZeroDivisionError	Raised when division or modulo by zero takes place for all numeric types.
AssertionError	Raised in case of failure of the Assert statement.
AttributeError	Raised in case of failure of attribute reference or assignment.
EOFError	Raised when there is no input from either the raw_input() or input() function and the end of file is reached.
ImportError	Raised when an import statement fails.

KeyboardInterrupt	Raised when the user interrupts program execution, usually by pressing Ctrl+c.
LookupError	Base class for all lookup errors.
IndexError KeyError	Raised when an index is not found in a sequence. Raised when the specified key is not found in the dictionary.
NameError	Raised when an identifier is not found in the local or global namespace.
UnboundLocalError EnvironmentError	Raised when trying to access a local variable in a function or method but no value has been assigned to it. Base class for all exceptions that occur outside the Python environment.
IOError IOError	Raised when an input/ output operation fails, such as the print statement or the open() function when trying to open a file that does not exist. Raised for operating system-related errors.
SyntaxError IndentationError	Raised when there is an error in Python syntax. Raised when indentation is not specified properly.
SystemError	Raised when the interpreter finds an internal problem, but when this error is encountered the Python interpreter does not exit.

SystemExit	Raised when Python interpreter is quit by using the sys.exit() function. If not handled in the code, causes the interpreter to exit.
TypeError	Raised when an operation or function is attempted that is invalid for the specified data type.
ValueError	Raised when the built-in function for a data type has the valid type of arguments, but the arguments have invalid values specified.
RuntimeError	Raised when a generated error does not fall into any category.
NotImplementedError	Raised when an abstract method that needs to be implemented in an inherited class is not actually implemented.

What is Exception?

An exception is an event, which occurs during the execution of a program that disrupts the normal flow of the program's instructions. In general, when a Python script encounters a situation that it cannot cope with, it raises an exception. An exception is a Python object that represents an error.

When a Python script raises an exception, it must either handle the exception immediately otherwise it terminates and quits.

Handling an exception

If you have some *suspicious* code that may raise an exception, you can defend your program by placing the suspicious code in a **try:** block. After the try: block, include an **except:** statement, followed by a block of code which handles the problem as elegantly as possible.

The Python standard for database interfaces is the Python DB-API. Most Python database interfaces adhere to this standard.

You can choose the right database for your application. Python Database API supports a wide range of database servers such as −

- GadFly

- mSQL
- MySQL
- PostgreSQL
- Microsoft SQL Server 2000
- Informix
- Interbase
- Oracle
- Sybase

The DB API provides a minimal standard for working with databases using Python structures and syntax wherever possible. This API includes the following:

- Importing the API module.
- Acquiring a connection with the database.
- Issuing SQL statements and stored procedures.
- Closing the connection

9.2 SOURCE CODE

MANAGE.PY

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE',
'campus_abnormal_behavior_recognition.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if _name_ == '_main_':
    main()
```

INDEX.HTML

```
 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
{% load static %}
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Home Page</title>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

  <link rel="stylesheet"  type="text/css" href="{% static 'style.css'%} "  />
<link rel="stylesheet"   type="text/css" href="{% static 'coin-slider.css'%}" />

<script type="text/javascript" src="{% static 'cufon-yui.js'%} "></script>
```

```
<script type="text/javascript" src="{% static 'cufon-aller.js'%}"></script>
<script type="text/javascript" src="{% static 'jquery-1.4.2.min.js'%}"></script>
<script type="text/javascript" src="{% static 'script.js'%}"></script>
<script type="text/javascript" src="{% static 'coin-slider.min.js'%}"></script>
<style type="text/css">
<!--
.style5 {
	font-size: 24px;
	color: #FF0000;
}
.style12 {font-weight: bold}
.style13 {font-size: 24px; color: #FF0000; font-weight: bold; }
.style16 {
	color: #FF0000;
	font-weight: bold;
}
-->
</style>
</head>
<body>
<div class="main">
  <div class="header">
    <div class="header_resize">
      <div class="menu_nav">
        <p>&nbsp;</p>
      </div>
      <div class="mainbar">
        <h1 align="center"><a href="index.html"><span class="content style5">Campus
Abnormal Behavior Recognition With Temporal Segment Transformers</span></a></h1>
```

```
        <div class="tab-content tab-space">
            <div class="tab-pane active" id="preview-alerts">

        <link href="https://fonts.googleapis.com/css?family=Open+Sans:300,400,600,700"
rel="stylesheet" />
        <script src="https://kit.fontawesome.com/42d5adcbca.js"
crossorigin="anonymous"></script><link href="https://unpkg.com/soft-ui-design-
system@1.0.1/assets/css/soft-design-system.min.css" rel="stylesheet" /><div class="container
py-5">
  <div class="row">
    <div class="alert alert-primary text-white font-weight-bold" role="alert">
     <p align="center"><span class="active"><span class="style12"><a href="{% url
'index' %}">Home|  </a><a href="{% url 'login' %}">Remote User </a>|<a href="{% url
'serviceproviderlogin' %}"> Service Provider </a></span></span></p>
    </div>
    <div>
</div>




        <img src="{% static 'Banner.jpg'%}" width="1297" height="421" alt="" class="fl" />
</div>
      <div class="clr"></div>
      <div class="slider">


