# EyeCare Defender Predicting Vision Health from Digital Insights

Mrs. P.Yamuna[1] Akhila Thalla[2] Kolavennupu Naga Venkata Praneeth[3]

Lingala Vinay Kumar[4] Sangati Chaitanya[5]

[1] Assoc. Professor, ACE Engineering College Hyderabad, India [2]Student, ACE Engineering College Hyderabad, India [3]Student, ACE Engineering College Hyderabad, India [4]Student, ACE Engineering College Hyderabad, India [5]Student, ACE Engineering College Hyderabad, India

Email pyamunaram@gmail.com akhilathalla56@gmail.com , Kpnvpraneeth123@gmail.com vinayvinaykumar7913@gmail.com, chaithanyasangati123@gmail.com

## ABSTRACT:

In an era dominated by digital devices, prolonged screen time has become a significant factor influencing vision health. The "Eye Care Defender Predicting Vision Health from Digital Insights" project leverages advanced data analytics and machine learning to predict and mitigate vision health issues arising from excessive digital device usage. By collecting and analyzing data on screen time duration, intensity, and usage patterns from smartphones, tablets, and computers, this system aims to identify early signs of eye strain and digital eye syndrome. The predictive model developed in this project utilizes comprehensive data inputs to generate personalized recommendations for users.

## I. INTRODUCTION:

The EyeCare Defender Predicting Vision Health from Digital Insights project aims to leverage digital data and predictive analytics to monitor improve eye health. In today's digital age, prolonged exposure to screens and digital devices has become a common part of daily life, increasing the risk of eye strain, digital eye syndrome, and other vision issues. This project utilizes machine learning algorithms and large datasets from digital behavior, such as screen time, lighting conditions, and blink rates, to predict and identify potential vision health problems early on. By analyzing these insights, *EyeCare Defender* seeks to create a preventive approach to eye care, offering personalized recommendations for healthy vision habits and timely interventions for individuals at risk.

## II. OBJECTIVES:

The objectives of the *EyeCare Defender: Predicting Vision Health from Digital Insights* project center on advancing eye health monitoring through digital insights and predictive analytics. This project aims to identify key digital behaviours such as screen exposure, lighting preferences, and viewing distances that significantly impact vision health. By building a predictive model, the project seeks to assess and quantify the risks associated with these factors, identifying individuals who may be prone to digital eye strain or other vision issues. Additionally, EyeCare Defender aims to provide personalized, data-driven recommendations to encourage healthier digital habits, reduce eye strain, and ultimately improve long-term vision health.

## III. PROBLEM STATEMENT

*Predicting Vision Health from Digital Insights* project addresses the growing concern over digital eye strain and vision health degradation due to prolonged screen exposure in the modern world. With people spending increased hours on digital devices for work, education, and leisure, issues like eye fatigue, dryness, and blurred vision have become widespread, leading to a decline in overall eye health and productivity. However, there is a lack of tools that can proactively monitor digital behaviors affecting vision and offer real-time insights or guidance. This project seeks to bridge this gap by developing a predictive model that leverages data from individuals' screen time, lighting conditions, blink rates, and other digital interaction metrics to assess potential risks to eye health. By addressing this problem.

## IV. PROPOSED SYSTEM

The proposed system for EyeCare Defender utilizes advanced data analytics and machine learning to predict and mitigate vision health issues caused by excessive digital device usage. By collecting data on screen time duration, intensity, brightness settings, and usage patterns from smartphones, tablets, and computers, the system identifies early signs of eye strain and digital eye syndrome. The machine learning model analyzes these patterns to generate personalized recommendations, such as taking regular breaks, adjusting screen settings, and performing eye exercises, all aimed at promoting healthier screen habits. Users interact with the system through a dashboard, receive notifications to prevent strain, and track their long-term digital habits. Additionally, the system provides automated screen adjustments based on environmental factors and integrates with existing health apps for a comprehensive approach to preserving vision health in the digital age.

## V. SOFTWARE REQUIREMENTS

VI. Platform : Jupyter Notebook and Visual Studio code Frontend Technologies : Visualization Techniques Backend Technologies : Python

## VII. TECHNOLOGY DESCRIPTION

**Python:**

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used inlanguages such as C++or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open- source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

PACKAGES USED : A few packages have been used in order to build this project.The packages include pandas, NumPy, matplotlib. pyplot, sklearn, seaborn etc. Which are used in the data visualization, data cleaning, data preprocessing and the over all data analysis process. There are many libraries available within these packages which we import and utilize.

