

Configuration Manual

MSc Research Project
(MSc in AI for Business) Practicum

Raazia Liaqat
x23405970

School of Computing
National College of Ireland

Supervisor: Victor del Rosal

National College of Ireland
MSc Project Submission Sheet
School of Computing



Student Name: Raazia Liaqat
x23405970

Student ID:

Programme: Masters in AI for Business **Year:** 2024-2025

Module: Practicum

Supervisor: Victor del Rosal

Submission Due Date:

July 31st, 2025,

Project Title:
**Optimizing Customer Engagement and Business Growth in the
Photography Industry Through Predictive AI-Powered Analytics**

Word Count: **Pages 22**
Count 1809.

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Raazia Liaqat

Date: 30/07/2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	●
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	●
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	●

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

1. Introduction

This configuration manual details the installation, setup, execution, and testing instructions for the predictive analytics system developed as part of the MSc Research Project. The system utilizes Python-based computer vision and machine learning pipelines to analyze photographic content and predict engagement levels, offering value to photographers and creative small businesses.

2. System Requirements

- Operating System: Windows 10/11 or macOS (Ventura+)
- Python Version: 3.12 via Anaconda
- IDE: Visual Studio Code or Jupyter Notebook
- RAM: Minimum 8 GB
- Disk Space: At least 2 GB free

3. Installation and Setup

Step 1: Install Anaconda from <https://www.anaconda.com/products/distribution>

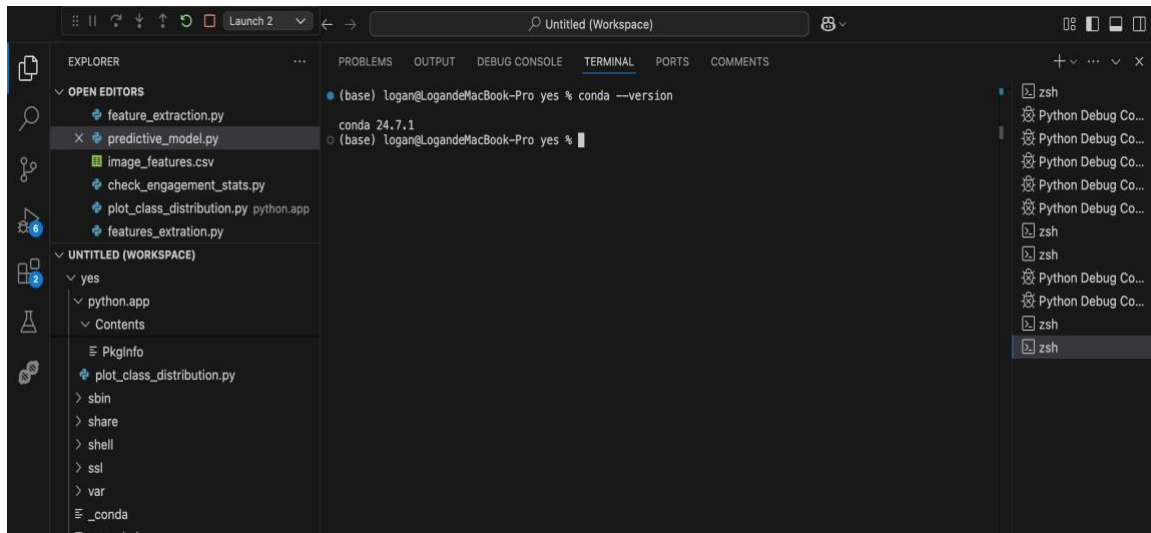


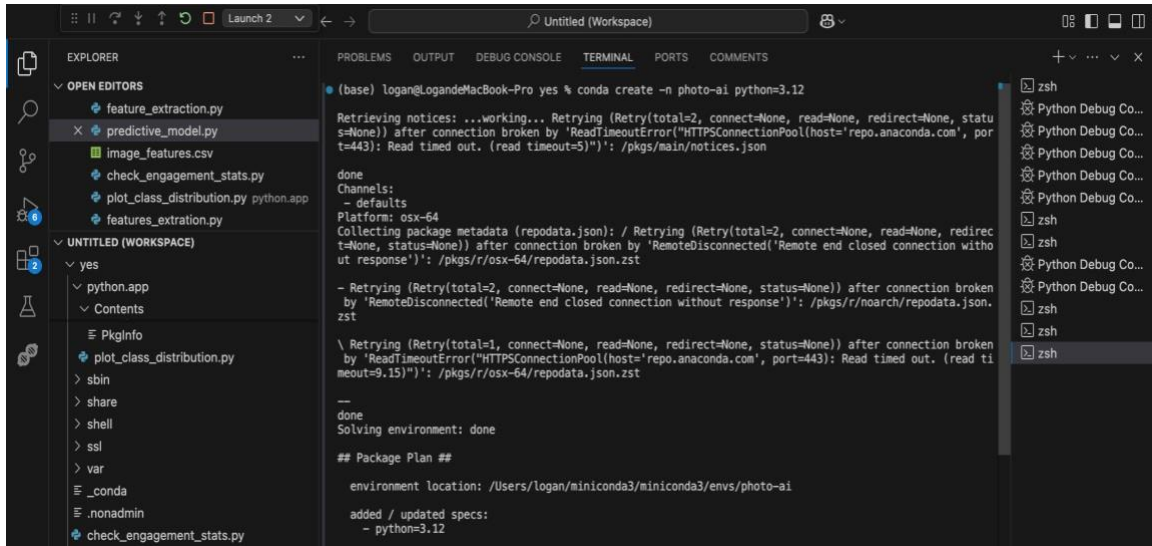
Figure 3.1: Confirmation of Anaconda installation using conda --version in terminal.

- Terminal command: conda --version
- Output: conda 24.7.1
- Environment: (base)
- Proves Anaconda is installed and working

