

**A**

**INTERNSHIP REPORT**

**Under Subject of**

**Data Science and Analytics**

**SUMMER INTERNSHIP - 1(ITIS1)**

**B.Tech. SEMESTER-V**

**Submitted by**

**Patel Manmeet Miteshkumar (21IT456)**



**Information Technology Department**

**Birla Vishvakarma Mahavidyalaya Engineering College**

**(An Autonomous Institution)**

**AY: 2022-23, Semester II**



**Birla Vishvakarma Mahavidyalaya Engineering College**

**(An Autonomous Institution)**

**Information Technology Department**

**AY: 2022-23, Semester II**

## **CERTIFICATE**

This is to certify that Patel Manmeet Miteshkumar with ID Number 21IT456 has been successfully completed summer Internship -1 (ITIS1) at SmartSense Consulting Solutions Pvt. Ltd. under my guidance during the academic year 2022-23, Semester II.

**Date: 19/10/23**  
**Prof. Kanu Patel**

IT Department  
BVM

## Table of contents

<b>Sr no.</b>	<b>Topic</b>	<b>Page no.</b>
1.	Completion letter	4
2.	Acknowledgment	5
3.	Company profile	6
4.	Executive summary	7
5.	Daily tasks	8
6.	Chapter 1: Introduction	25
7.	Chapter 2: Analysis, Design Methodology, and Implementation Strategy	27
8.	Chapter 3: Implementation and testing	29
9.	Chapter 4: Conclusion	33
10.	References	34

## Completion letter



## Acknowledgment

I wish to express my profound gratitude to the entire **SmartSense** team, specifically, **Mr. Mayur J. Pabari**, **Mr. Krupal Purohit**, **Mr. Sudip Das**, and **Mr. Suraj Zala**, for their invaluable guidance and unwavering support throughout the duration of my internship. Each of these experts possesses an impressive depth of knowledge in their respective roles, and their expertise played an instrumental role in facilitating my rapid progress in the expansive and forward-looking field of **Machine Learning** and **Artificial Intelligence**.

I would also like to extend my appreciation to **Birla Vishvakarma Mahavidyalaya**, our esteemed Principal, **Prof. (Dr.) Indrajit Patel**, the Head of the IT Department, **Mr. Keyur Brahmbhatt**, and the Coordinators, **Ms. Zankhana Shah** and **Mr. Dharmesh Patel**. It is through their gracious provision of this exceptional opportunity that I was able to gain practical, hands-on experience and receive invaluable lessons from industry luminaries.

## Company Profile

**smartSense Consulting Solutions Pvt Ltd:**

**Website:** <https://www.smartsensesolutions.com/>

### **Aim:**

Our aim is to remain one of the most innovative organizations around the world that provides concrete solutions to the clients in various domains around the globe. And we believe that it always makes smart sense if the customer wins. That's why our product solutions, ranging from Education domain, well-developed organisations, to smartHome, have been customized to extend the benefits of our customers.

### **Mission:**

Our mission is to continue implementing smart development Solutions for the customers and providing them outstanding IT services. We always keep the business perspective along with the technical knowledge in order to help our clients develop robust platform technically as well as practically. All in all “**Customer's Win is our Mission**”.

### **Expertise:**

Blockchain | Data Science | Web development | Mobile app development | Internet of things | Database and server management | Quality analysis

## **Executive summary**

During my two weeks of work as a summer intern, I learned a lot about data science and analytics. It was mostly concentrated on the machine learning component, which was entirely new to me. I discussed machine learning fundamentals including supervised and unsupervised learning. Additionally learnt some fundamentals of elastic search, neural networks, deep learning and API's. At the end of it, I started working on my first machine learning project under experts' guidance. In that first of all I executed simple feed forward neural network from scratch for xor functionality. Then tried to create cricket match highlights generating model using CNN+RNN. Eventually created a model for calculating Calculating the Screen Time of Actors in any Video with Deep Learning

# Daily Tasks

## Day-1:

### Introduction to ML and Python basics

We learnt some of the fundamentals of ML and were introduced to its huge scope.

#### 1. Supervised learning

Supervised Learning is a type of machine learning where the algorithm is trained on a labeled dataset, meaning that the input data consists of both the input features and the corresponding correct output. The goal is for the algorithm to learn a mapping function that can accurately predict the output for new, unseen data.

#### 2. Unsupervised learning

Unsupervised Learning is a type of machine learning where the algorithm is given input data without explicit instructions on what to do with it. The system tries to learn the patterns and structure from the data without any labeled outputs.