      </div>
      <div class="clr"></div>
    </div>
  </div>
  <div class="content">
    <div class="content_resize">
      <div class="mainbar">
        <div class="article">
          <h2 align="center" class="style13">Action recognition, campus abnormal behavior,
computer vision, motion sequence features, temporal segment transformer.</h2>
          <div class="img">
            <div align="center"><img src="{% static 'img1.jpg'%}" width="630" height="221"
alt="" class="fl" /></div>
          </div>
          <div class="post_content">
            <p align="center" class="style16">The intelligent campus surveillance system is
beneficial to improve safety in school. Abnormal behavior recognition, a field of action
recognition in computer vision, plays an essential role in intelligent surveillance systems.
Computer vision has been actively applied to action recognition systems based on Convolutional
Neural Networks (CNNs). However, capturing sufficient motion sequence features from videos
remains a significant challenge in action recognition. This work explores the challenges of video-
based abnormal behavior recognition on campus. In addition, a novel framework is established
on long-range temporal video structure modeling and a global sparse uniform sampling strategy
that divides a video into three segments of identical durations and uniformly samples each
snippet. The proposed method incorporates a consensus of three temporal segment transformers
(TST) that globally connects patches and computes selfattention with joint spatiotemporal
factorization. The proposed model is developed on the newly created campus abnormal behavior
recognition (CABR50) dataset, which contains 50 human abnormal action classes with an
average of over 700 clips per class. Experiments show that it is feasible to implement abnormal
behavior recognition on campus and that the proposed method is competitive with other peer
video recognition in terms of Top-1 and Top-5 recognition accuracy. The results suggest that
TST-L+ can improve campus abnormal behavior recognition, corresponding to Top-1 and Top-5
accuracy results of 83.57% and 97.16%, respectively..</p>
          </div>

        <div class="tab-content tab-space">
```

```
            <div class="tab-pane active" id="preview-alerts">

          <link href="https://fonts.googleapis.com/css?family=Open+Sans:300,400,600,700"
rel="stylesheet" />
          <script src="https://kit.fontawesome.com/42d5adcbca.js"
crossorigin="anonymous"></script><link href="https://unpkg.com/soft-ui-design-
system@1.0.1/assets/css/soft-design-system.min.css" rel="stylesheet" /><div class="container
py-5">
  <div class="row">
    <div class="alert alert-primary text-white font-weight-bold" role="alert">
     <p align="center"><span class="active"><span class="style12"><a href="{% url
'login' %}">Home|  </a><a href="{% url 'login' %}">Remote User </a>|<a href="{% url
'serviceproviderlogin' %}"> Service Provider </a></span></span></p>
    </div>
    <div>
</div>
          <div class="clr"></div>
        </div>
      </div>
      <div class="sidebar">
        <div class="searchform"></div>
        <div class="clr"></div>
      </div>
      <div class="clr"></div>
    </div>
  </div>
  <div class="fbg"></div>
  <div class="footer"></div>
</div>
<div align=center></div>
</body>
```

```
</html>
```

PREDICT_CAMPUS_BEHAVIOR_TYPE

```
{% extends 'RUser/Header.html' %}
{% block userblock %}
<link rel="icon" href="images/icon.png" type="image/x-icon" />

        <link href="https://fonts.googleapis.com/css?family=Lobster" rel="stylesheet">
                <link href="https://fonts.googleapis.com/css?family=Righteous"
rel="stylesheet">
<link href="https://fonts.googleapis.com/css?family=Fredoka+One" rel="stylesheet">

                <style>
                        body {background-color:#000000;}
                        .container-fluid {padding:50px;}
                        .container{background-color:white;padding:50px;   }
                        #title{font-family: 'Fredoka One', cursive;

                        .text-uppercase{
                        font-family: 'Righteous', cursive;

                        }
                        .tweettext{

   border: 2px solid yellowgreen;
   width: 904px;
   height: 202px;
   overflow: scroll;
   background-color:;
}
                .style1 {
        color: #FF0000;
```

```
            font-weight: bold;
}
.style4 {color: #FFFF00; font-weight: bold; }
.style7 {font-size: 18px}
        .style8 {color: #FFFF00}
        </style>

            <body>
            <div class="container-fluid">
                        <div class="container">

                                    <div class="row">
                                                <div class="col-md-5">

                  <form role="form" method="POST" >
                                                            {% csrf_token %}
                                                            <fieldset>
                                                                        <p class="text-uppercase pull-center
style1">PREDICTION OF CAMPUS BEHAVIOR TYPE!!! </p>
                         <hr>

                                {% csrf_token %}
                                <table width="992" align="center">
                                  <tr>
                                    <td height="44" colspan="4" bgcolor="#FF0000"><div align="center"
class="style4">ENTER DATASETS DETAILS HERE !!! </div></td>
                                  </tr>
                                  <tr>
                                    <td width="289" height="44" bgcolor="#FF0000"><div align="center"
class="style4">Enter SID</div></td>
                                    <td width="200"><input type="text" name="Sid"></td>
                                    <td width="340" bgcolor="#FF0000"><div align="center"
class="style4">Enter Certification_Course</div></td>
                                    <td width="273"><input type="text" name="Certification_Course"></td>
                                  </tr>
                                  <tr>
                                    <td height="44" bgcolor="#FF0000"><div align="center"
class="style4">Enter Gender</div></td>
                                    <td><input type="text" name="Gender"></td>
                                    <td bgcolor="#FF0000"><div align="center" class="style4">Enter
Department</div></td>
                                    <td><input type="text" name="Department"></td>
                                  </tr>
                                  <tr>
                                    <td height="44" bgcolor="#FF0000"><div align="center" class="style4">
                                      <div align="center">Enter Height_CM</div>
                                    </div></td>
                                    <td><input type="text" name="Height_CM"></td>
                                    <td bgcolor="#FF0000"><div align="center" class="style4">Select
Weight_KG</div></td>
                                    <td><input type="text" name="Weight_KG"></td>
                                  </tr>
                                  <tr>
                                    <td height="44" bgcolor="#FF0000"><div align="center" class="style4">
                                      <div align="center">Enter Tenth_Mark</div>
                                    </div></td>
                                    <td><textarea name="Tenth_Mark"></textarea></td>
                                    <td bgcolor="#FF0000"><div align="center" class="style4">Enter
Twelth_Mark</div></td>
                                    <td><input type="text" name="Twelth_Mark"></td>
                                  </tr>
                                  <tr>
```