## VIII. ALGORITHM

Divide the code into smaller functions, each handling a specific task. This makes the code modular and easier to understand.

**Data Acquisition:**

Collecting the dataset for the analysis is the most crucial part.As some datasets might not have required features and somemight have more null values.

**Data Cleaning:**

In the process of data cleaning the null values are identified, unwanted columns are removed, null values are filled with the mean or some default values and also the outliers are removed.

**Data Integration:**

Data integration refers to the process of bringing together data from multiple sources across an organization to providea complete, accurate, and up-to-date dataset for BI, data analysis and other applications and business processes.

**Data Preprocessing:**

The data preprocessing includes the exploratory data analysisand some other important functions which helps in model building.

**Model building and model evaluation:**

Model building is the most important part of any analysis. It requires to build an accurate model to get the best accuracy for the dataset.

## IX. METHODS

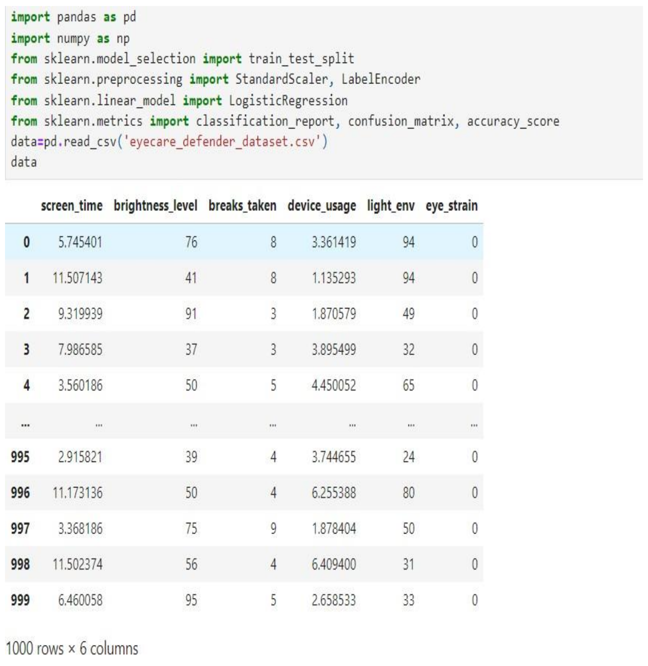
**Frontend:** The user interface of the application is designed using Python-based frameworks such as Flask or Django for web development. These frameworks provide robust tools for creating dynamic, user-friendly web applications. Additionally, libraries like Tkinter or PyQt may be used for building desktop-based graphical user interfaces, ensuring a seamless interaction experience for users.

**Machine Learning Algorithms:** The core functionality of the application is powered by various machine learning algorithms implemented in Python. Libraries such as Scikit-learn, TensorFlow, and PyTorch are utilized for building and training models. The algorithms include supervised learning techniques like regression and classification, as well as unsupervised learning methods such as clustering and dimensionality reduction. The selection of algorithms is based on the specific requirements of the problem domain, ensuring accuracy and efficiency.

**Backend:** The backend is implemented using Python to manage data processing and model deployment. Flask or FastAPI is used to create RESTful APIs that facilitate seamless communication between the frontend and the machine learning models. The backend handles tasks such as data validation, model inference, and result generation, ensuring a secure and scalable architecture.

**Model Training and Evaluation:** The machine learning models are trained using datasets that are preprocessed and cleaned to ensure data quality. Evaluation metrics such as accuracy, precision, recall, and F1 score are used to assess model performance. Techniques such as cross-validation and hyperparameter tuning are applied to optimize the models, ensuring they deliver reliable and accurate predictions.