#### 3. Reinforcement learning

Reinforcement Learning is a type of machine learning where an agent learns to make decisions by interacting with an environment. The agent receives feedback in the form of rewards or punishments, allowing it to learn optimal strategies over time.

#### 4) Semisupervised learning:

Reinforcement Learning is a type of machine learning where an agent learns to make decisions by interacting with an environment. The agent receives feedback in the form of rewards or punishments, allowing it to learn optimal strategies over time.

Then, we did some basic Python programs and understood the versatility of Python that it provides.

## Day-2:



### **Difference between AI/ML/DL:**

AI: Engineering of making intelligent programs and machines

ML: Ability to learn without being explicitly programmed

DL: Learning based on deep neural network

- **Data collection:**

**Definition:** Data collection involves gathering relevant data from various sources to create a dataset that can be used for training and evaluating machine learning models.

**Challenges:**

**Data Availability:** Sometimes, obtaining sufficient and relevant data can be a challenge.

**Data Quality:** Ensuring that the collected data is accurate and representative of the problem is crucial.

- **Data cleaning:**

**Definition:** Data cleaning involves identifying and correcting errors or inconsistencies in the dataset.

**Challenges:**

**Identifying Errors:** Detecting errors or inconsistencies in the dataset can be challenging.

**Balancing Classes:** Addressing class imbalances, especially in datasets where certain classes are underrepresented.

- **Data preprocessing:**

**Definition:** Data preprocessing involves transforming raw data into a format suitable for machine learning. It includes cleaning, formatting, and organizing the data.

**Challenges:**

**Choosing Imputation Methods:** Selecting appropriate methods for imputing missing values.

**Scaling Challenges:** Deciding whether to use normalization or standardization based on the characteristics of the data.

**Feature Engineering Complexity:** Creating meaningful features that enhance model performance.

## Day-3

### **Classification:**

#### **Definition:**

Classification is a supervised learning task where the goal is to predict the categorical class labels of new instances based on past observations. The output is a discrete label or category.

#### **Key Characteristics:**

#### **Output:**

Categorical labels or classes (e.g., spam or not spam, digit recognition, sentiment analysis - positive/negative/neutral).

#### **Problem Type:**

Used for problems where the target variable is a category or class.

#### **Algorithms:**

Common algorithms include Decision Trees, Random Forests, Support Vector Machines, Naive Bayes, and Neural Networks.

#### **Evaluation Metrics:**

Accuracy, precision, recall, F1-score, and confusion matrix are commonly used metrics for evaluating classification models.

Example:

**Problem:** Email spam detection.

**Input Features:** Email content, sender, subject, etc.

**Output:** Spam or Not Spam.

**Workflow:****1) Data Collection:**

Gather labeled data with examples of each class.

**2) Data Preprocessing:**

Handle missing values, encode categorical variables, and split data into training and testing sets.

**3) Model Training:**

Use a classification algorithm to train the model on the training data.

**4) Model Evaluation:**

Evaluate the model's performance on the testing data using appropriate metrics.

**5) Prediction:**

Use the trained model to predict the class labels of new, unseen instances.

**Regression:****Definition:**

Regression is a supervised learning task where the goal is to predict a continuous numerical value based on input features. The output is a real-valued quantity.

**Key Characteristics:****Output:**

Continuous numerical values (e.g., house prices, temperature, stock prices).

**Problem Type:**

Used for problems where the target variable is a numeric value.

**Algorithms:**

Linear Regression, Decision Trees, Random Forests, Support Vector Regression, and Neural Networks are common regression algorithms.

### **Evaluation Metrics:**

Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared are commonly used metrics for evaluating regression models.

Example:

**Problem:** Predicting house prices.

**Input Features:** Size of the house, number of bedrooms, location, etc.

**Output:** Continuous numerical value representing the price.

### **Workflow:**

#### **1) Data Collection:**

Gather labeled data with numerical target values.

#### **2) Data Preprocessing:**

Handle missing values, encode categorical variables, and split data into training and testing sets.

#### **3) Model Training:**

Use a regression algorithm to train the model on the training data.

#### **4) Model Evaluation:**

Evaluate the model's performance on the testing data using appropriate regression metrics.

#### **5) Prediction:**

Use the trained model to predict the numerical values of new, unseen instances.

### **Key Differences:**

#### **Classification:**

**Output:** Categorical labels.

**Goal:** Group instances into predefined categories.

**Regression:**

**Output:** Continuous numerical values.

**Goal:** Predict a specific quantity.

In summary, classification and regression are both types of supervised learning, but they differ in terms of the nature of the output they predict—categories for classification and continuous values for regression. The choice between them depends on the nature of the problem and the type of predictions needed.