```
                              <td height="44" bgcolor="#FF0000"><div align="center"
class="style4">Enter college_mark</div></td>
                              <td><input type="text" name="college_mark"></td>
                              <td bgcolor="#FF0000"><div align="center"><strong><span
class="style7 style2 style8">Enter hobbies</span></strong></div></td>
                              <td><input type="text" name="hobbies"></td>
                             </tr>
                             <tr>
                              <td height="44" bgcolor="#FF0000"><div align="center"
class="style4">Enter daily_studing_time</div></td>
                              <td><span class="style4">
                              <input type="text" name="daily_studing_time">
                              </span></td>
                              <td bgcolor="#FF0000"><div align="center" class="style4"><span
class="style2">Enter</span> prefer_to_study_in</div></td>
                              <td><input type="text" name="prefer_to_study_in"></td>
                             </tr>
                             <tr>
                              <td height="44" bgcolor="#FF0000"><div align="center"
class="style4"><span class="style7 style2">Enter</span> degree_willingness </div></td>
                              <td><input type="text" name="degree_willingness"></td>
                              <td bgcolor="#FF0000"><div align="center"><strong><span
class="style8"><span class="style2">Enter</span>
social_media_video</span></strong></div></td>
                              <td><input type="text" name="social_medai_video"></td>
                             </tr>
                             <tr>
                              <td height="44" bgcolor="#FF0000"><div align="center"
class="style4"><span class="style7 style2">Enter</span> Travelling_Time </div></td>
                              <td><input type="text" name="Travelling_Time"></td>
```

```
                                  <td bgcolor="#FF0000"><div align="center"><strong><span
class="style8"><span class="style2">Enter</span> Stress_Level</span></strong></div></td>
                                  <td><input type="text" name="Stress_Level"></td>
                                </tr>
                                <tr>
                                  <td height="44" bgcolor="#FF0000"><div align="center"
class="style4"><span class="style7 style2">Enter</span> Financial_Status</div></td>
                                  <td><input type="text" name="Financial_Status"></td>
                                  <td bgcolor="#FF0000"><div align="center"><strong><span
class="style8"><span class="style2">Enter</span> part_time_job</span></strong></div></td>
                                  <td><input type="text" name="part_time_job"></td>
                                </tr>


                                <tr>
                                  <td height="44" bgcolor="#FFFFFF">&nbsp;</td>
                                  <td><input name="submit" type="submit" class="style1"
value="Predict"></td>
                                  <td>&nbsp;</td>
                                  <td>&nbsp;</td>
                                </tr>
                              </table>




                            </div>


              <form role="form" method="POST" >
                                                          {% csrf_token %}
                                                          <fieldset>


                            <hr>
```

```
                    <div>
                      <table width="844" height="71" border="0" align="center" >
                            <tr><td width="609" bgcolor="#FF0000"><div align="center"><span
class="style8 style6"><strong>PREDICTED CAMPUS BEHAVIOR TYPE </strong></span>
<span class="style4">:: --&gt;</span> </div></td>



                              <td width="225" bgcolor="#FFFFFF" style="color:red; font-size:20px;
font-family:fantasy" ><div align="center"><strong>{{objs}}</strong></div></td>
                            </tr>
                </table>




                    </div>
                                              </fieldset>
                                        </form>

                                              </fieldset>
                                        </form>
                                  </div>