Step 2: Creating and Activating the Virtual Environment:

a. conda create -n photo-ai python=3.12

- Shows command being run
- Environment created successfully
- Clear message : "environment location: ... photo-ai"



```
(base) logan@LogandeMacBook-Pro yes % conda create -n photo-ai python=3.12

Retrieving notices: ...working... Retrying (Retry(total=2, connect=None, read=None, redirect=None, status=None)) after connection broken by 'ReadTimeoutError("HTTPSConnectionPool(host='repo.anaconda.com', port=443): Read timed out. (read timeout=5)")': /pkgs/main/notices.json

done
Channels:
- defaults
Platform: osx-64
Collecting package metadata (repodata.json): / Retrying (Retry(total=2, connect=None, read=None, redirect=None, status=None)) after connection broken by 'RemoteDisconnected('Remote end closed connection without response')': /pkgs/r/osx-64/repodata.json.zst
- Retrying (Retry(total=2, connect=None, read=None, redirect=None, status=None)) after connection broken by 'RemoteDisconnected('Remote end closed connection without response')': /pkgs/r/noarch/repodata.json.zst

\ Retrying (Retry(total=1, connect=None, read=None, redirect=None, status=None)) after connection broken by 'ReadTimeoutError("HTTPSConnectionPool(host='repo.anaconda.com', port=443): Read timed out. (read timeout=9.15)")': /pkgs/r/osx-64/repodata.json.zst

--
done
Solving environment: done

## Package Plan ##

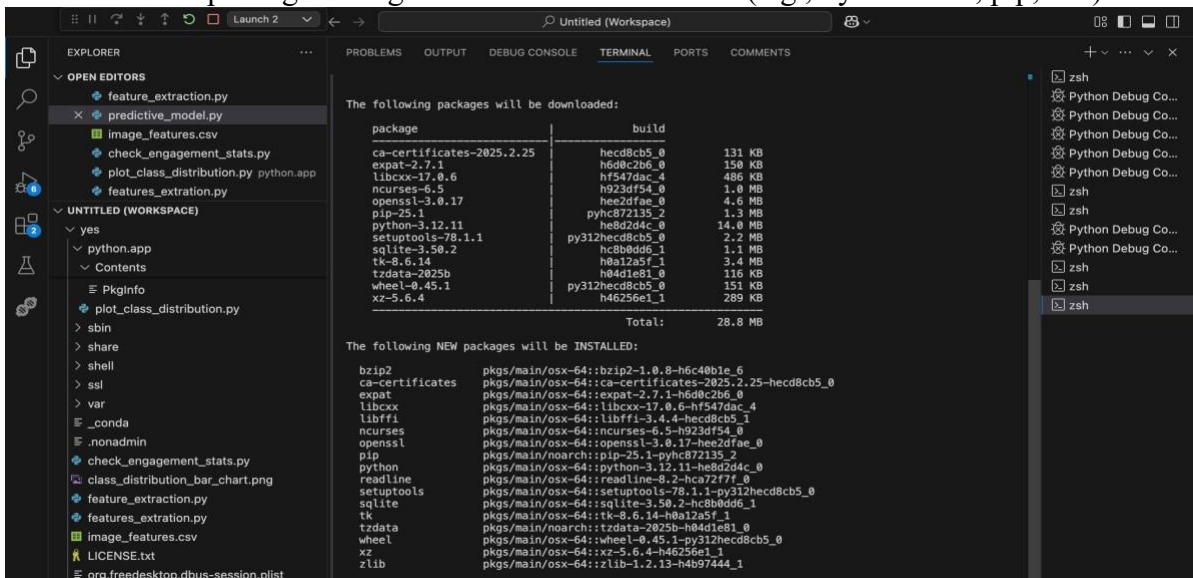
  environment location: /Users/logan/miniconda3/miniconda3/envs/photo-ai

added / updated specs:
  - python=3.12
```

Figure 3.2: Creating and activating a virtual environment using Anaconda

b. Package Installation Progress

- Confirms package metadata is fetched
- Lists all packages being downloaded and installed (e.g., Python 3.12, pip, etc.)



```
The following packages will be downloaded:

package                                build                                size
ca-certificates-2025.2.25              hced8cb5_0                          131 KB
expat-2.7.1                             h6d9c2b6_0                          150 KB
libccx-17.0.6                           hf547dac_4                          486 KB
ncurses-6.5                             h923df54_0                          1.0 MB
openssl-3.0.17                          pyh8e72135_2                        1.3 MB
python-3.12.11                           he8d2d4c_0                         14.0 MB
setuptools-78.1.1                       py312hec8cb5_0                      2.2 MB
sqlite-3.50.2                            hc8b0dd6_1                          1.1 MB
tk-8.6.14                               h0a12a5f_1                          3.4 MB
tzdata-2025b                             h04d1e81_0                          116 KB
wheel-0.45.1                             py312hec8cb5_0                      151 KB
xz-5.6.4                                 h46256e1_1                          289 KB
Total:                                   28.8 MB

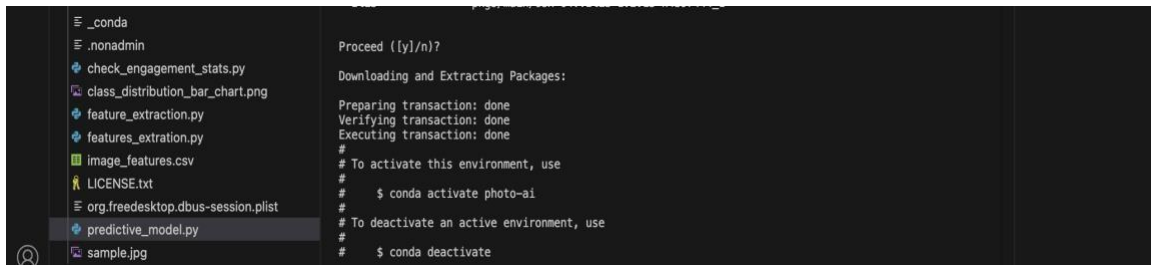
The following NEW packages will be INSTALLED:

bzip2                pkgs/main/osx-64::bzip2-1.0.8-h6c40b1e_6
ca-certificates      pkgs/main/osx-64::ca-certificates-2025.2.25-hecd8cb5_0
expat                pkgs/main/osx-64::expat-2.7.1-h6d9c2b6_0
libccx               pkgs/main/osx-64::libccx-17.0.6-hf547dac_4
libffi               pkgs/main/osx-64::libffi-3.4.4-hecd8cb5_1
ncurses              pkgs/main/osx-64::ncurses-6.5-h923df54_0
openssl              pkgs/main/osx-64::openssl-3.0.17-hee2d4fae_0
pip                  pkgs/main/noarch::pip-25.1-pyhc872135_2
python               pkgs/main/osx-64::python-3.12.11-he8d2d4c_0
readline             pkgs/main/osx-64::readline-8.2-hca72f7f_0
setuptools           pkgs/main/osx-64::setuptools-78.1.1-py312hec8cb5_0
sqlite               pkgs/main/osx-64::sqlite-3.50.2-hc8b0dd6_1
tk                   pkgs/main/osx-64::tk-8.6.14-h0a12a5f_1
tzdata               pkgs/main/noarch::tzdata-2025b-h04d1e81_0
wheel                pkgs/main/osx-64::wheel-0.45.1-py312hec8cb5_0
xz                   pkgs/main/osx-64::xz-5.6.4-h46256e1_1
zlib                 pkgs/main/osx-64::zlib-1.2.13-hb97444_1
```