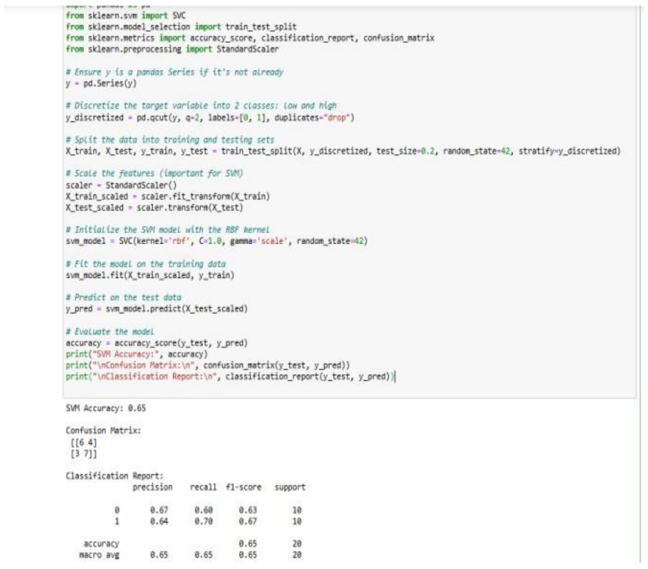
## X. OUTPUT SCREENS



```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
data=pd.read_csv('eyecare_defender_dataset.csv')
data
```

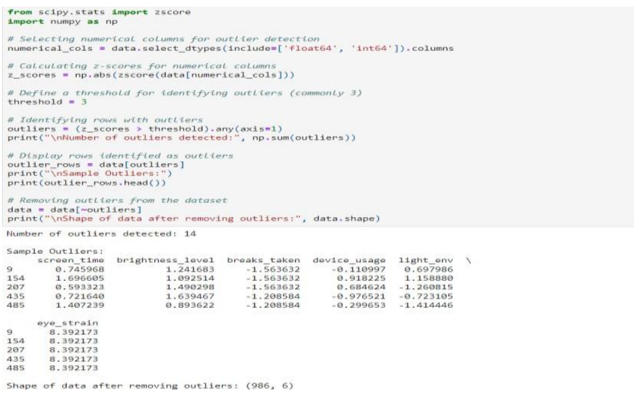
| | screen_time | brightness_level | breaks_taken | device_usage | light_env | eye_strain |
|---|---|---|---|---|---|---|
| 0 | 5.745401 | 76 | 8 | 3.361419 | 94 | 0 |
| 1 | 11.507143 | 41 | 8 | 1.135293 | 94 | 0 |
| 2 | 9.319939 | 91 | 3 | 1.870579 | 49 | 0 |
| 3 | 7.986585 | 37 | 3 | 3.895499 | 32 | 0 |
| 4 | 3.560186 | 50 | 5 | 4.450052 | 65 | 0 |
| ... | ... | ... | ... | ... | ... | ... |
| 995 | 2.915821 | 39 | 4 | 3.744655 | 24 | 0 |
| 996 | 11.173136 | 50 | 4 | 6.255388 | 80 | 0 |
| 997 | 3.368186 | 75 | 9 | 1.878404 | 50 | 0 |
| 998 | 11.502374 | 56 | 4 | 6.409400 | 31 | 0 |
| 999 | 6.460058 | 95 | 5 | 2.658533 | 33 | 0 |

1000 rows × 6 columns



```
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.preprocessing import StandardScaler

# Ensure y is a pandas Series if it's not already
y = pd.Series(y)

# Discretize the target variable into 2 classes: low and high
y_discretized = pd.qcut(y, q=2, labels=[0, 1], duplicates="drop")

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y_discretized, test_size=0.2, random_state=42, stratify=y_discretized)

# Scale the features (important for SVM)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Initialize the SVM model with the RBF kernel
svm_model = SVC(kernel='rbf', C=1.0, gamma='scale', random_state=42)

# Fit the model on the training data
svm_model.fit(X_train_scaled, y_train)

# Predict on the test data
y_pred = svm_model.predict(X_test_scaled)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("SVM Accuracy:", accuracy)
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

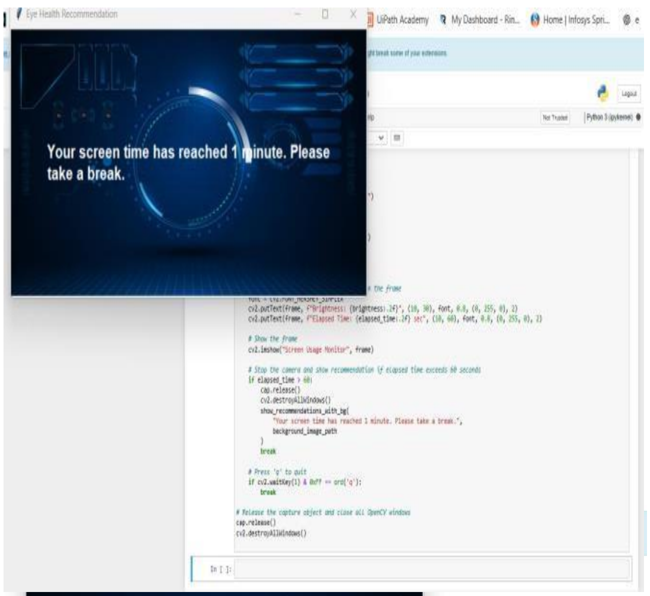
```
SVM Accuracy: 0.65