                                  <div class="col-md-2">
                                        <!-------null------>
                                  </div>
            </div>
          </div>
       </div>
{% endblock %}
       <tr>
```

CHAPTER-10

RESULTS/DISCUSSION

10.1 SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

FUNCTIONAL TEST

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
Systems/Procedures	: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

SYSTEM TEST

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

WHITE BOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

UNIT TESTING

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

TEST STRATEGY AND APPROACH

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

ACCEPTANCE TESTING

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

10.1.1 TEST CASES:

Test case1:

Test case for Login form:

FUNCTION:	LOGIN
EXPECTED RESULTS:	Should Validate the user and check his existence in database
ACTUAL RESULTS:	Validate the user and checking the user against the database
LOW PRIORITY	No
HIGH PRIORITY	Yes

Test case2:

Test case for User Registration form:

FUNCTION	REMOTE USER REGISTRATION
EXPECTED RESULTS:	Should check if all the fields are filled by the user and saving the user to database.

ACTUAL RESULTS:	Checking whether all the fields are filled by user or not through validations and saving user.
LOW PRIORITY	No
HIGH PRIORITY	Yes

Test case3:

Test case for Change Password:

When the old password does not match with the new password , then this results in displaying an error message as " OLD PASSWORD DOES NOT MATCH WITH THE NEW PASSWORD".

Test case 4:

Test case for Forget Password:

When a user forgets his password he is asked to enter Login name, ZIP code, Mobile number. If these are matched with the already stored ones then user will get his Original password.

Modu le	**Functio nality**	**Test Case**	**Expected Results**	**Actual Results**	**Res ult**	**Priori ty**
User	Login Usecase	1. Navigate To Www.Sample.Co m 2. Click On Submit Button Without Entering Username and Password	A Validation Should Be As Below "Please Enter Valid Username & Password"	A Validation Has Been Populated As Expected	Pass	High

		1. aNavigate To Www.Sample.Co m 2. Click On Submit Button With Out Filling Password And With Valid Username	A Validation Should Be As Below "Please Enter Valid Password Or Password Field Can Not Be Empty "	A Validation Is Shown As Expected	Pass	High
		1. NNavigate To Www.Sample.Co m 2. Enter Both Username And Password Wrong And Hit Enter	A Validation Shown As Below "The Username Entered Is Wrong"	A Validation Is Shown As Expected	Pass	High
		1. Navigate To Www.Sample.Co m 2. Enter Validate Username And Password And Click On Submit	Validate Username And Password In DataBase And Once If They Correct Then Show The Main Page	Main Page/ Home Page Has Been Displayed	Pass	High