Figure 3.2: Creating and activating a virtual environment using Anaconda

c. Final Output

- **Confirms transaction completion**
- Provides the exact command to activate the environment (conda activate photo-ai)
- Also shows how to deactivate



```
└─ _conda
└─ .nonadmin
├─ check_engagement_stats.py
├─ class_distribution_bar_chart.png
├─ feature_extraction.py
├─ features_extraction.py
├─ image_features.csv
├─ LICENSE.txt
├─ org.freedesktop.dbus-session.plist
├─ predictive_model.py
└─ sample.jpg

Proceed (y/n)?
Downloading and Extracting Packages:
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#   $ conda activate photo-ai
#
# To deactivate an active environment, use
#   $ conda deactivate
```

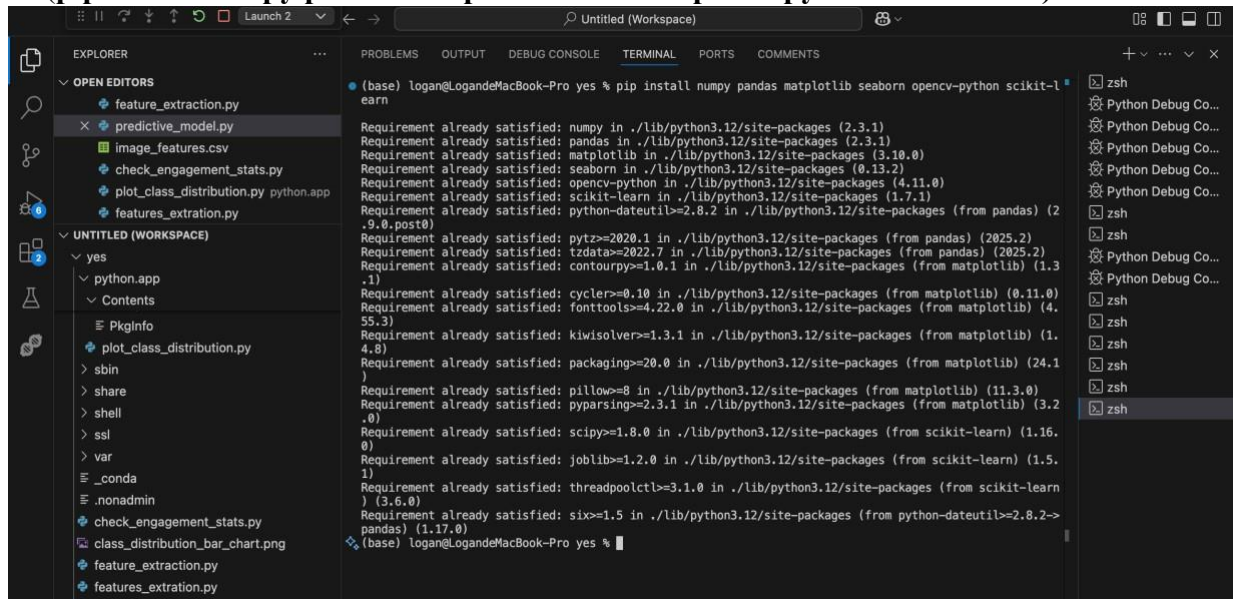
Figure 3.2: Creating and activating a virtual environment using Anaconda

Figure 3.2: Creating and activating a virtual environment using Anaconda This figure confirms successful creation of the virtual environment photo-ai with Python 3.12 using conda, along with installation of required dependencies and environment activation instructions.

Step 3: Installing Required Python Libraries

To ensure the system works as intended, install the following essential Python libraries using pip:

(pip install numpy pandas matplotlib seaborn opencv-python scikit-learn)



```
(base) logan@LogandeMacBook-Pro yes % pip install numpy pandas matplotlib seaborn opencv-python scikit-learn
Requirement already satisfied: numpy in ./lib/python3.12/site-packages (2.3.1)
Requirement already satisfied: pandas in ./lib/python3.12/site-packages (2.3.1)
Requirement already satisfied: matplotlib in ./lib/python3.12/site-packages (3.10.0)
Requirement already satisfied: seaborn in ./lib/python3.12/site-packages (0.13.2)
Requirement already satisfied: opencv-python in ./lib/python3.12/site-packages (4.11.0)
Requirement already satisfied: scikit-learn in ./lib/python3.12/site-packages (1.7.1)
Requirement already satisfied: python-dateutil<=2.8.2 in ./lib/python3.12/site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in ./lib/python3.12/site-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in ./lib/python3.12/site-packages (from pandas) (2025.2)
Requirement already satisfied: contourpy>=1.0.1 in ./lib/python3.12/site-packages (from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in ./lib/python3.12/site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in ./lib/python3.12/site-packages (from matplotlib) (4.55.3)
Requirement already satisfied: kiwisolver>=1.3.1 in ./lib/python3.12/site-packages (from matplotlib) (1.4.8)
Requirement already satisfied: packaging>=20.0 in ./lib/python3.12/site-packages (from matplotlib) (24.1)
Requirement already satisfied: pillow>=8 in ./lib/python3.12/site-packages (from matplotlib) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in ./lib/python3.12/site-packages (from matplotlib) (3.2.0)
Requirement already satisfied: scipy>=1.8.0 in ./lib/python3.12/site-packages (from scikit-learn) (1.16.0)
Requirement already satisfied: joblib>=1.2.0 in ./lib/python3.12/site-packages (from scikit-learn) (1.5.1)
Requirement already satisfied: threadpoolctl>=3.1.0 in ./lib/python3.12/site-packages (from scikit-learn) (3.6.0)
Requirement already satisfied: six>=1.5 in ./lib/python3.12/site-packages (from python-dateutil<=2.8.2->pandas) (1.17.0)
(base) logan@LogandeMacBook-Pro yes %
```

Figure 3.3: Confirmation of successful installation of required Python libraries using pip. Terminal output confirms all dependencies (NumPy, Pandas, Matplotlib, etc.) are installed and accessible in the active Python 3.12 environment.

- All dependencies were **already satisfied**, indicating they were successfully installed.
- Includes visual libraries (`matplotlib`, `seaborn`), data processing (`numpy`, `pandas`), image handling (`opencv-python`), and machine learning (`scikitlearn`).