## Day-4,5

### Linear regression

**Definition:**

Linear Regression is a supervised machine learning algorithm used for predicting a continuous outcome variable (dependent variable) based on one or more predictor variables (independent variables). It assumes a linear relationship between the features and the target variable.

**Key Characteristics:**

**Equation:**

The linear regression equation is of the form  $y=mx+b$ , where  $y$  is the predicted output,  $x$  is the input feature,  $m$  is the slope, and  $b$  is the intercept.

**Objective:**

Minimize the sum of squared differences between the actual and predicted values.

**Use Cases:**

Predicting house prices, stock prices, or any continuous variable.

**Algorithm:**

Ordinary Least Squares (OLS) are often used to find the optimal parameters (slope and intercept).

## **Logistic regression**

**Definition:**

Logistic Regression is a classification algorithm used to predict the probability of an instance belonging to a particular class. Despite its name, it's used for binary (two-class) and multi-class classification problems.

**Key Characteristics:**

**Equation:**

The logistic regression equation uses the logistic function (sigmoid function) to map linear combinations of features to values between 0 and 1.

**Objective:**

Minimize the log-likelihood function, which measures the likelihood of the observed class labels given the features.

**Use Cases:**

Spam detection, disease diagnosis, sentiment analysis.

**Algorithm:**

Optimized using techniques like Maximum Likelihood Estimation (MLE).

## **Stochastic Gradient Descent regression**

**Definition:**

Stochastic Gradient Descent (SGD) is an optimization algorithm used for training machine learning models, including linear and logistic regression. It updates the model parameters with each training example rather than the entire dataset.

**Key Characteristics:**

**Update Rule:**

The parameters are updated with each training example using the gradient of the loss function with respect to the parameters.

**Batch Size:**

It often works with small batches of data, introducing randomness into the optimization process.

**Convergence:**

Converges faster than batch gradient descent but may have more oscillations.

**Use Cases:**

Suitable for large datasets where processing the entire dataset in one go is computationally expensive.

**Key Differences:**

**1) Linear Regression:**

**Output:** Continuous values.

**Objective:** Minimize the sum of squared differences.

**2) Logistic Regression:**

**Output:** Probabilities between 0 and 1.

**Objective:** Minimize the negative log-likelihood.

**3) Stochastic Gradient Descent:**

**Optimization:** Updates model parameters with each training example.

**Use Cases:** Particularly useful for large datasets.

In summary, Linear Regression is for predicting continuous values, Logistic Regression is for classification tasks, and Stochastic Gradient Descent is an optimization algorithm that can be used with both regression and classification models.

## Day-6

We learned about these two libraries and carried out a number of fundamental operations on them.

### **NumPy:**

NumPy (Numerical Python) is a powerful numerical library in Python that provides support for large, multi-dimensional arrays and matrices, along with mathematical functions to operate on these arrays.

### **Pandas:**

Pandas is an open-source data manipulation and analysis library for Python. It provides data structures like Series and DataFrame for efficient manipulation and analysis of structured data.

Further toying and exploring with NumPy and Pandas was carried out, with hands on.

## Day-7,8

## Decision Tree

### **Definition:**

A Decision Tree is a supervised machine learning algorithm that makes decisions by recursively partitioning the data into subsets based on the values of input features. It is a tree-like model where each node represents a decision or a test on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label.

### **Key Characteristics:**

#### **1) Nodes:**

**Decision nodes:** Represent a decision or a test on an attribute.

**Branches:** Represent the outcome of a test.

**Leaf nodes:** Represent the class label.

#### **2) Splitting Criteria:**



Decision Trees use various criteria (e.g., Gini impurity, entropy, or mean squared error) to decide the best feature and split at each node.

### **3) Recursive Splitting:**

The process of splitting continues recursively until a stopping criterion is met, such as a maximum depth or a minimum number of samples per leaf.

### **4) Classification and Regression:**

Decision Trees can be used for both classification and regression tasks.

### **Workflow:**

#### **1) Root Node:**

Select the best feature to split the data at the root node.

#### **2) Splitting:**

Recursively split the data into subsets based on feature values.

#### **3) Stopping Criteria:**

Stop splitting when a stopping criterion is met.

#### **4) Leaf Nodes:**

Assign class labels to leaf nodes.

#### **5) Prediction:**

For a new input, traverse the tree to a leaf node and predict the corresponding class.

### **Advantages:**

- **Interpretability:**

Decision Trees are easy to interpret and visualize, making them useful for understanding decision-making processes.