10.2 SCREEN SHOTS

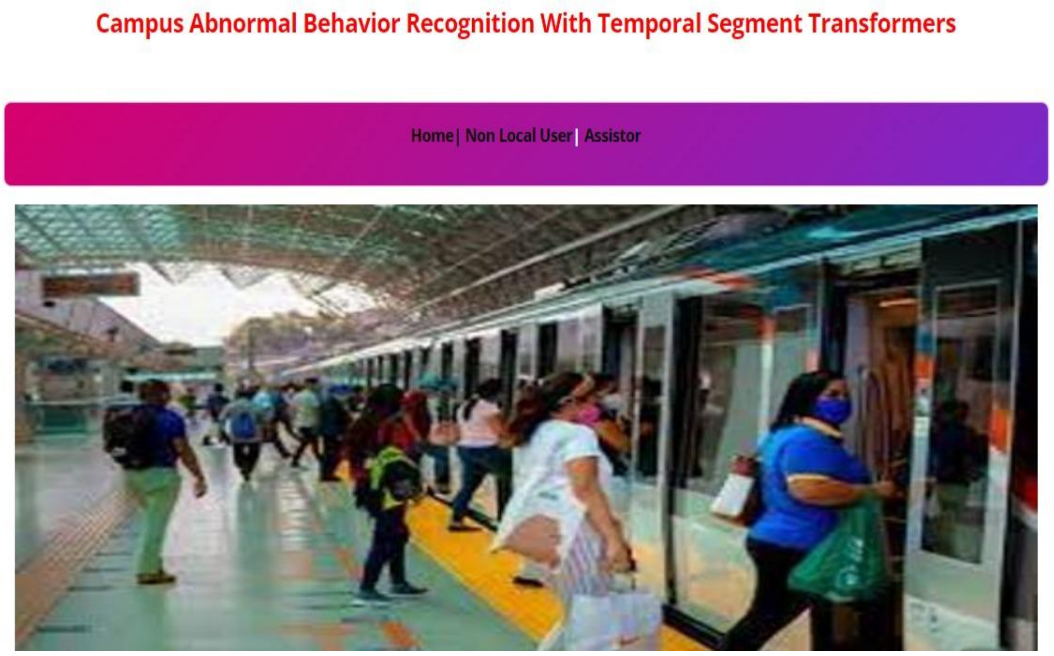

Campus Abnormal Behavior Recognition With Temporal Segment Transformers
Home| Non Local User| Assistor

FIG-1 Home page

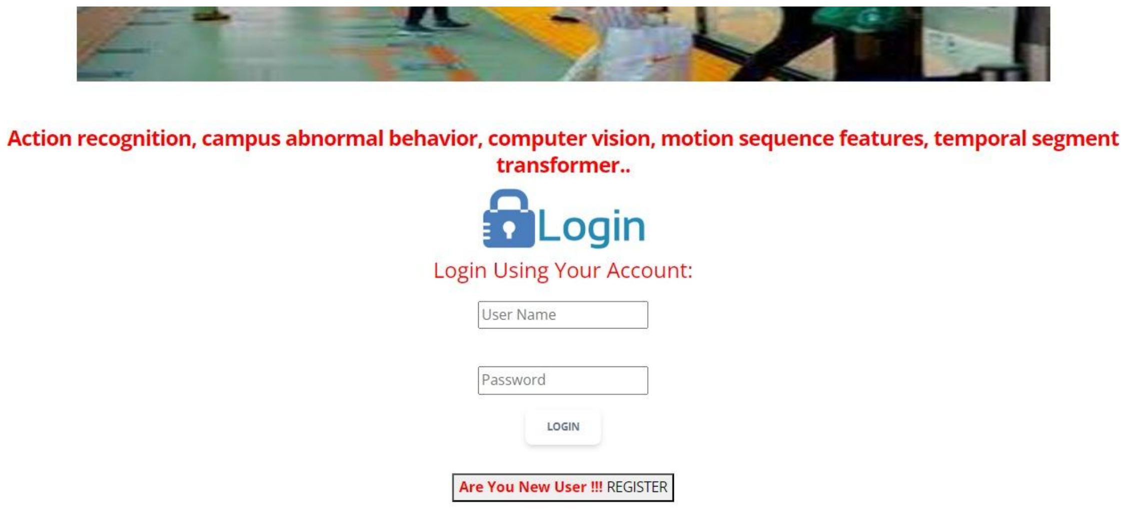
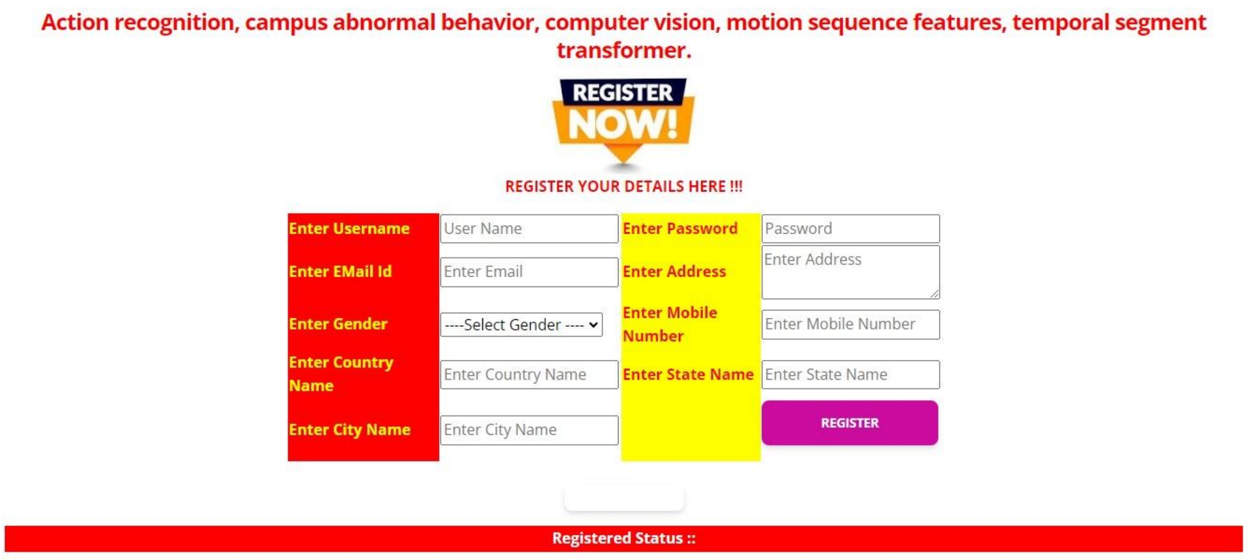

Action recognition, campus abnormal behavior, computer vision, motion sequence features, temporal segment transformer..

Fig-2 Non local user login page login page

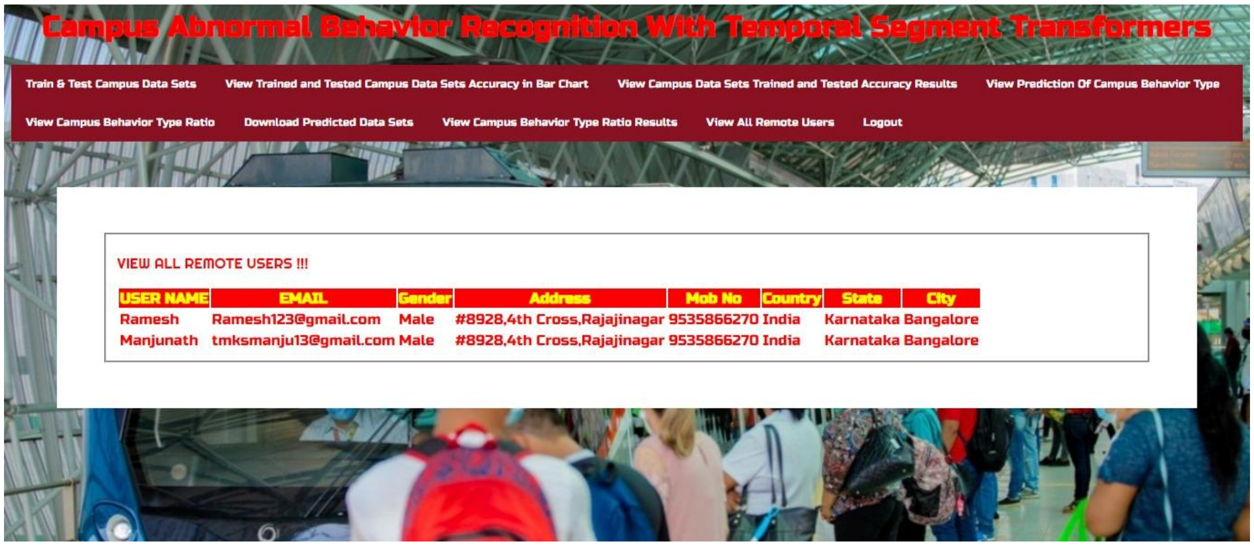

Action recognition, campus abnormal behavior, computer vision, motion sequence features, temporal segment transformer.

REGISTER YOUR DETAILS HERE !!!

Enter Username	User Name	Enter Password	Password
Enter EMail Id	Enter Email	Enter Address	Enter Address
Enter Gender	----Select Gender ----	Enter Mobile Number	Enter Mobile Number
Enter Country Name	Enter Country Name	Enter State Name	Enter State Name
Enter City Name	Enter City Name		REGISTER

Registered Status ::

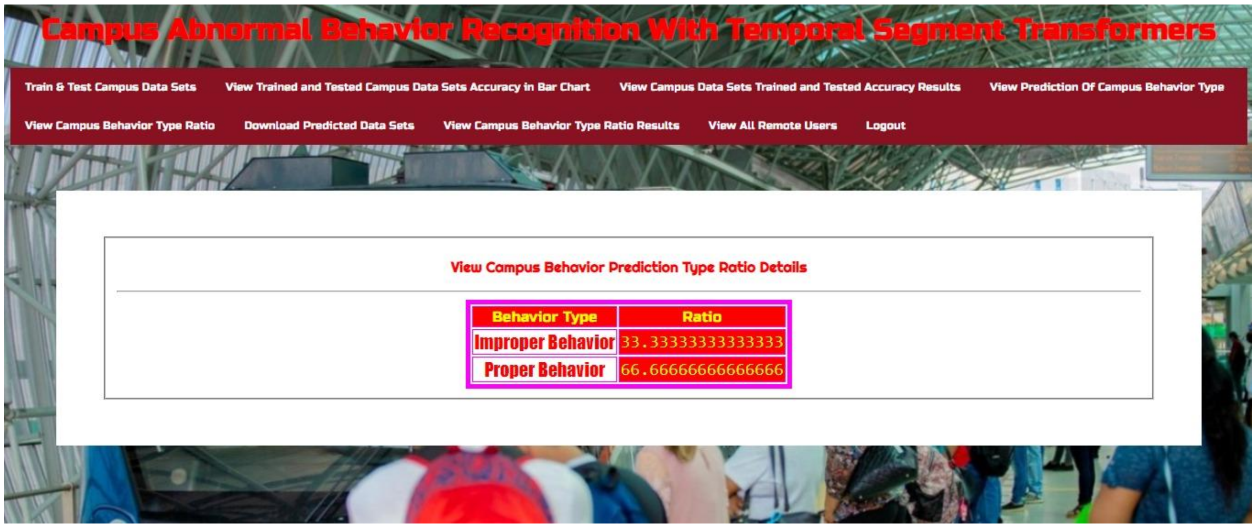
FIG-3 Non local user Register page


Campus Abnormal Behavior Recognition With Temporal Segment Transformers
Train & Test Campus Data Sets
View Trained and Tested Campus Data Sets Accuracy in Bar Chart
View Campus Data Sets Trained and Tested Accuracy Results
View Prediction Of Campus Behavior Type
View Campus Behavior Type Ratio
Download Predicted Data Sets
View Campus Behavior Type Ratio Results
View All Remote Users
Logout

VIEW ALL REMOTE USERS !!!

USER NAME	EMAIL	Gender	Address	Mob No	Country	State	City
Ramesh	Ramesh123@gmail.com	Male	#8928,4th Cross,Rajajinagar	9535866270	India	Karnataka	Bangalore
Manjunath	tmksmanju13@gmail.com	Male	#8928,4th Cross,Rajajinagar	9535866270	India	Karnataka	Bangalore

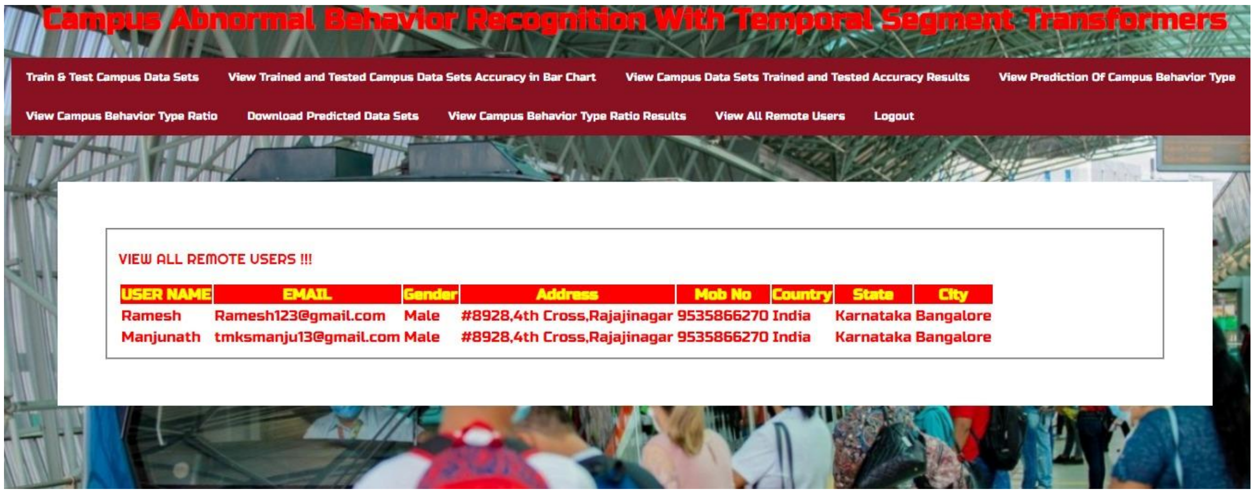
FIG-4 view all remote users


Campus Abnormal Behavior Recognition With Temporal Segment Transformers
Train & Test Campus Data Sets
View Trained and Tested Campus Data Sets Accuracy in Bar Chart
View Campus Data Sets Trained and Tested Accuracy Results
View Prediction Of Campus Behavior Type
View Campus Behavior Type Ratio
Download Predicted Data Sets
View Campus Behavior Type Ratio Results
View All Remote Users
Logout

View Campus Behavior Prediction Type Ratio Details

Behavior Type	Ratio
Improper Behavior	33.33333333333333
Proper Behavior	66.66666666666666

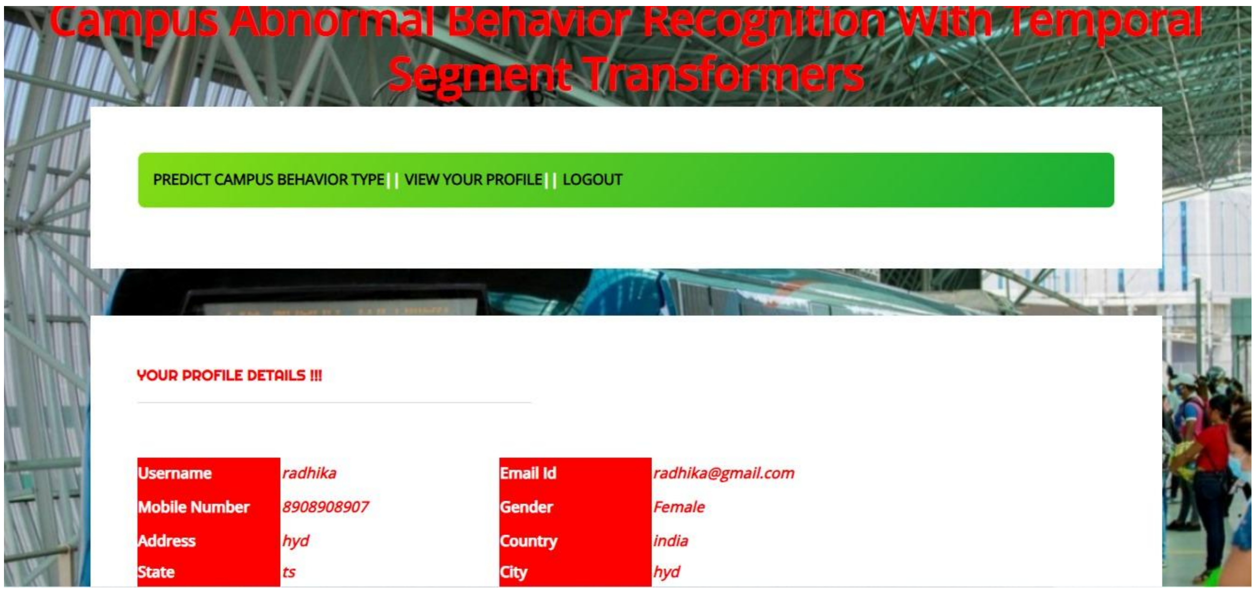
FIG-5 view campus behavior prediction type ratio details


Campus Abnormal Behavior Recognition With Temporal Segment Transformers
Train & Test Campus Data Sets
View Trained and Tested Campus Data Sets Accuracy in Bar Chart
View Campus Data Sets Trained and Tested Accuracy Results
View Prediction Of Campus Behavior Type
View Campus Behavior Type Ratio
Download Predicted Data Sets
View Campus Behavior Type Ratio Results
