ASSIGNMENT # 03

University of Sialkot
Faculty of Computing and IT

Time Waster On Social Media

Bs Computer Science

Spring 2022-Fall 2022

Submitted to
Sir Advin Masih

Submitted By

Name	Roll Number
Adil Arshad	017
Muhammad Abu Hurairah Baig	015
Muhammad Abu Bakr Baig	014

Table of Contents

1. Project Title……..6

1.1 Introduction……6

1.2 Literature Review……...6

1.3 Methodology……9

1.4 Training Phase……10

1.5 Testing Phase…….16

1.6 Parameter Comparision……..18

1.7 References……..18

List of Table

Table 1.1 Existing Research Work Articles……..8

Table 1.2 Parameter Comparision……18

List of Figures

Figure 1.1 Random Forest Training and Evaluation……11

Figure 1.2 Navie Bayes Training and Evaluation…….12

Figure 1.3 Logistic Regression Training and Evaluation……..13

Figure 1.4 SVM Training and Evaluation…….15

Figure 1.5 KNN Training and Evaluation……..16

Abstract

The exponential rise in social media usage has significantly altered user behavior, often leading to excessive time spent online, particularly among students and young adults. This study aims to detect and analyze time-wasting patterns on social media platforms using machine learning techniques. A labeled dataset was utilized, comprising behavioral and usage features indicative of social media overuse. The data required minimal preprocessing, as it contained no missing or duplicate values—only encoding and scaling were applied. Multiple classification models were trained and evaluated, including Random Forest, Naive Bayes, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Logistic Regression. Among them, Random Forest and Naive Bayes achieved the highest performance across accuracy, precision, recall, and F1-score metrics, while KNN showed the weakest results. The entire workflow—from preprocessing to training and evaluation—was executed in the Google Colab environment. The results highlight the potential of AI-based approaches in identifying and mitigating digital behavior concerns, and suggest practical applications in building tools for productivity and digital well-being.

Project Title:

Time-Waster on Social Media

Introduction:

Social media has become an inseparable part of modern life, offering platforms for communication, entertainment, and information exchange. While these platforms have transformed how people interact and stay informed, they have also introduced a growing concern: the excessive and unproductive use of time. This phenomenon, commonly referred to as "time-wasting on social media," is particularly prevalent among students and young adults who often find themselves spending long hours scrolling through feeds, watching videos, or engaging in non-essential online activities. Such behavior can lead to decreased productivity, poor academic performance, and adverse effects on mental health and overall well-being.

In response to this issue, the present study aims to leverage machine learning to identify patterns in user behavior that indicate a tendency toward time-wasting on social media. By analyzing a dataset containing behavioral and demographic information, several classification algorithms were applied to predict users who are more likely to exhibit such tendencies. The models used include K-Nearest Neighbors, Support Vector Machine, Naive Bayes, Logistic Regression, and Random Forest. Each model was trained and evaluated using standard performance metrics such as precision, recall, and F1-score to ensure a comprehensive assessment of their effectiveness.

The findings reveal that some models are significantly better at capturing the patterns associated with excessive social media use, while others struggle to generalize well across the dataset. This report explores the methodology used in building and evaluating these models, analyzes their comparative performance, and discusses the implications of using machine learning to address the broader issue of digital well-being. Ultimately, this study seeks to contribute meaningful insights that can help in early detection and intervention for individuals who may be at risk of wasting valuable time on social media.

Literature Review

Tahir Ehsan & Jamshaid Basit (2025) explores how machine learning algorithms can identify addictive patterns in social media users by analyzing behavioral traits and mental health indicators. The study employs a dataset containing self-reported mental health data and social media habits to train classification models. Random Forest and Logistic Regression performed notably well in detecting users with high addiction risks. The work highlights the connection between digital behavior and mental health, providing a framework for early intervention. It underscores the potential of AI to support digital wellness efforts in society.

PostOwl (2025), a productivity assistant designed to help users reduce time spent unproductively on social media. It utilizes AI to analyze user engagement, deliver optimal posting schedules, and cut down on meaningless scrolling. The tool provides

behavioral insights and personalized feedback, promoting more intentional usage of platforms. It serves as a practical solution for content creators and marketers trying to balance online presence with productivity. The AI model focuses on saving time while maintaining digital influence.

Himaya Chamudini (2024) presents a case study on using supervised machine learning to predict social media addiction in students. The model is built on labeled data reflecting users' habits, screen time, and emotional patterns. It emphasizes how early predictions can help educational institutions and parents intervene effectively. The article breaks down the training process and showcases how even simple models like Logistic Regression can achieve high accuracy. It advocates for responsible AI usage to improve student focus and well-being.