Confusion Matrix:
 [[6 4]
 [3 7]]

Classification Report:
              precision    recall  f1-score   support

           0       0.67      0.60      0.63        10
           1       0.64      0.70      0.67        10

    accuracy                           0.65        20
   macro avg       0.65      0.65      0.65        20
```



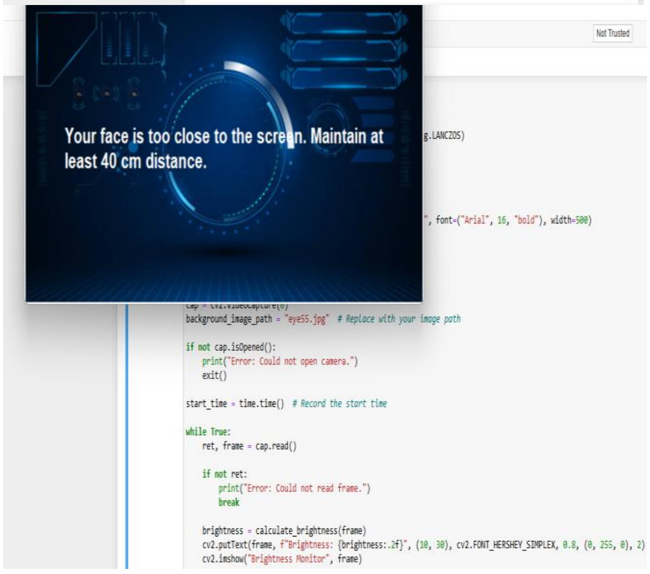
```
from scipy.stats import zscore
import numpy as np

# Selecting numerical columns for outlier detection
numerical_cols = data.select_dtypes(include=['float64', 'int64']).columns

# Calculating z-scores for numerical columns
z_scores = np.abs(zscore(data[numerical_cols]))

# Define a threshold for identifying outliers (commonly 3)
threshold = 3

# Identifying rows with outliers
outliers = (z_scores > threshold).any(axis=1)
print("\nNumber of outliers detected:", np.sum(outliers))

# Display rows identified as outliers
outlier_rows = data[outliers]
print("\nSample Outliers:")
print(outlier_rows.head())

# Removing outliers from the dataset
data = data[~outliers]
print("\nShape of data after removing outliers:", data.shape)
```

```
Number of outliers detected: 14

Sample Outliers:
     screen_time  brightness_level  breaks_taken  device_usage  light_env  \
9       0.745968          1.241683     -1.563632     -0.110997   0.697986
154     1.696605          1.092514     -1.563632      0.918225   1.158880
207     0.593323          1.490298     -1.563632      0.684624  -1.260815
435     0.721640          1.639467     -1.208584     -0.976521  -0.723105
485     1.407239          0.893622     -1.208584     -0.299653  -1.414446

     eye_strain
9      8.392173
154    8.392173
207    8.392173
435    8.392173
485    8.392173

Shape of data after removing outliers: (986, 6)
```

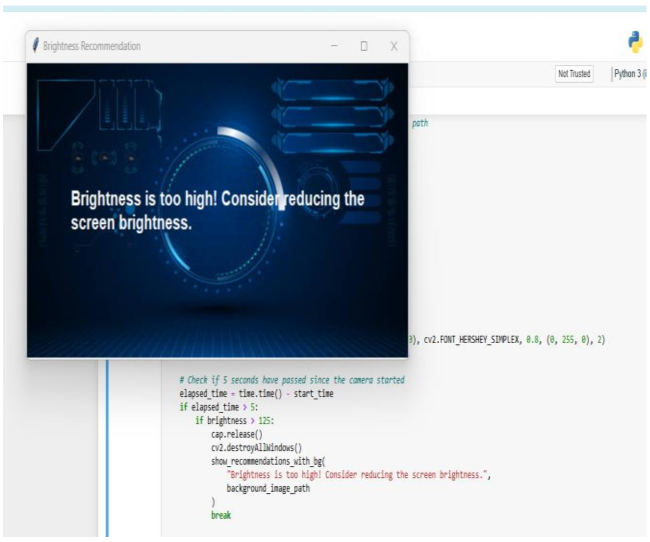
ISSN: 2583-6129
DOI: 10.55041/ISJEM02222


Eye Health Recommendation
Your screen time has reached 1 minute. Please take a break.
Not Trusted
Python 3 (ipykernel)