View All Remote Users
Logout

VIEW ALL REMOTE USERS !!!

USER NAME	EMAIL	Gender	Address	Mob No	Country	State	City
Ramesh	Ramesh123@gmail.com	Male	#8928,4th Cross,Rajajinagar	9535866270	India	Karnataka	Bangalore
Manjunath	tmksmanju13@gmail.com	Male	#8928,4th Cross,Rajajinagar	9535866270	India	Karnataka	Bangalore

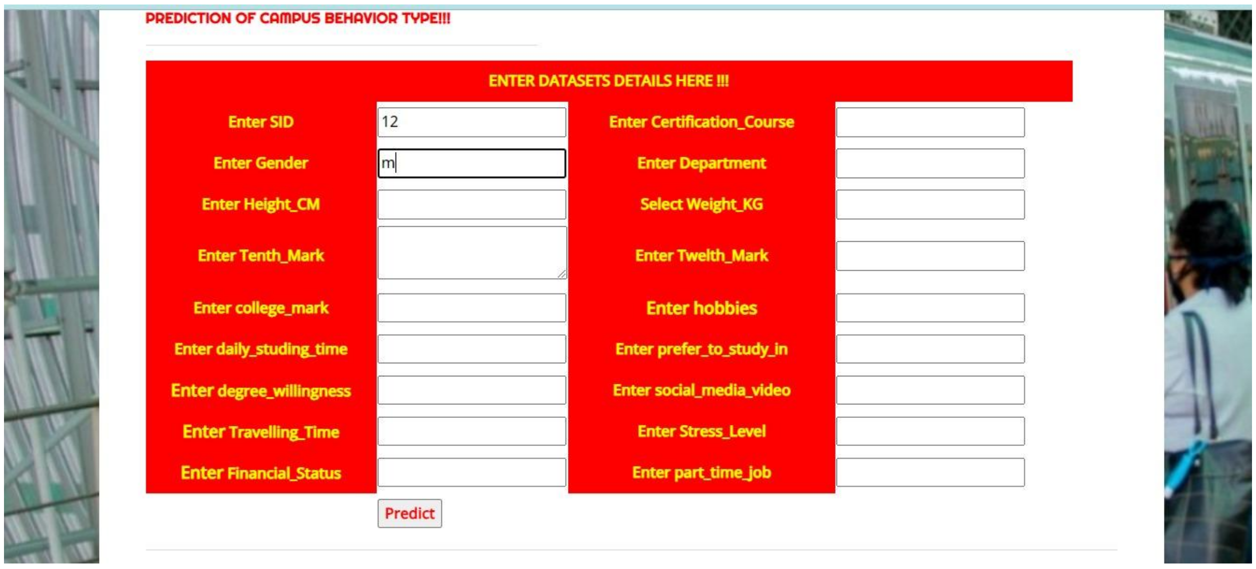
Fig-6 view all remote uesers


Campus Abnormal Behavior Recognition With Temporal Segment Transformers
PREDICT CAMPUS BEHAVIOR TYPE | | VIEW YOUR PROFILE | | LOGOUT

YOUR PROFILE DETAILS !!!

Username	radhika	Email Id	radhika@gmail.com
Mobile Number	8908908907	Gender	Female
Address	hyd	Country	india
State	ts	City	hyd

FIG-7 User page


PREDICTION OF CAMPUS BEHAVIOR TYPE!!!

ENTER DATASETS DETAILS HERE !!!

Enter SID	12	Enter Certification_Course	
Enter Gender	m	Enter Department	
Enter Height_CM		Select Weight_KG	
Enter Tenth_Mark		Enter Twelth_Mark	
Enter college_mark		Enter hobbies	
Enter daily_studing_time		Enter prefer_to_study_in	
Enter degree_willingness		Enter social_media_video	
Enter Travelling_Time		Enter Stress_Level	
Enter Financial_Status		Enter part_time_job	

Fig-8 Prediction of campus behaviou r type

CHAPTER-11
CONCLUSION

11.1 CONCLUSION

Abnormal behavior recognition plays a significant role in intelligent campus surveillance systems. In this study, a CABR50 dataset was created, and a framework named temporal segment transformers was proposed to address the problem of identifying abnormal behavior on campus. Specifically, TST divides the input video into three segments of equal duration from the original video and obtains snippets uniformly sampled from its segment. These snippets are used as the inputs for the backbone network. Each snippet produces its initial prediction of the class, followed by a consensus function between the snippets exported as the final prediction, enabling dynamic global video modeling. Extensive experiments are carried out on the three split CABR50 datasets to verify the classification accuracy: TSN, Slow fast, Swin -B, and TST methods. In addition, the superiority of the proposed method in classifying abnormal behaviors on campus is verified in terms of the analysis result. To explore the model's generalization performance, we experimented with it on the UCF-101 dataset and achieved promising results. In summary, this work demonstrates the feasibility of using abnormal campus behavior recognition. In addition, our proposed TST can effectively model long-range behavior and achieve competitive results on CABR50. However, the model should perform better on the abnormal campus behavior categories of coughing, debating, and yelling that belong to hard data. In the future, we will try to combine multimodal approaches , such as extracting audio features from videos, to assist in classifying hard data for abnormal campus behavior. In addition, we can take an efficient approach to reduce the complexity of the model and overcome the problem of imbalance in the corresponding category data, such as GAN, which generates new training data for unbalanced categories