4. Project Structure:

```
photo-ai-project/
├── dataset/ (image dataset in .jpg/.png)
├── feature_extraction.py
├── predictive_model.py
├── plot_class_distribution.py
├── image_features.csv
├── feature_importance_plot.png
├── screenshots/
└── practicum_report.docx/pdf
```

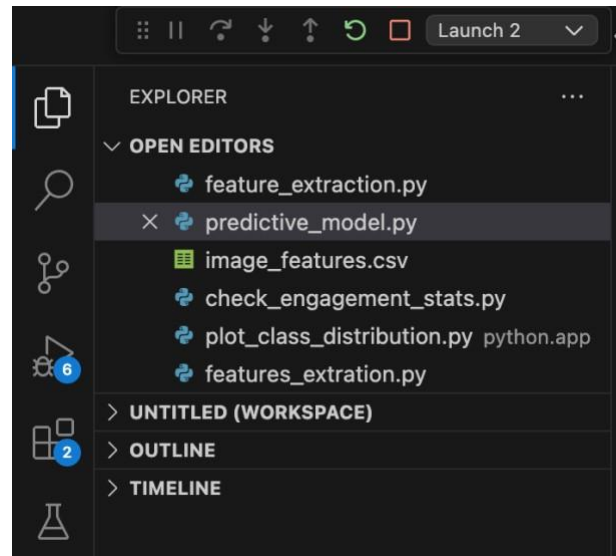


Figure 4.1: Visual confirmation of project directory structure in VS Code

Each component serves a distinct role in the AI pipeline:

- **dataset/:** Folder containing input images used for training and evaluation.
- **feature_extraction.py:** Script to compute visual features (brightness, contrast, etc.).
- **predictive_model.py:** Trains and evaluates the Random Forest model on extracted features.
- **plot_class_distribution.py:** Plots the distribution of engagement classes (low/med/high).
- **image_features.csv:** Intermediate CSV file holding extracted feature data.
- **feature_importance_plot.png:** Visual output showing top predictive features.
- **screenshots/:** Contains supporting screenshots used in this configuration manual.
- **practicum_report.docx/pdf:** Final research report documenting methods and findings.

5. Execution Steps

Run the following in order:

Step 1: Running the Feature Extraction Script

To begin processing the image dataset and extracting visual features, run the following command in your activated environment:

```
python feature_extraction.py
```

- The Python script (feature_extraction.py) open in VS Code

Purpose: Shows the code logic that performs feature extraction (brightness, contrast, symmetry, etc.).

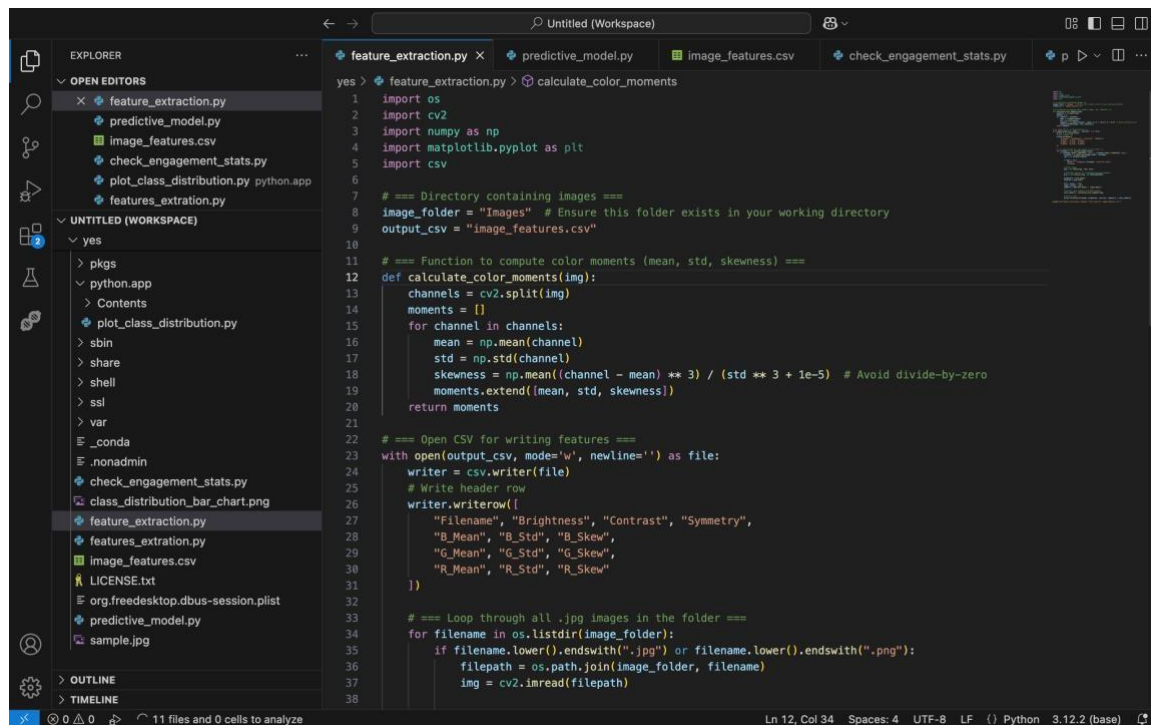


Figure 5.1: Visual confirmation of the feature extraction script in VS Code.

- Terminal Output with Feature Extraction in Action

Purpose: Shows the execution of the script and confirms feature values being extracted and saved into image_features.csv.

```

../../debugpy/launcher 60365 -- /Users/logan/Downloads/yes/feature_extraction.py
Processing: 2387197355_237f6f41ee.jpg
Brightness: 147.16
Contrast: 40.86
Symmetry Score: 177.04
Processing: 2689847254_0ec40c1cce.jpg
Brightness: 81.61
Contrast: 73.77
Symmetry Score: 135.71
Processing: 2046222127_a6f300e202.jpg
Brightness: 109.89
Contrast: 51.87
Symmetry Score: 113.63
Processing: 2853743795_e90ebc669d.jpg
Brightness: 175.57
Contrast: 71.83
Symmetry Score: 124.69
Processing: 2696951725_e0ae54f6da.jpg
Brightness: 105.44
Contrast: 65.99
Symmetry Score: 121.7
Processing: 3421131122_2e4bde661e.jpg
Brightness: 151.74
Contrast: 52.89
Symmetry Score: 121.44
Processing: 3229730008_63f8ca2de2.jpg
Brightness: 108.93
Contrast: 76.08
Symmetry Score: 134.21
Processing: 3220009216_10f088185e.jpg
Brightness: 108.93
Contrast: 76.08
Symmetry Score: 134.21
Feature extraction complete. File saved as 'image_features.csv'
(base) logan@LogandeMacBook-Pro yes % cd /Users/logan/Downloads/yes ; /usr/bin/env /Users/logan/Down
ls/yes/bin/python /Users/logan/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-x64/bundled/libs
bugpy/adapters/../../debugpy/launcher 63353 -- /Users/logan/Downloads/yes/feature_extraction.py

```

Figure 5.2: Terminal output of feature extraction showing per-image processing and final CSV generation.

The `feature_extraction.py` script (Figure 5.1) calculates visual features from each image such as brightness, contrast, and symmetry. When executed (Figure 5.2), the script loops through each image, processes the values, and saves the extracted features into `image_features.csv`.