Shanmugasundaram & Tamilarasu (2023) investigates the broader cognitive consequences of excessive social media usage and the role AI plays in influencing attention and memory. The study provides evidence that prolonged engagement with social media can impair cognitive control and increase distraction, especially in younger users. It also explores how recommender systems and addictive design patterns powered by AI contribute to time-wasting behaviors. The authors call for more ethical design practices and digital literacy programs. This work is crucial for understanding long-term neurological impacts of social platforms.

Life Management Science Labs (2023) focuses on AI tools that enhance productivity by limiting addictive scrolling and news feed consumption. It discusses AI-powered browser extensions, time trackers, and personalized recommendations that help users set boundaries with digital content. The article positions AI not just as a cause but also as a solution to digital overload. It encourages building self-awareness through smart alerts and usage summaries. This piece offers both theoretical insights and practical strategies to reduce time waste online.

ACM UMAP (2023) presents a novel approach that leverages personalized time-loss aversion to nudge users away from overusing social media. By customizing interventions based on user preferences and time perception, the AI models create real-time prompts to increase self-control. The study found these personalized nudges significantly reduced daily usage durations without harming user experience. The research contributes to persuasive technology design with AI, focusing on behavioral change and self-regulation. It bridges psychology and computer science for digital wellness.

MDPI Electronics (2023) provides a comprehensive review of AI techniques used in social media analysis, including natural language processing, user profiling, and behavioral prediction. Although not solely focused on time-wasting, it highlights how AI can detect anomalies in usage patterns that indicate overuse or addiction. The review covers classification, clustering, and sentiment analysis methods commonly employed in the domain. It underscores the growing role of AI in monitoring and moderating digital engagement. The paper serves as a solid foundation for further research in behavior prediction.

S.No	Author & Year	Research Topic	Research Methodology	Research Result
1	Ehsan & Basit (2025)	Detecting Social Media Addiction Patterns Using ML	Collected behavioral and mental health data; applied classification models like Random Forest and Logistic Regression	ML models accurately predicted addiction levels; Random Forest performed best
2	PostOwl (2025)	Reducing Time Waste on Social Media Using PostOwl AI	Descriptive analysis of PostOwl's AI features for managing social media time	Significant reduction in unproductive scrolling; improved user productivity
3	Chamudini (2024)	Predicting Social Media Addiction in Students	Used student behavior data with ML models such as Logistic Regression and Decision Trees	Logistic Regression achieved high accuracy; ML can effectively identify addiction-prone students
4	Shanmugasundaram & Tamilarasu (2023)	Impact of AI and Social Media on Cognitive Functions	Literature review and neuroscience-based analysis	AI-driven social platforms negatively impact attention and memory; increase digital distraction
5	Life Management Science Labs (2023)	Using AI for Productivity and Reducing Social Media Addiction	Case-based analysis of productivity tools and time-management AI	Tools helped reduce screen time and improve digital discipline
6	Morgenroth & Krämer (2023)	Personalized Time Loss Aversion to Reduce Social Media Use	Experimental AI study with real-time nudging based on user behavior	Personalized nudges reduced usage time while maintaining satisfaction
7	Albahli (2023)	AI Algorithms for Social Media Behavior Analysis	Systematic review of AI techniques like NLP, clustering, and behavioral modeling	AI models effectively identified overuse and patterns linked to addiction

Table 1.1 Existing Research Work

3. Methodology:

This study aims to identify patterns of social media addiction and time-wasting behaviors using supervised machine learning techniques. The following steps outline the methodology followed in this research:

3.1 Data Collection

The dataset used in this study, titled *"Time Wasters on Social Media"*, was sourced from a structured CSV file. It contains several user behavior attributes, including demographic features, usage patterns, and self-reported indicators of social media overuse.

3.2 Data Preprocessing

Upon loading the dataset, it was found to be clean and free of any missing or duplicate values. The preprocessing pipeline involved two primary steps:

Label Encoding: All categorical variables were converted to numerical format using label encoding to ensure compatibility with machine learning algorithms.

Feature Scaling: Numerical features were scaled using standard normalization techniques to bring them to a uniform range and improve model convergence and performance.

3.3 Model Training

The processed data was then used to train and evaluate five widely used classification algorithms:

- **Random Forest**
- **Naive Bayes**
- **Support Vector Machine (SVM)**
- **K-Nearest Neighbors (KNN)**
- **Logistic Regression**

Each model was evaluated based on four standard classification metrics:

- Accuracy
- Precision
- Recall
- F1-Score

The results were analyzed to determine the relative performance of each model:

- **Best Models:** Random Forest and Naive Bayes showed the highest performance across all four metrics.