11.2 FUTURE SCOPE:

The future scope of a project focused on campus abnormal behavior recognition using Temporal Segment Transformers (TSTs) could involve several avenues of exploration and development

Fine-tuning and Optimization,Multimodal Integration.By pursuing these avenues of research and development, a project on campus abnormal behavior recognition with Temporal Segment Transformers can contribute to enhancing campus safety, security, and well-being while addressing privacy concerns and ethical considerations.

CHAPTER-12

REFERENCES

[1] Q. Hao and L.Qin, ‘‘The design of intelligent transportation video processing system in big data environment,’’ *IEEE Access*, vol. 8,pp. 13769–13780, 2020.

[2] K. Muhammad, J. Ahmad, Z. Lv, P. Bellavista, P. Yang, and S. W. Baik,‘‘Efficient deep CNN-based fire detection and localization in video surveillanceapplications,’’ *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 49, no. 7,pp.
1419–1434, Jul. 2019.

[3] L. Simoni, A. Scarton, C. Macchi, F. Gori, G. Pasquini, and S. Pogliaghi,
‘‘Quantitative and qualitative running gait analysis through an innovative
video-based approach,’’ *Sensors*, vol. 21, no. 9, p. 2977, Apr. 2021.

[4] M. Shorfuzzaman, M. S. Hossain, and M. F. Alhamid, ‘‘Towards the
sustainable development of smart cities through mass video surveillance:
A response to the COVID-19 pandemic,’’ *Sustain. Cities Soc.*, vol. 64,
Jan. 2021, Art. no. 102582.

[5] A. B. Mabrouk and E. Zagrouba, ‘‘Abnormal behavior recognition for
intelligent video surveillance systems: A review,’’ *Expert Syst. Appl.*,vol. 91,
pp. 480–491, Jan. 2018.

[6] A. B. Tanfous, H. Drira, and B. B. Amor, ‘‘Sparse coding of shape trajectories
for facial expression and action recognition,’’ *IEEE Trans. PatternAnal. Mach. Intell.*, vol. 42, no. 10, pp. 2594–2607, Oct. 2020.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ‘‘ImageNet classification
with deep convolutional neural networks,’’ in *Proc. NIPS*, Dec. 2012,pp.

1097–1105.

[8] C. Szegedy,W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, ‘‘Going deeper with convolutions,’’in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[9] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, ‘‘Rethinking the inception architecture for computer vision,’’ in *Proc. IEEE Conf.Comput Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.