```
    cv2.putText(frame, f"Brightness: {brightness:.2f}", (10, 30), font, 0.8, (0, 255, 0), 2)
    cv2.putText(frame, f"Elapsed Time: {elapsed_time:.2f} sec", (10, 60), font, 0.8, (0, 255, 0), 2)

    # Show the frame
    cv2.imshow("Screen Usage Monitor", frame)

    # Stop the camera and show recommendation if elapsed time exceeds 60 seconds
    if elapsed_time > 60:
        cap.release()
        cv2.destroyAllWindows()
        show_recommendations_with_bg(
            "Your screen time has reached 1 minute. Please take a break.",
            background_image_path
        )
        break

    # Press 'q' to quit
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release the capture object and close all OpenCV windows
cap.release()
cv2.destroyAllWindows()
```


Your face is too close to the screen. Maintain at least 40 cm distance.
Not Trusted

```
background_image_path = "eye55.jpg"  # Replace with your image path

if not cap.isOpened():
    print("Error: Could not open camera.")
    exit()

start_time = time.time()  # Record the start time

while True:
    ret, frame = cap.read()

    if not ret:
        print("Error: Could not read frame.")
        break

    brightness = calculate_brightness(frame)
    cv2.putText(frame, f"Brightness: {brightness:.2f}", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 0), 2)
    cv2.imshow("Brightness Monitor", frame)
```

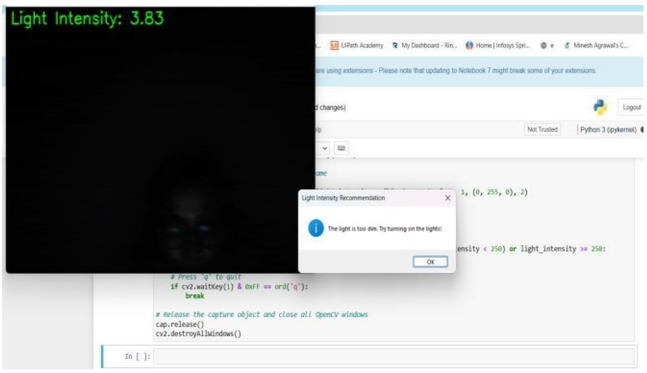

Brightness Recommendation
Brightness is too high! Consider reducing the screen brightness.
Not Trusted
Python 3

```
    # Check if 5 seconds have passed since the camera started
    elapsed_time = time.time() - start_time
    if elapsed_time > 5:
        if brightness > 125:
            cap.release()
            cv2.destroyAllWindows()
            show_recommendations_with_bg(
                "Brightness is too high! Consider reducing the screen brightness.",
                background_image_path
            )
            break
```

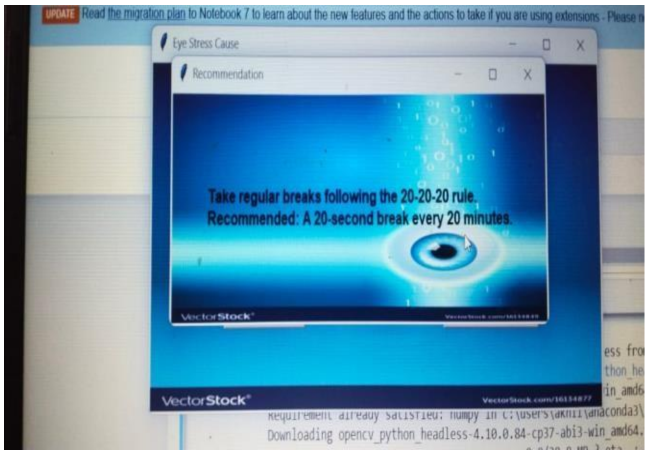
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
data=pd.read_csv('eyecare_defender_dataset.csv')
data
```