- **No Feature Scaling:**

Decision Trees do not require feature scaling, making them suitable for datasets with different scales.

- **Handling Nonlinear Relationships:**

Decision Trees can model nonlinear relationships in the data.

**Limitations:**

- **Overfitting:**

Decision Trees are prone to overfitting, capturing noise in the training data.

- **Instability:**

Small changes in the data can lead to different tree structures.

**Random Forest:**

**Definition:**

Random Forest is an ensemble learning method that constructs a multitude of decision trees during training and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. It builds multiple Decision Trees and merges them together to get a more accurate and stable prediction.

**Key Characteristics:**

**Bagging:**

Random Forest uses a technique called bagging (Bootstrap Aggregating) to train multiple Decision Trees.

**Random Subsets:**

Each tree is trained on a random subset of the training data with replacement (bootstrap samples).

**Random Feature Subset:**

At each node of the tree, a random subset of features is considered for splitting.

### **Voting or Averaging:**

For classification, the mode (most frequent class) of the predictions is taken. For regression, the average of the predictions is considered.

### **Workflow:**

#### **1) Bootstrapped Samples:**

Create multiple bootstrapped samples from the training data.

#### **2) Decision Tree Training:**

Train a Decision Tree on each bootstrapped sample using a random subset of features at each node.

#### **3) Voting or Averaging:**

Combine the predictions of individual trees through voting (classification) or averaging (regression).

#### **4) Prediction:**

For a new input, obtain predictions from each tree and combine them.

### **Advantages:**

#### **1) Reduced Overfitting:**

Random Forest mitigates overfitting by combining multiple trees.

#### **2) Improved Generalization:**

The ensemble nature of Random Forest improves generalization to new data.

#### **3) Feature Importance:**

Random Forest provides a measure of feature importance based on the contribution of features to the ensemble.

### **Limitations:**

#### **1) Complexity:**

Random Forest models can be computationally intensive and may require more resources.

## **2) Less Interpretability:**

While Random Forest provides feature importance, interpreting the entire ensemble can be challenging.

In summary, Decision Trees are single trees that make decisions based on feature values, while Random Forest is an ensemble of multiple trees that reduces overfitting and improves generalization by combining the predictions of individual trees. Random Forest is a powerful and versatile algorithm used in various machine-learning applications.

## **Day-9**

### **Discrete Input, Discrete Output:**

Refers to situations where both the input and output are categorical and distinct. In the context of machine learning, this often involves making predictions for categorical variables, where the output is a discrete set of categories.

### **Random Forest:**

A sophisticated ensemble learning method that utilizes multiple decision trees. Each tree independently makes predictions, and the final output is determined through voting (for classification) or averaging (for regression). This ensemble approach enhances model robustness and generalization.

### **Decision Tree:**

An intuitive, hierarchical model that makes decisions based on conditions at each node. It operates like a flowchart, where each internal node represents a test on an attribute, each branch represents the outcome of the test, and each leaf node holds a class label.

### **Entropy and Decision-Making:**

Entropy is a measure of impurity in decision tree nodes. The concept is discussed in relation to decision-making, highlighting that achieving a balance between yes and no responses is crucial

for effective decision trees. This is especially pertinent when there is more entropy, indicating increased uncertainty.

### **Bagging and Boosting:**

#### **Bagging:**

A technique that aims to reduce the variance of a model by combining predictions from multiple models trained on different subsets of the dataset. Commonly used with high-variance models.

#### **Boosting:**

A method that combines multiple weak learners (models with high bias) to create a strong learner. By iteratively correcting errors, boosting reduces bias and enhances overall model performance.

### **Recurrent Neural Networks (RNNs):**

#### **RNNs:**

Specialized neural networks are designed for processing sequences of data, making them well-suited for tasks involving time series data or natural language processing. The discussion introduces concepts like recurrent neurons, layers, and the challenges of short-term memory.

#### **Short-Term Memory Issues:**

RNNs face difficulties in retaining information over long sequences due to short-term memory limitations. Solutions such as Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM) architectures are introduced to address these challenges.

### **GPT Training Process:**

#### **GPT Training Phases:**

Outlines the multi-phase training approach for models like GPT. Initial language learning, the creation of conversational datasets, and reinforcement learning using human feedback are key steps.

#### **Reward Function and RLHF:**

Reinforcement Learning from Human Feedback (RLHF) involves training a reward function for reinforcement learning. The model is fine-tuned for question-answering using ELI5-style questions, and the reward function is crucial for aligning the model with human expectations.