[10] K. He, X. Zhang, S. Ren, and J. Sun, ‘‘Deep residual learning for image recognition,’’ in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[11] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, H. Lin, Z. Zhang, Y. Sun, T. He, J. Mueller, R. Manmatha, M. Li, and A. Smola, ‘‘ResNeSt: Split-attention networks,’’ in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.Workshops (CVPRW)*, Jun. 2022, pp. 2735–2745.

[12] K. Simonyan and A. Zisserman, ‘‘Two-stream convolutional networks for action recognition in videos,’’ in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, Dec. 2014, pp. 1–9.

[13] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. van Gool, ‘‘Temporal segment networks: Towards good practices for deep action recognition,’’ in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland:Springer, Oct. 2016, pp. 20–36.

[14] Z. Lan, Y. Zhu, A. G. Hauptmann, and S. Newsam, ‘‘Deep local video feature for action recognition,’’ in *Proc. IEEE Conf. Comput. Vis. Pattern*

Recognit. Workshops (CVPRW), Jul. 2017, pp. 1219–1225.

[15] A. Diba, V. Sharma, and L. Van Gool, ‘‘Deep temporal linear encoding networks,’’ in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1541–1550.

[16] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, ‘‘Learning spatiotemporal features with 3D convolutional networks,’’ in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 4489–4497.

[17] J. Carreira and A. Zisserman, ‘‘Quo vadis, action recognition? A new model and the kinetics dataset,’’ in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6299–6308.

[18] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, ‘‘A closer look at spatiotemporal convolutions for action recognition,’’ in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6450–6459.
[19] C. Feichtenhofer, ‘‘X3D: Expanding architectures for efficient video recognition,’’ in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 200–210.

[20] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy, ‘‘Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification,’’ in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 305–321.

[21] M. Zolfaghari, K. Singh, and T. Brox, ‘‘ECO: Efficient convolutional network for online video understanding,’’ in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 713–730.

[22] C. Feichtenhofer, H. Fan, J. Malik, and K. He, ‘‘SlowFast networks for

video recognition,'' in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6202–6211.

[23] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, ''An image is worth 16 × 16 words: Transformers for image recognition at scale,'' 2020, *arXiv:2010.11929*.

[24] M. Naseer, K. Ranasinghe, S. Khan, M. Hayat, F. S. Khan, and M.-H. Yang, ''Intriguing properties of vision transformers,'' in *Proc. NeurIPS*, Dec. 2021, pp. 23296–23308.

[25] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, ''Swin transformer: Hierarchical vision transformer using shifted windows,'' in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 10012–10022.