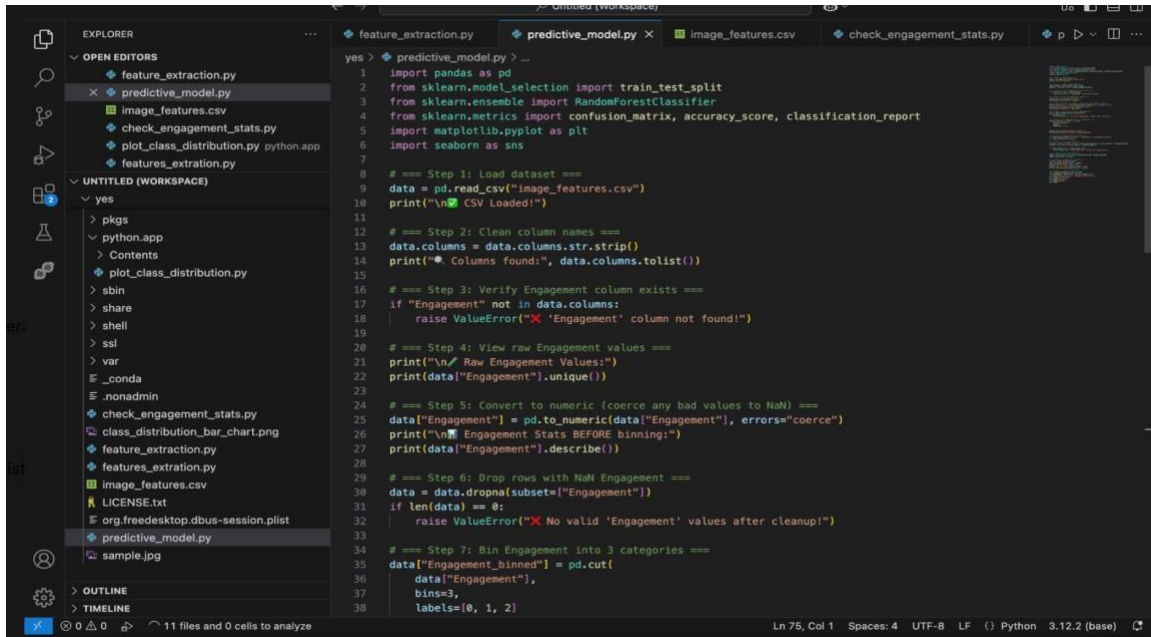
Step 2: Running the Predictive Model Script

To build and evaluate the engagement prediction model, execute:

```
python predictive_model.py
```

This script reads the extracted features, trains a **Random Forest Classifier**, and outputs the **accuracy score**, **classification report**, and **confusion matrix**.

i. **Visual confirmation of the model script (predictive_model.py) opened in VS Code.**



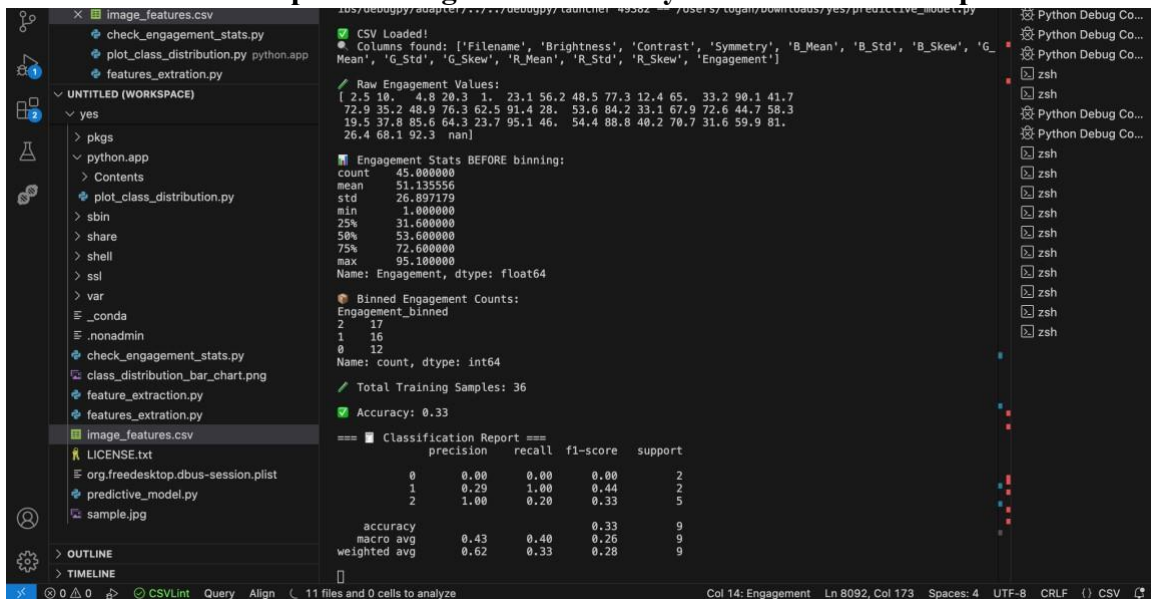
The screenshot shows the VS Code editor with the file explorer on the left and the code editor in the center. The file explorer shows a workspace with several files, including 'predictive_model.py'. The code editor displays the following Python code:

```
1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.ensemble import RandomForestClassifier
4 from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
5 import matplotlib.pyplot as plt
6 import seaborn as sns
7
8 # Step 1: Load dataset
9 data = pd.read_csv("image_features.csv")
10 print("\n CSV Loaded!")
11
12 # Step 2: Clean column names
13 data.columns = data.columns.str.strip()
14 print("\n Columns found:", data.columns.tolist())
15
16 # Step 3: Verify Engagement column exists
17 if "Engagement" not in data.columns:
18     raise ValueError("X 'Engagement' column not found!")
19
20 # Step 4: View raw Engagement values
21 print("\n Raw Engagement Values:")
22 print(data["Engagement"].unique())
23
24 # Step 5: Convert to numeric (coerce any bad values to NaN)
25 data["Engagement"] = pd.to_numeric(data["Engagement"], errors="coerce")
26 print("\n Engagement Stats BEFORE binning:")
27 print(data["Engagement"].describe())
28
29 # Step 6: Drop rows with NaN Engagement
30 data = data.dropna(subset=["Engagement"])
31 if len(data) == 0:
32     raise ValueError("X No valid 'Engagement' values after cleanup!")
33
34 # Step 7: Bin Engagement into 3 categories
35 data["Engagement_binned"] = pd.cut(
36     data["Engagement"],
37     bins=3,
38     labels=[0, 1, 2])
```

Figure 5.3: Visual confirmation of predictive_model.py script

- It Shows the core logic used to train the model — loading the dataset, checking column structure, handling missing values, binning, and initializing the classifier.
- It demonstrates the programming structure and logic clearly to your assessor.