### **Image Captioning:**

#### **Image Captioning and BLEU-4:**

Delves into the realm of image captioning, where models generate textual descriptions for images. The BLEU-4 metric, measuring the similarity between predicted and training captions through 4-grams, is introduced. Various datasets like Flickr8k and Flickr30k are briefly discussed.

### **Hyperparameters:**

#### **Dropout and Learning Rate Scheduler:**

Dropout is explained as a regularization technique to prevent overfitting by randomly dropping out neurons during training. The learning rate scheduler adjusts the learning rate dynamically during training, impacting the convergence and performance of the model.

### **Checkpoints:**

Highlights the importance of checkpoints in training. Checkpoints allow the model's state to be saved periodically, providing the ability to resume training or evaluate model performance at different stages.

### **DALL-E by OpenAI:**

Briefly introduces DALL-E, an OpenAI model capable of generating images from textual descriptions. The model showcases the expanding capabilities of AI in creative tasks beyond traditional machine learning applications.

By providing a more detailed explanation, we aim to enhance the understanding of each concept within the given data.

## **Day-10**

### **Elastic search**

Elasticsearch is a **powerful tool for managing and searching through large sets of documents efficiently**. Suppose you have a collection of documents, and someone wants to ask questions related to that content. In such a scenario, the first step is to index those documents, making them searchable through Elasticsearch.

Notably, companies like **Jaguar, Land Rover, Pfizer, Delhivery, and Engadget** leverage Elasticsearch for various purposes.

When running applications, it's common to have three different environments: **development (dev), quality assurance (QA), and production**. Each environment generates different types of logs, including error, exception, debug, and info logs. Kibana, a user interface, operates on port 5601 and serves as a visualization tool for Elasticsearch logs.

In the production environment, it's advisable to change the default port number for added security.

One essential use case is to index documents and enable AI-driven search capabilities. This involves implementing a retriever and a reader. The retriever is responsible for finding relevant documents for a given functionality. The process involves word embedding, where numeric values are assigned to words and placed in a vector space. The default dimension for word embedding is 768, but recent models support up to 2048 dimensions. Transformers play a crucial role in this process, enabling parallelization.

For the retriever, there are techniques like one-hot encoding and normal encoding.

The reader, on the other hand, is tasked with finding answers to content and queries. It employs an encoder and decoder model to accomplish this task effectively.

In summary, **Elasticsearch is employed to index documents and facilitate efficient and intelligent searching through the implementation of retriever and reader functionalities**. It's a valuable tool used by various companies for tasks ranging from managing logs in different environments to enabling sophisticated search capabilities through AI-driven processes.

## Day-11

### Project selection and discussion:

On this day, after a pleasant conversation about the projects, we were instructed to come up with original concepts and consider potential ML projects that we might carry out either solo or in couples. I gave it some serious thought as well, and in the end, one of my pals came up with a fantastic project concept that closely matched our areas of interest. In essence, it dealt with "Match highlights generation with event detection in cricket" with regards to whether the delivery resulted in a dot ball, boundary, wicket, or runs. For analysis and decision-making regarding one delivery, a collection of photos was required. We had carefully read a research paper or two regarding this, explaining different approaches. Its dataset was bit of a trouble as there was no pre-manufactured one, it need to be scrapped turning out to be tedious process. So this project was shelved for some other time and pondering for next was brewing.

## **Day-12**

### **Project planning and implementation:**

After shelving "Match highlights generation with event detection in cricket" project. Another project "Calculating the Screen Time of Actors in any Video with Deep Learning" was taken into hand. Under the guidance of industry experts planning and implementation of the project was accomplished. This project discription is mentioned further. This implementation walked us through how real industrial projects are approached.



# Chapter 1: Introduction

## 1.1 Aim of the Project

To develop a deep learning-based solution for analyzing videos from the popular 'Tom and Jerry' cartoon series and calculating the screen time of both 'Tom' and 'Jerry' in any given video. This project addresses the challenge of quantifying the amount of time these characters appear on the screen, providing valuable insights into their presence within the video content. The goal is to create a tool that can automate the screen time calculation process, which is of significant importance for actors, as it directly impacts their compensation.

## 1.2 Project Scope

This project's significance lies in its potential applications for content creators, media companies, and actors who rely on accurate screen time measurements. By accurately quantifying the time each character appears on the screen, the project could provide valuable insights for the entertainment industry, potentially leading to better compensation models for actors based on their actual screen time. It has potential in advertisement revenue management, like in sports watching.