| | screen_time | brightness_level | breaks_taken | device_usage | light_env | eye_strain |
|---|---|---|---|---|---|---|
| **0** | 5.745401 | 76 | 8 | 3.361419 | 94 | 0 |
| **1** | 11.507143 | 41 | 8 | 1.135293 | 94 | 0 |
| **2** | 9.319939 | 91 | 3 | 1.870579 | 49 | 0 |
| **3** | 7.986585 | 37 | 3 | 3.895499 | 32 | 0 |
| **4** | 3.560186 | 50 | 5 | 4.450052 | 65 | 0 |
| **...** | ... | ... | ... | ... | ... | ... |
| **995** | 2.915821 | 39 | 4 | 3.744655 | 24 | 0 |
| **996** | 11.173136 | 50 | 4 | 6.255388 | 80 | 0 |
| **997** | 3.368186 | 75 | 9 | 1.878404 | 50 | 0 |
| **998** | 11.502374 | 56 | 4 | 6.409400 | 31 | 0 |
| **999** | 6.460058 | 95 | 5 | 2.658533 | 33 | 0 |

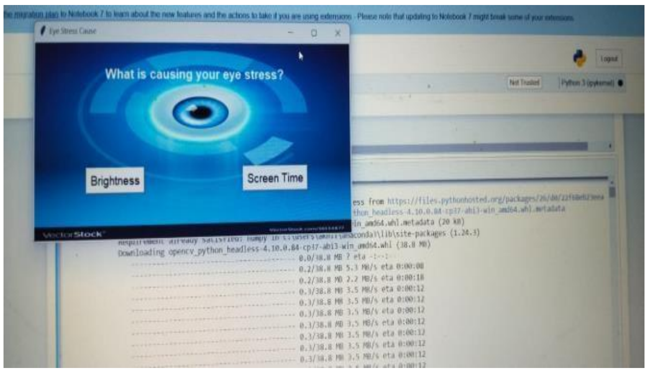
1000 rows × 6 columns


Light Intensity: 3.83
Light Intensity Recommendation
The light is too dim. Try turning on the lights!
OK
Not Trusted
Python 3 (ipykernel)

```
    ensity < 250) or light_intensity >= 250:
    # Press 'q' to quit
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release the capture object and close all OpenCV windows
cap.release()
cv2.destroyAllWindows()
```


UPDATE Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions
Eye Stress Cause
Recommendation
Take regular breaks following the 20-20-20 rule.
Recommended: A 20-second break every 20 minutes.
VectorStock
VectorStock.com/16134877

```
Requirement already satisfied: numpy in c:\users\akhil\anaconda3\
Downloading opencv_python_headless-4.10.0.84-cp37-abi3-win_amd64.
```



```
Downloading opencv_python_headless-4.10.0.84-cp37-abi3-win_amd64.whl (38.8 MB)
```

ISSN: 2583-6129
DOI: 10.55041/ISJEM02222

## XI. CONCLUSION

In Conclusion, The "Eye Care Defender Predicting Vision Health from Digital Insights" project addresses the critical impact of prolonged digital screen exposure on vision health. By leveraging advanced data analytics and machine learning, the system provides an innovative approach to predicting and mitigating issues like eye strain and digital eye syndrome. Through the collection and analysis of screen time, the platform enables early detection of potential vision problems. This solution emphasizes the importance of proactive vision care in the digital age. By raising awareness and promoting healthier screen usage, the project contributes to enhancing users' overall eye health. Its data-driven approach ensures precision and scalability, making it a valuable tool for modern health management. This initiative underscores the potential of technology in fostering preventive healthcare solutions.

## XII. REFERENCES

https://journals.sagepub.com/doi/full/10.1177/11206721221131397
https://link.springer.com/article/10.1186/s12913-019-4493-3
https://onlinelibrary.wiley.com/doi/abs/10.1111/opo.12699
IP-Koon-ching-2022.-The-Role-of-Smartphones-in-Eyecare-A-Systemic-Review.-SAERA.pdf
https://link.springer.com/article/10.1007/s13167-024-00372-6
https://link.springer.com/article/10.1186/s12913-019-4493-3
3rdEdition,MorganKaufmannPublishers.