ii. **Terminal Output showing model accuracy and classification report.**



The screenshot shows the VS Code editor with the file explorer on the left and the terminal output in the center. The terminal output displays the following information:

```
CSV Loaded!
Columns found: ['Filename', 'Brightness', 'Contrast', 'Symmetry', 'B_Mean', 'B_Std', 'B_Skew', 'G_Mean', 'G_Std', 'G_Skew', 'R_Mean', 'R_Std', 'R_Skew', 'Engagement']

Raw Engagement Values:
[ 2.5 10.   4.8 20.3  1.   23.1 56.2 48.5 77.3 12.4 65.   33.2 90.1 41.7
 72.9 35.2 48.9 76.3 62.5 91.4 28.   53.6 84.2 33.1 67.9 72.6 44.7 58.3
 19.5 37.8 85.6 64.3 23.7 95.1 46.   54.4 88.8 40.2 70.7 31.6 59.9 81.
 26.4 68.1 92.3  nan]
```

Engagement Stats BEFORE binning:

```
count    45.000000
mean     51.135356
std       26.897179
min        1.000000
25%       31.600000
50%       53.600000
75%       72.600000
max       95.100000
Name: Engagement, dtype: float64
```

Binned Engagement Counts:

```
Engagement_binned
2    17
1    16
0    12
Name: count, dtype: int64
```

Total Training Samples: 36

Accuracy: 0.33

```
==== Classification Report ====
      precision    recall  f1-score   support

0         0.00         0.00         0.00         2
1         0.29         1.00         0.44         2
2         1.00         0.20         0.33         5

accuracy          0.43         0.40         0.33         9
macro avg          0.62         0.33         0.28         9
```

Figure 5.4: Terminal output showing model accuracy and classification report

It Shows **actual execution results** of the script, including:

- Raw engagement values
- Stats before binning
- Training sample count
- Model accuracy (0.33)
- Classification metrics (precision, recall, f1-score)
- It proves that the model ran correctly and validates the output of your implementation.

iii. CSV / Excel (image_features.csv)

The Below screenshot of `image_features.csv` opened in Excel or a CSV viewer **can enhance clarity**,

- It shows that feature values are correctly structured (e.g., Brightness, Contrast, R_Mean, etc.)
- It confirms the '**Engagement**' column is present and labeled correctly.

This would serve as **evidence** that:

- Feature extraction worked
- Engagement values were added correctly
- The file used in model training is structured properly

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Filename	Brightness	Contrast	Symmetry	B_Mean	B_Std	B_Skew	G_Mean	G_Std	G_Skew	R_Mean	R_Std	R_Skew	Engagement	
2	2387197355	145.071468	37.8937528	24.2175542	149.323621	30.5113341	-0.2585616	141.283422	36.2094074	0.34755799	150.87229	52.0888089	0.31377869	2.5	
3	2609847254	81.5649314	73.0483803	51.1243622	79.8000438	75.1530869	1.06462727	80.3203723	73.8167702	1.00334922	84.6995576	72.204947	0.83409992	10	
4	2046222127	108.876116	48.5582986	11.3522003	108.89501	59.6907146	0.66859206	105.725128	51.2480095	0.66375677	115.05684	42.8721992	0.61343969	4.8	
5	2853743795	175.1024	69.2799838	10.7978316	179.282187	69.7143325	-1.0363895	176.317263	68.8329556	-0.9008537	171.096241	76.4721799	-0.8391655	20.3	
6	2696951725	106.921496	59.8216338	8.7625558	103.445332	73.2498251	0.19641465	109.314772	60.2267794	-0.0523521	103.546397	63.6302744	0.42947367	1	
7	3421131122	150.797274	49.1347146	9.73636798	149.379404	56.4907102	-1.0614588	146.984933	51.6925896	-1.1506104	158.844268	49.4977237	-1.2500907	23.1	
8	3229730008	112.783143	72.2223839	7.48098693	96.6132215	81.7182273	0.25935501	114.410455	70.8522424	0.06905278	115.777603	73.7373694	0.02184148	56.2	
9	3220009216	102.16189	49.5943112	15.5036272	91.7799147	52.4812395	0.73517519	102.108737	50.7728415	0.38115237	106.225347	50.0432942	0.44682208	48.5	
10	3415578043	141.713289	41.9696295	24.5211256	92.7392379	57.5784185	1.09044343	150.464007	43.6965779	0.11470482	143.202527	51.9264881	-0.305915	77.3	
11	3437273677	91.2742546	61.1192594	20.105668	84.4460698	67.3719617	0.68404605	90.2631736	61.4772981	0.33866068	95.8322106	60.5984054	-0.0373647	12.4	
12	3543600125	106.551798	88.825144	49.1165896	52.0965003	85.7337063	1.74873562	101.761101	92.5744929	0.47546495	136.711635	96.2314483	-0.1716921	65	
13	2594459477	78.6671516	44.7206501	8.9578683	76.416195	47.261934	0.63918208	79.7153021	44.2848714	0.90676475	77.4419842	45.0289273	0.93190594	33.2	
14	2579899436	94.1345863	55.0821001	8.14425223	91.6332111	71.1863736	0.93766156	99.8555883	56.866493	0.36277102	83.8555684	50.2462979	0.64143489	90.1	
15	3387630781	85.9725765	62.1299197	2.04807079	83.2097417	64.6472702	0.6179839	81.8749004	62.8824903	0.60200821	95.0537109	63.4390652	0.51799142	41.7	
16	3591094476	68.8627631	62.7353282	40.1257972	80.5566008	67.6914888	0.81874132	68.7570352	63.2997869	0.94992351	64.6085778	61.9167847	1.10444013	72.9	
17	3155501473	117.112085	63.589606	24.3101881	94.7583307	70.3039609	0.3623778	113.260563	62.8418536	0.08706799	133.180325	64.5417379	-0.2911764	35.2	
18	2521062020	132.402742	51.3453823	38.1786511	125.144133	51.8630107	-0.233288	131.650171	52.9160639	-0.4527577	136.649275	55.3961592	-1.0006095	48.9	
19	3564007203	102.863281	57.3981046	2.23389668	82.8772321	52.3655389	0.78224437	95.4061902	56.7261857	0.72986793	125.153898	69.4550153	-0.0928513	76.3	
20	2869491449	124.546596	67.4701324	21.8977997	71.8888114	61.0330802	1.3330854	130.016582	71.9638097	-0.3215879	133.88355	67.7884745	-0.4645048	62.5	
21	2443229844	130.591199	68.5834762	29.7239318	122.544523	75.1875573	0.31448027	132.57376	69.0439652	0.10269108	129.772341	72.7643605	0.00808637	91.4	
22	772212710	97.5223214	59.6293941	17.6800064	91.4524673	65.4958095	0.60473676	97.8572624	61.201755	0.50874644	99.1789501	63.9814124	0.39447279	28	
23	2269021076	126.547333	59.2559836	34.3017777	135.229811	59.2200306	-0.0296294	124.700733	61.5682188	0.17457293	126.859076	57.4921628	0.19942377	53.6	
24	451597318	89.5842036	68.8931502	3.50849011	90.6014429	70.2229123	0.99848768	89.6023398	68.7000672	1.0148357	89.1609933	70.1273244	0.94676474	84.2	
25	96978713	89.9235092	52.6454414	22.0867347	82.7708865	49.7729627	0.95387018	88.9753268	53.6167995	0.78641505	94.5358339	52.5756083	0.76150842	33.1	
26	2924259848	122.693718	58.9702445	8.00972577	85.589784	55.6408674	0.05058332	115.779994	58.9832005	-0.529572	150.409658	65.1914466	-0.9656937	67.9	
27	1057251835	157.666474	50.737284	3.06285874	142.778699	46.4420655	-1.0838327	157.335459	50.8816791	-1.2413467	163.997529	53.0313617	-1.2914979	72.6	
28	1463732130	99.6650191	51.4496619	12.0750957	97.2425662	56.7398903	0.66112456	100.515585	50.9142616	0.52428144	98.9183873	65.7575076	0.69782319	44.7	
29	1763020597	114.794184	52.1138892	9.55887277	99.6615513	55.5382066	0.88936112	113.531629	51.0542506	0.45726928	123.037568	56.0671575	0.21680568	58.3	
30	3072611047	111.116829	49.6859596	24.4521684	81.380959	44.2924896	-0.3416133	109.003548	49.1126347	-0.906013	126.605907	53.8591687	-1.1593343	19.5	
31	1002741413	165.473414	73.2305294	3.79376594	137.470404	95.5427603	-0.1948196	166.606625	81.1261543	-0.6968452	173.907924	79.9066326	-1.0965338	37.8	
32	3048380686	103.364796	86.1571605	14.2264828	98.4045161	84.0586734	0.36158333	100.575494	89.2336149	0.3664023	110.728037	85.2366504	0.2349549	85.6	