- **Good Performance:** Logistic Regression performed reasonably well but was slightly outperformed by the top two.
- **Average Performance:** SVM achieved moderate results, indicating limited generalization in this context.
- **Poor Performance:** KNN showed the lowest scores on all evaluation metrics.

3.4 Comparative Analysis

To effectively compare model performance, a tabular and graphical comparison of all four metrics across the five algorithms was prepared. This comparison highlights the trade-offs and consistency between precision, recall, and overall accuracy. Bar plots and line graphs can be used to visually interpret performance differences and support model selection.

3.5 Tools and Environment

All experiments in this study were conducted on **Google Colab**, an online Python-based development environment that supports high-performance computing and ease of collaboration. Core Python libraries such as **Pandas**, **Scikit-learn**, **NumPy**, and **Matplotlib** were used for data loading, preprocessing, model training, evaluation, and visualization. Google Colab's cloud-based infrastructure enabled efficient execution of machine learning models without the need for local computational resources.

Training Phase:

- The **Random Forest** and **Naive Bayes** classifiers demonstrated exceptional performance across all four evaluation metrics. Random Forest, an ensemble method, effectively captured complex patterns by combining multiple decision trees, resulting in high precision and recall. Naive Bayes, though based on the assumption of feature independence, performed surprisingly well, indicating that the dataset's structure aligned well with its probabilistic approach, making it both efficient and accurate.

```
# 1. Import Libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, classification_report

# 2. Load Data
df = pd.read_csv('/content/Time-Wasters on Social Media.csv')  # adjust
if file path differs
```

```
# 3. Encode Categorical Features
label_encoders = {}
```

```
for column in df.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    df[column] = le.fit_transform(df[column])
    label_encoders[column] = le
```

```
# 4. Define Features (X) and Target (y)
X = df.drop('Addiction Level', axis=1)
y = df['Addiction Level']
```

```
# 5. Split Data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

```
# 6. Train Model
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)
```

```
# 7. Evaluate Model
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision     (macro):",     precision_score(y_test,     y_pred,
average='macro'))
print("Recall (macro):", recall_score(y_test, y_pred, average='macro'))
print("F1 Score (macro):", f1_score(y_test, y_pred, average='macro'))
```

```
print("\nDetailed Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.995
Precision (macro): 0.9967105263157895
Recall (macro): 0.9962121212121212
F1 Score (macro): 0.9964102564102564
```

Fig 1.1 Random Forest Training and Evaluation

```
# 1. Import Required Libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, classification_report

# 2. Load Dataset
df = pd.read_csv('/content/Time-Wasters on Social Media.csv')  # Use your
path if different
```

```
# 3. Encode Categorical Columns
label_encoders = {}
for col in df.select_dtypes(include='object').columns:
    le = LabelEncoder()
```

```
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le
# 4. Define Features and Target
X = df.drop('Addiction Level', axis=1)
y = df['Addiction Level']
```

```
# 5. Scale Features (optional but okay to use for consistency)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
# 6. Split Dataset
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.2, random_state=42)
```

```
# 7. Train Naive Bayes Model
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)
```

```
# 8. Evaluate Model
y_pred = nb_model.predict(X_test)
```

```
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision (macro):", precision_score(y_test, y_pred,
average='macro'))
print("Recall (macro):", recall_score(y_test, y_pred, average='macro'))
print("F1 Score (macro):", f1_score(y_test, y_pred, average='macro'))
print("\nDetailed Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.995
Precision (macro): 0.9970238095238095
Recall (macro): 0.9962121212121212
F1 Score (macro): 0.9965708989805375
```

Fig 1.1 Naive Bayes Training and Evaluation

- Logistic Regression showed solid performance, particularly in terms of accuracy and precision. As a linear model, it handled the classification task effectively, but it slightly underperformed in recall and F1 score compared to Random Forest and Naive Bayes. This indicates that while it was good at making correct positive predictions, it may have missed some of the less obvious cases of social media overuse.