## 1.3 Project Objective

### 1.3.1 Develop an Accurate Screen Time Detection Model:

The primary objective is to create a robust deep learning model that can accurately detect and track the appearances of characters, such as Tom and Jerry, in video content. This involves implementing state-of-the-art computer vision techniques to handle complex scenes, occlusions, and character movements, ensuring precise screen time measurements.

## 1.4 Project Modules

### 1.4.1 Video Preprocessing

This module focuses on video preprocessing, frame extraction, and object detection. It would include technologies like OpenCV and deep learning models for object detection (e.g., CNN, Faster R-CNN). Additionally, it may involve techniques for tracking objects across frames and accuracy of object identification.

### 1.4.2 Machine Learning and Deep Learning Module:

This module is dedicated to developing the core algorithms for screen time calculation. You would implement deep learning architectures to detect and track actors (e.g., Tom and Jerry) in video frames. Libraries such as TensorFlow or PyTorch would be used for model development and training.

#### **1.4.3 Data Visualization and Reporting Module:**

To convey the results effectively, this module focuses on data visualization techniques and reporting tools. You would use libraries like Matplotlib, Seaborn, or visualization tools such as Tableau or Power BI to present insights from the screen time analysis. This module is essential for providing clear and actionable information to stakeholders in the entertainment industry.

### **1.5 Project Basic Requirements**

1. Training video : tom\_jerry.mp4
2. Test video : evaluate\_tom\_jerry.mp4
3. Label image for train : mapping.csv
4. Store label for test : test.csv
5. Libraries :
  - Numpy
  - Pandas
  - Matplotlib
  - Keras
  - Skimage
  - OpenCV (cv2)

## Chapter 2: Analysis, Design Methodology and Implementation Strategy

### 2.1 Video Preprocessing:

Videos are nothing but a collection of a set of images. These images are called frames and can be combined to get the original video. So, a problem related to video data is not that different from an image classification or an object detection problem. There is just one extra step of extracting frames from the video.

#### 2.1.1 Import and read the video, extract frames from it, and save them as images:

The initial step in the project involves the ingestion of video data. By utilizing libraries such as OpenCV, the video is imported and read frame by frame, with each frame treated as an image. These frames are sequentially extracted and subsequently saved as individual image files, usually in JPEG or PNG format, to form a comprehensive collection of images representing the video's content.

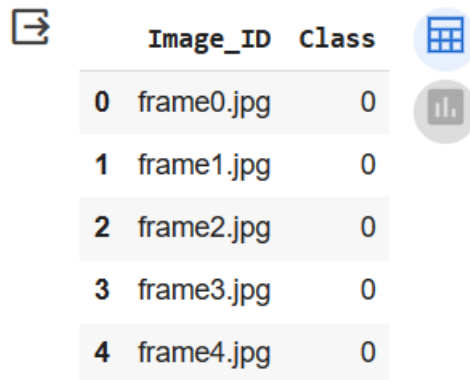


Figure 1: Extracted frame

#### 2.1.2 Label a few images for training the model:

In some instances, a subset of these images may require manual labeling to indicate the presence of specific elements, such as the actors Tom and Jerry, serving as the ground truth

for supervised learning.



	Image_ID	Class
0	frame0.jpg	0
1	frame1.jpg	0
2	frame2.jpg	0
3	frame3.jpg	0
4	frame4.jpg	0

Figure 2:Frames labeling

## 2.2 Machine Learning and Deep Learning Module

### 2.2.1 Building the Model:

We need to make changes to these images similar to the ones we did for the training images. We will preprocess the images, use the `base_model.predict()` function to extract features from these images using the VGG16 pretrained model, reshape these images to 1-D form, and make them zero-centered. We have a hidden layer with 1,024 neurons and an output layer with 3 neurons (since we have 3 classes to predict)

### 2.2.2 Compiling the Model:

Compiling a deep learning model for a screen time calculation project involves several key steps. First, choose an optimizer, such as Adam or SGD, to define how the model's weights will be updated during training. Then, select an appropriate loss function to measure the disparity between predicted and actual screen times.

### 2.2.3 Training the Model:

During training, the model adjusts its parameters to minimize the difference between its predictions and the actual target values in the training data. This iterative process uses optimization techniques to fine-tune the model, making it capable of making accurate predictions on unseen data. Training typically involves splitting data into training and validation sets, selecting an appropriate loss function and optimization algorithm, and setting hyperparameters.

## Chapter 3: Implementation and Testing

### 3.1 Software and Tools

The implementation of the project involves the use of a variety of software and tools across different stages of the project. Here is a comprehensive list of the software and tools utilized:

#### 3.1.1 Video Preprocessing

##### a) Python:

**Description:** Python serves as the primary programming language for implementing various data preprocessing tasks.

**Purpose:** Python's extensive libraries and flexibility make it suitable for handling data and performing essential preprocessing steps.