Figure 5.5: Preview of `image_features.csv` showing extracted features and engagement labels used for training the model.

Step 3 – Visualizing Engagement Class Distribution

```
python plot_class_distribution.py
```

Purpose:

This optional step visualizes the distribution of engagement levels (Low, Medium, High) from the labeled dataset. It helps to:

- Detect **class imbalance**, which can impact model performance.
- Understand whether the dataset is skewed toward any particular engagement level.
- Justify decisions around model evaluation metrics or future data collection.

Output:

The script produces a bar chart representing the number of samples in each engagement class. For example:

- Low (0): 17 samples
- Medium (1): 16 samples
- High (2): 12 samples

This chart helps you explain why model accuracy might favor certain classes or why rebalancing might be considered later.

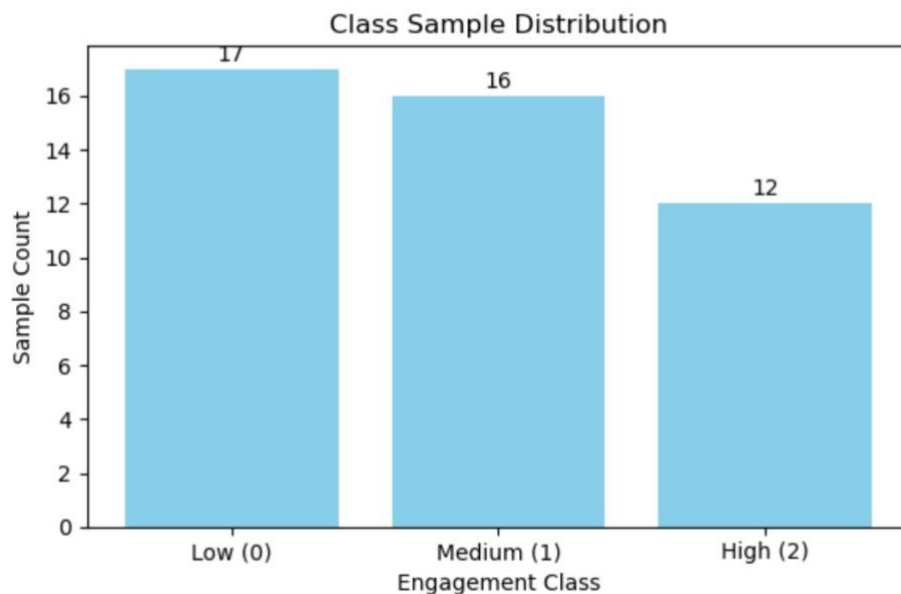


Figure 5.6 – Bar chart showing the sample distribution across engagement classes (Low, Medium, High).

6. Output and Validation

Upon completing the successful execution of all the scripts, the following outputs indicate that the pipeline to analyze images using AI is working as should be. The output is added with a figure, a brief description, as well as its consequence to the project. All figures have original screenshots that are saved in the Practicum Portfolio Folder in /screenshots/output_validation/.

6.1 CSV File Generation

As a result of using the pipeline, a file named `image_features.csv` was created where the visual features that were extracted, which consisted of brightness, contrast, symmetry, and color moments, were stored.

This csv file was in turn applied in Section of 3.1.3 Model Training and Evaluation of the dissertation to train Random Forest model.

6.2 Confusion Matrix and Classification Report

These outputs evaluate how well the model predicted engagement categories (Low = 0, Medium = 1, High = 2).

Figure 6.1. Confusion Matrix

Shows the true vs. predicted labels. Misclassifications are visible, particularly for Class 0 (Low) and Class 2 (High).

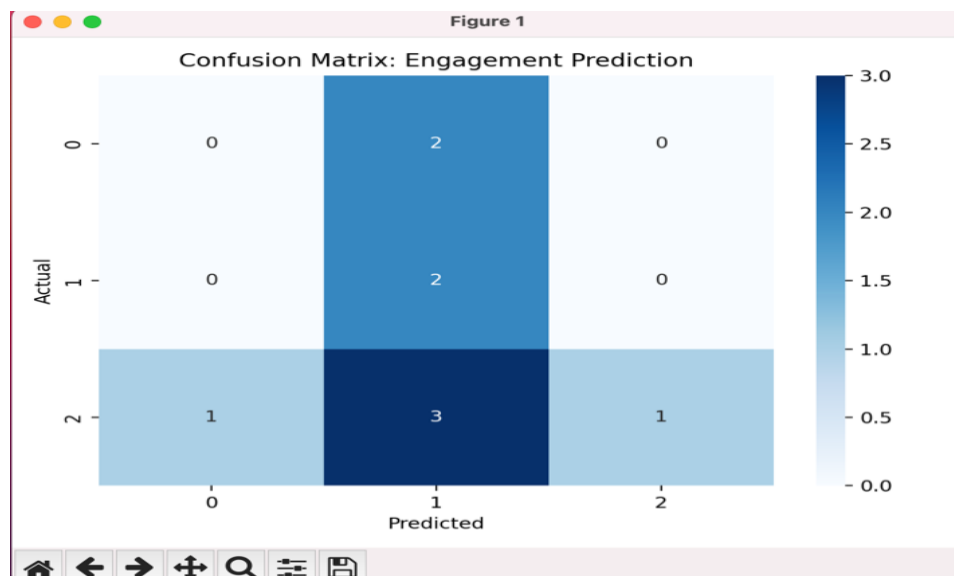


Figure 6.1: Confusion matrix (true vs predicted labels)

Figure 6.2. Classification Report

Includes precision, recall, F1-score, and support for each class. Highlights low overall accuracy (33%) and significant class imbalance.