```
# 1. Import Libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, classification_report
```

```
# 2. Load Dataset
df = pd.read_csv('/content/Time-Wasters on Social Media.csv')  # Update
path if needed
```

```
# 3. Encode Categorical Columns
label_encoders = {}
for col in df.select_dtypes(include='object').columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le
```

```
# 4. Define Features and Target
X = df.drop('Addiction Level', axis=1)
y = df['Addiction Level']
```

```
# 5. Scale the Features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
# 6. Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.2, random_state=42)
```

```
# 7. Train Logistic Regression Model
log_reg_model = LogisticRegression(max_iter=1000)
log_reg_model.fit(X_train, y_train)
```

```
# 8. Predict and Evaluate
y_pred = log_reg_model.predict(X_test)
```

```
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision (macro):", precision_score(y_test, y_pred,
average='macro'))
print("Recall (macro):", recall_score(y_test, y_pred, average='macro'))
print("F1 Score (macro):", f1_score(y_test, y_pred, average='macro'))
print("\nDetailed Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.89
Precision (macro): 0.8739272492082608
Recall (macro): 0.7142750644745326
F1 Score (macro): 0.7287829380764164
```

Fig 1.1 Logistic Regression Training and Evaluation

- The **SVM model** achieved moderate accuracy and precision, but its performance dropped in terms of recall and F1 score. This suggests that while SVM could distinguish between classes in many cases, it struggled with generalizing well across all patterns in the data. This could be due to sensitivity to feature scaling or the linear separability of classes in the feature space.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, classification_report
# Load dataset
df = pd.read_csv('/content/Time-Wasters on Social Media.csv')

# Drop missing values
df.dropna(inplace=True)
```

```
# Label encode categorical columns
label_encoders = {}
for col in df.select_dtypes(include='object').columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le
```

```
# Split features and label
target_col = 'Addiction Level'  # Update if different
X = df.drop(columns=[target_col])
y = df[target_col]
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

```
# Feature scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
# Initialize and train the SVM model
svm = SVC(kernel='rbf', C=1, gamma='scale')  # 'rbf' is a good default
kernel
svm.fit(X_train_scaled, y_train)
# Predict on test set
y_pred = svm.predict(X_test_scaled)
```

```
# Evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
precision   =   precision_score(y_test,   y_pred,   average='weighted',
zero_division=0)
recall = recall_score(y_test, y_pred, average='weighted', zero_division=0)
f1 = f1_score(y_test, y_pred, average='weighted', zero_division=0)
```

```
# Print results
print("📊 SVM Model Evaluation")
print(f"Accuracy : {accuracy:.4f}")
```

```
print(f"Precision: {precision:.4f}")
print(f"Recall   : {recall:.4f}")
print(f"F1 Score : {f1:.4f}")
print("\nClassification Report:\n", classification_report(y_test, y_pred,
zero_division=0))D
```

```
SVM Model Evaluation
Accuracy : 0.8600
Precision: 0.8301
Recall   : 0.8600
F1 Score : 0.8200
```

Fig 1.1 SVM Training and Evaluation

- **KNN** was the least effective model in this analysis. It showed the lowest performance across all evaluation metrics. KNN's reliance on distance metrics may have been less effective in this dataset, possibly due to high-dimensional data or the lack of clear clustering in feature space. Its relatively poor generalization suggests it is not well-suited for detecting nuanced behavioral patterns in this specific context.

```
# 1. Import Libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, classification_report

# 2. Load Dataset
df = pd.read_csv('/content/Time-Wasters on Social Media.csv')  # Adjust
path if needed
```

```
# 3. Encode Categorical Features
label_encoders = {}
for col in df.select_dtypes(include='object').columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le
```

```
# 4. Feature & Target
X = df.drop('Addiction Level', axis=1)
y = df['Addiction Level']
```

```
# 5. Scale Features (VERY important for KNN)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
# 6. Train/Test Split
```

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

```
# 7. Train KNN Model
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train, y_train)
```

```
# 8. Evaluation
y_pred = knn_model.predict(X_test)
```

```
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision (macro):", precision_score(y_test, y_pred, average='macro'))
print("Recall (macro):", recall_score(y_test, y_pred, average='macro'))
print("F1 Score (macro):", f1_score(y_test, y_pred, average='macro'))
print("\nDetailed Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.57
Precision (macro): 0.4693456073496396
Recall (macro): 0.3676425062841773
F1 Score (macro): 0.3592057987640971
```

Fig 1.1 KNN Training and Evaluation

4. Testing Phase

After successfully training the machine learning models, the testing phase was conducted to evaluate how well the models generalized to unseen data. This phase plays a crucial role in verifying the performance and reliability of each algorithm on real-world-like scenarios.

All testing procedures were carried out in **Google Colab**, leveraging its flexible cloud-based Python environment. The trained models were validated using the reserved test portion of the dataset, which had not been seen during training. Each model was subjected to the same test data to ensure a fair comparison of predictive performance across multiple evaluation metrics: **Accuracy**, **Precision**, **Recall**, and **F1 Score**.

The testing process involved:

- Feeding the test set into each trained model.
- Generating predicted labels.
- Comparing the predictions with the actual labels to compute evaluation metrics.

This consistent evaluation approach enabled an objective performance assessment. The results confirmed that **Random Forest** and **Naive Bayes** significantly outperformed the other models, showing high predictive power and generalizability. **Logistic Regression** also performed reasonably well, while **SVM** offered moderate results. **KNN**, however, displayed the weakest performance, likely due to its sensitivity to data scaling and the structure of the feature space, e.g:

```
# Helper: show available values from the dataset for each feature
def show_possible_values(df, label_encoders):
     print("Enter values ONLY from the following options for each
feature:\n")
    for column in df.columns:
        if column in label_encoders:
            original_values = label_encoders[column].classes_
            print(f"{column}: {list(original_values)}")
        else:
                                print(f"{column}:    min={df[column].min()},
max={df[column].max()}")


# Step 1: Show possible values for user
show_possible_values(X, label_encoders)
```

```
# Step 2: Manually input each feature value
# For simplicity, using input() for each field – can be replaced with a
form in GUI
```

```
input_data = {}
for column in X.columns:
    if column in label_encoders:
        value = input(f"Enter value for '{column}' from the options above:
")
        # Convert to encoded label
        encoded_val = label_encoders[column].transform([value])[0]
        input_data[column] = encoded_val
    else:
        value = input(f"Enter numerical value for '{column}': ")
        input_data[column] = int(value)
```

```
# Step 3: Convert input into DataFrame
input_df = pd.DataFrame([input_data])
```

```
# Step 4: Predict using trained model
predicted = model.predict(input_df)
print(f"\nPredicted Addiction Level: {predicted[0]}")
```

In conclusion, the testing phase validated the effectiveness of ensemble and probabilistic models for detecting patterns of social media overuse. The high-scoring models demonstrate practical potential for implementation in predictive systems aimed at identifying and mitigating time-wasting behaviors on social media platforms.

Parameter Comparision:

S.No	Model	Accuracy	Precision (Macro)	Recall (Macro)	F1 Score (Macro)
1	Naive Bayes	0.995	0.9970	0.9962	0.9966
2	Random Forest	0.995	0.9967	0.9962	0.9964
3	Logistic Regression	0.890	0.8739	0.7143	0.7288
4	Support Vector Machine (SVM)	0.860	0.8301	0.8600	0.8200
5	K-Nearest Neighbors (KNN)	0.570	0.4693	0.3676	0.3592

Table 1.2 Comparision Table

References:

[1] T. Ehsan and J. Basit, "Machine Learning for Detecting Social Media Addiction Patterns: Analyzing User Behavior and Mental Health Data," *ResearchGate*, 2025. Available: https://www.researchgate.net/publication/387798954_Machine_Learning_for_Detecting_Social_Media_Addiction_Patterns_Analyzing_User_Behavior_and_Mental_Health_Data.

[2] PostOwl, "Stop Wasting Time on Social Media: How PostOwl AI Helps You Focus," 2025. [Online]. Available: https://postowl.io/blog/stop-wasting-time-social-media-postowl-ai?utm_source=chatgpt.com.

[3] H. Chamudini, "Predicting Social Media Addiction in Students: A Machine Learning Approach," *Medium*, 2024. [Online]. Available: https://medium.com/@himayachamudini123/predicting-social-media-addiction-in-students-a-machine-learning-approach-can-help-us-manage-b2da3c358530.

[4] S. Shanmugasundaram and M. Tamilarasu, "The Impact of Digital Technology, Social Media, and Artificial Intelligence on Cognitive Functions," *Frontiers in Cognition*, vol. 9, article 1203077, Nov. 2023. Available: https://www.frontiersin.org/articles/10.3389/fcogn.2023.1203077/full.

[5] Life Management Science Labs, "Using AI for Productivity: Outsmarting News Feeds and Social Media Addiction," 2023. [Online]. Available: https://insights.lifemanagementsciencelabs.com/using-ai-for-productivity/?utm_source=chatgpt.com.

[6] A. Morgenroth and N. C. Krämer, "Personalizing Time Loss Aversion to Reduce Social Media Use," in *Proc. 31st ACM User Modeling, Adaptation and Personalization Conf.* (UMAP '23), Alicante, Spain, Jul. 2023. DOI:10.1145/3565472.3592951. Available: https://dl.acm.org/doi/abs/10.1145/3565472.3592951.

[7] S. Albahli, "A Review on Artificial Intelligence Algorithms for Social Media Analysis," *Electronics*, vol. 10, no. 1, article 5, Jan. 2023. Available: https://www.mdpi.com/2079-9292/10/1/5.