##### b) pandas:

**Description:** Pandas is a powerful data manipulation library in Python.

**Purpose:** It is used for tasks such as handling missing values, data format conversion, and creating unified "tags" columns.

##### c) OpenCV:

**Description:** Pandas is a powerful data manipulation library in Python.

**Purpose:** It is used for tasks such as handling missing values, data format conversion, and creating unified "tags" columns.

##### d) keras:

**Description:** keras is an open-source deep learning framework that provides a high-level, user-friendly interface for building, training, and deploying neural networks.

**Purpose:** It simplifies the process of developing machine learning models and is widely used for various AI applications, including image and text classification, object detection, and natural language processing.

##### e) skimage:

**Description:** Scikit-Image (skimage) is an open-source image processing library in Python used for tasks like image manipulation, analysis, and computer vision. Content-Based Filtering.

### 3.1.2 Collaboration and Communication

#### a) GitHub:

**Description:** GitHub is a web-based platform for version control and collaboration.

**Purpose:** It is used for hosting the project repository, enabling collaboration, and tracking issues and enhancements.

### 3.1.3 Text Editors or IDEs

#### VsCode:

**Description:** VsCode is an integrated development environment (IDE) for Python.

**Purpose:** It is used for coding, debugging, and managing the project.

### 3.1.4 Notebooks

#### Google Colab:

**Description:** Google Colab is an open-source web application that allows you to create and share documents containing live code, equations, visualizations, and narrative text.

**Purpose:** Google Colab is used for exploratory data analysis, prototyping, and presenting code in an interactive manner.

## 3.2 Implementation Snapshot

Provide snapshots or snippets of key parts of the implementation. This could include code snippets demonstrating data preprocessing, vectorization, and the recommendation algorithm. Screenshots of the web interface during testing may also be included.

```
# i. Building the model
model = Sequential()
model.add(InputLayer((7*7*512,))) # input layer
model.add(Dense(units=1024, activation='sigmoid')) # hidden layer
model.add(Dense(3, activation='softmax')) # output layer
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 1024)	25691136
dense_1 (Dense)	(None, 3)	3075

=====  
Total params: 25694211 (98.02 MB)  
Trainable params: 25694211 (98.02 MB)  
Non-trainable params: 0 (0.00 Byte)

Figure 3: Tweaking VGG16 fully connected layers

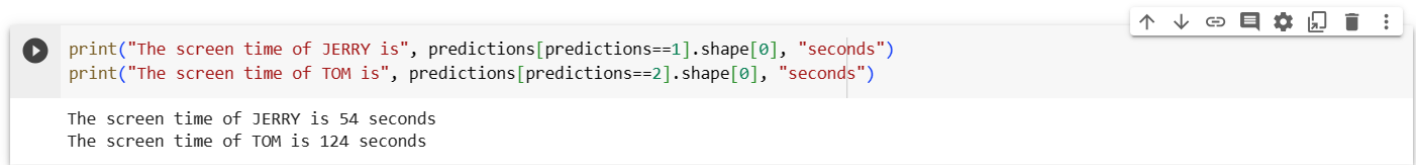
```
# ii. Compiling the model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Figure 4: Compiling the Model

```
# iii. Training the model
model.fit(train, y_train, epochs=100, validation_data=(X_valid, y_valid))
```