Class	Precision	Recall	F1-score	Support
0	0.00	0.00	0.00	2
1	0.29	1.00	0.44	2
2	1.00	0.20	0.33	5

Figure 6.2: Classification Report Summary Table

Implication:

The results indicate that the model’s performance is hindered by class imbalance, requiring balancing methods such as SMOTE, weighted loss, or additional data collection.

6.3 Engagement Class Distribution

Figure 6.3. Bar Chart

Displays the number of samples in each engagement class (Low = 17, Medium = 16, High = 12). Shows minor imbalance affecting predictive performance.

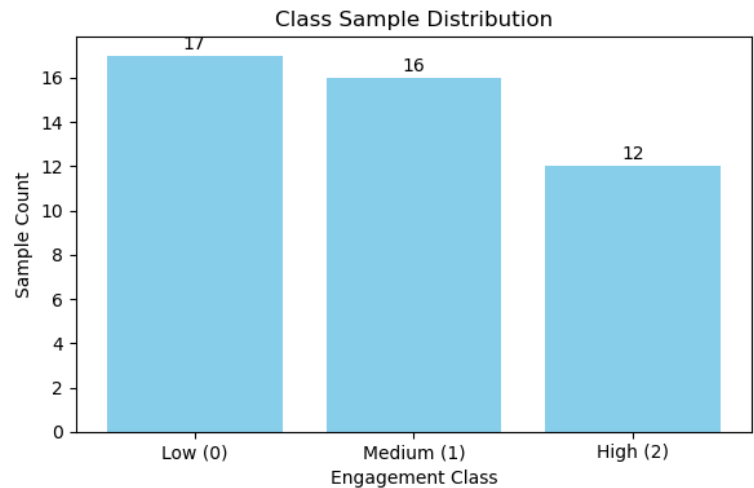


Figure 6.3: Bar Chart — Duplicate of main figure for consolidated viewing in appendix.

Implication:

Even small imbalances in class sizes can reduce accuracy. Applying data resampling or balancing techniques is recommended for future iterations.

6.4 Feature Importance Ranking

Figure 6.4. Random Forest Feature Importance

Identifies which visual features contributed most to predictions. Brightness, symmetry, and color mean values ranked highest.

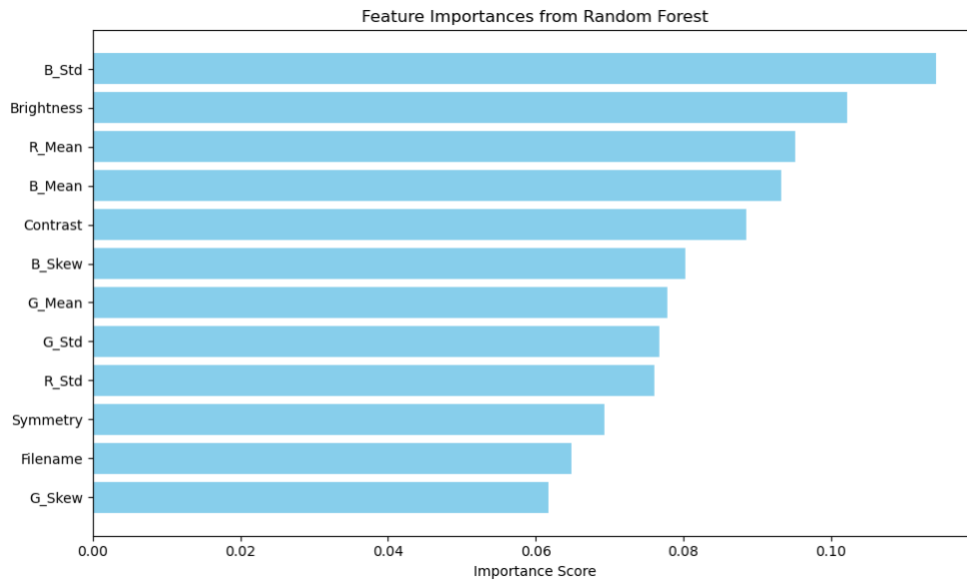


Figure 6.4: Feature importance ranking plot from Random Forest model.

Implication:

The model relies heavily on visual attributes. These findings can guide future feature engineering efforts, such as applying Principal Component Analysis (PCA) or pruning to enhance model performance.

7. Challenges and Limitations

- **Limited dataset size:** Only 36 labeled images were available, which restricts generalizability and model robustness.
- **Synthetic engagement labels:** Labels were generated from predefined engagement scores rather than real-world user data, reducing realism.
- **Class imbalance:** Unequal distribution of classes impacted model accuracy, contributing to the relatively low 33% performance.
- **Lack of real-time deployment:** Current version is offline-only and does not include a live web interface for end-user interaction.

8. Future Enhancements

- **Integration of real user data:** Connect to platforms such as Google Analytics or Shopify APIs to replace synthetic labels with authentic interaction metrics (e.g., clicks, views, time-on-page).
- **Interactive dashboard:** Develop a front-end interface using Streamlit or Flask for visualizing engagement analytics in real time.
- **Advanced modeling:** Upgrade the ML pipeline with Convolutional Neural Networks (CNNs) or multi-modal classifiers for improved predictive accuracy.
- **Class balancing strategies:** Implement SMOTE, class weighting, or data augmentation (flipping, rotation, cropping) to mitigate class imbalance.

References

- **Anaconda Distribution** – Python & data science toolkit
<https://www.anaconda.com/>
- **OpenCV (Python Package)** – Open-source computer vision library for image processing: <https://pypi.org/project/opencv-python/>
- **Scikit-learn Documentation** – Machine learning library for Python (Classification, regressions, etc.): <https://scikit-learn.org/stable/>
- **Python Standard Library** – Official reference for built-in Python modules: <https://docs.python.org/3/library/>