Epoch 1/100  
5/5 [=====] - 2s 61ms/step - loss: 0.8346 - accuracy: 0.5944 - val\_loss: 0.4602 - val\_accuracy: 0.8710  
Epoch 2/100  
5/5 [=====] - 0s 16ms/step - loss: 0.2423 - accuracy: 0.9441 - val\_loss: 0.2722 - val\_accuracy: 0.9194  
Epoch 3/100  
5/5 [=====] - 0s 17ms/step - loss: 0.1140 - accuracy: 0.9860 - val\_loss: 0.2172 - val\_accuracy: 0.9355  
Epoch 4/100  
5/5 [=====] - 0s 16ms/step - loss: 0.0654 - accuracy: 0.9860 - val\_loss: 0.1867 - val\_accuracy: 0.9355  
Epoch 5/100  
5/5 [=====] - 0s 20ms/step - loss: 0.0464 - accuracy: 0.9860 - val\_loss: 0.1909 - val\_accuracy: 0.9194  
Epoch 6/100  
5/5 [=====] - 0s 19ms/step - loss: 0.0332 - accuracy: 0.9930 - val\_loss: 0.1801 - val\_accuracy: 0.9194  
Epoch 7/100  
5/5 [=====] - 0s 16ms/step - loss: 0.0221 - accuracy: 0.9930 - val\_loss: 0.1733 - val\_accuracy: 0.9516  
Epoch 8/100  
5/5 [=====] - 0s 19ms/step - loss: 0.0160 - accuracy: 1.0000 - val\_loss: 0.1760 - val\_accuracy: 0.9516  
Epoch 9/100  
5/5 [=====] - 0s 19ms/step - loss: 0.0129 - accuracy: 1.0000 - val\_loss: 0.1844 - val\_accuracy: 0.9355  
Epoch 10/100  
5/5 [=====] - 0s 16ms/step - loss: 0.0106 - accuracy: 1.0000 - val\_loss: 0.1932 - val\_accuracy: 0.9355  
Epoch 11/100  
5/5 [=====] - 0s 17ms/step - loss: 0.0091 - accuracy: 1.0000 - val\_loss: 0.1986 - val\_accuracy: 0.9355  
Epoch 12/100  
5/5 [=====] - 0s 20ms/step - loss: 0.0081 - accuracy: 1.0000 - val\_loss: 0.2004 - val\_accuracy: 0.9355  
Epoch 13/100  
5/5 [=====] - 0s 20ms/step - loss: 0.0073 - accuracy: 1.0000 - val\_loss: 0.1994 - val\_accuracy: 0.9355  
Epoch 14/100  
5/5 [=====] - 0s 17ms/step - loss: 0.0067 - accuracy: 1.0000 - val\_loss: 0.1989 - val\_accuracy: 0.9355  
Epoch 15/100  
5/5 [=====] - 0s 17ms/step - loss: 0.0061 - accuracy: 1.0000 - val\_loss: 0.2005 - val\_accuracy: 0.9355

Figure 5: Training the Model



A screenshot of a Jupyter Notebook cell. The top bar contains a play button icon on the left and a toolbar with icons for up, down, link, comment, settings, copy, delete, and a menu on the right. The code area contains two lines of Python code: `print("The screen time of JERRY is", predictions[predictions==1].shape[0], "seconds")` and `print("The screen time of TOM is", predictions[predictions==2].shape[0], "seconds")`. The output area below shows the results: "The screen time of JERRY is 54 seconds" and "The screen time of TOM is 124 seconds".

```
print("The screen time of JERRY is", predictions[predictions==1].shape[0], "seconds")
print("The screen time of TOM is", predictions[predictions==2].shape[0], "seconds")
```

The screen time of JERRY is 54 seconds  
The screen time of TOM is 124 seconds

Figure 6:Result on evaluate\_tom\_jerry.mp4



## Chapter 4: Conclusion

First, I tried using the pretrained model without removing the top layer. The results were not satisfactory. The possible reason could be that these are the cartoon images and our pretrained model was trained on actual images and hence it was not able to classify these cartoon images. To tackle this problem, i retrained the pretrain model using few labelled images and the results were better from the previous results.

We got an accuracy of around 88% on the validation data and 64% on the test data using this model.

One possible reason for getting a low accuracy on test data could be a lack of training data. As the model does not have much knowledge of cartoon images like TOM and JERRY, we must feed it more images during the training process. My advice would be to extract more frames from different TOM and JERRY videos, label them accordingly, and use them for training the model. Once the model has seen a plethora of images of these two characters, there's a good chance it will lead to a better classification result.

### **Such models can help us in various fields:**

- We can calculate the screen time of a particular actor in a movie
- Calculate the screen time of your favorite superhero, helpful in fan wars!!
- As a buisness point of view it could be helpful in advertisement revenue manangement.
- Like in sports watching cost of advertisement could be different on the basis of viewers.
- These are just a few examples where this technique can be used.

## References

**Project idea and Reference:** <https://www.analyticsvidhya.com/blog/2018/09/deep-learning-video-classification>

**Datasets:** <https://www.youtube.com/watch?v=Si7DovGBDuQ>

**Andrej Karpathy Video Lectures:**

<https://www.youtube.com/playlist?list=PLkt2uSq6rBVctENoVBg1TpCC7OQi31A1C>

**Python documentation:** <https://docs.python.org/3/>

**streamlit documentation:** <https://docs.streamlit.io/library/api-reference>

**pandas documentation:** <https://pandas.pydata.org/docs/>

**numpy documentation:** <https://numpy.org/doc/>

**Project github repository:** <https://github.com/mann718/screentime-calc>