

## MALARIA DETECTION WITH MACHINE LEARNING

—

Martha Sharon Murugi Maina

15th May, 2024

## Overview

The following is an overview of the malaria detection assignment with machine Learning

## Tools & Software

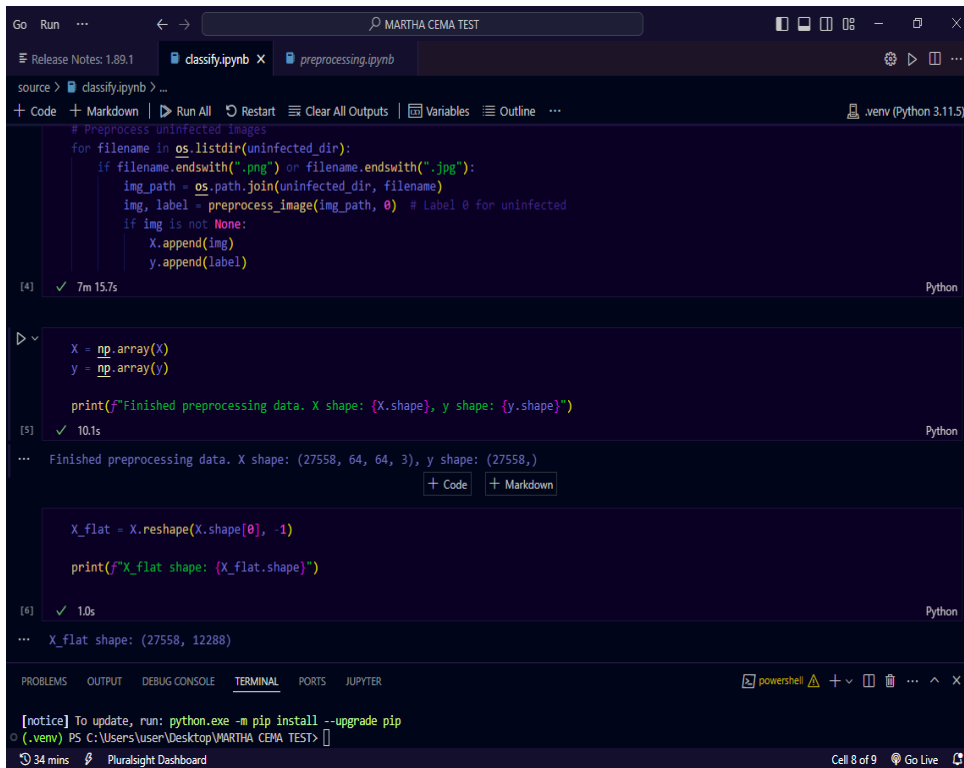
1. Visual Studio Code
2. Anaconda
3. Google Sheets
4. Google Docs
5. Python
6. Scikit-Learn
7. Pandas
8. Numpy
9. Pillow

## Data Collection & Analysis

The dataset was downloaded from Tensorflow Datasets. I analyzed it using Google Sheets. There were no duplicates or white spaces. However, there were two empty cells which I cleared.

## Data Pre-processing

I used Numpy and Pillow to read, resize and normalize the images. The images were in different subfolders and so preprocessing was for each individual folder.



The screenshot shows a Jupyter Notebook with two files: `classify.ipynb` and `preprocessing.ipynb`. The `preprocessing.ipynb` file is active, displaying the following code and output:

```
# Preprocess uninfected images
for filename in os.listdir(uninfected_dir):
    if filename.endswith('.png') or filename.endswith('.jpg'):
        img_path = os.path.join(uninfected_dir, filename)
        img, label = preprocess_image(img_path, 0) # Label 0 for uninfected
        if img is not None:
            X.append(img)
            y.append(label)
```

Output [4]: 7m 15.7s

```
X = np.array(X)
y = np.array(y)

print(f"Finished preprocessing data. X shape: {X.shape}, y shape: {y.shape}")
```

Output [5]: 10.1s

Finished preprocessing data. X shape: (27558, 64, 64, 3), y shape: (27558,)

```
X_flat = X.reshape(X.shape[0], -1)

print(f"X_flat shape: {X_flat.shape}")
```

Output [6]: 1.0s

X\_flat shape: (27558, 12288)

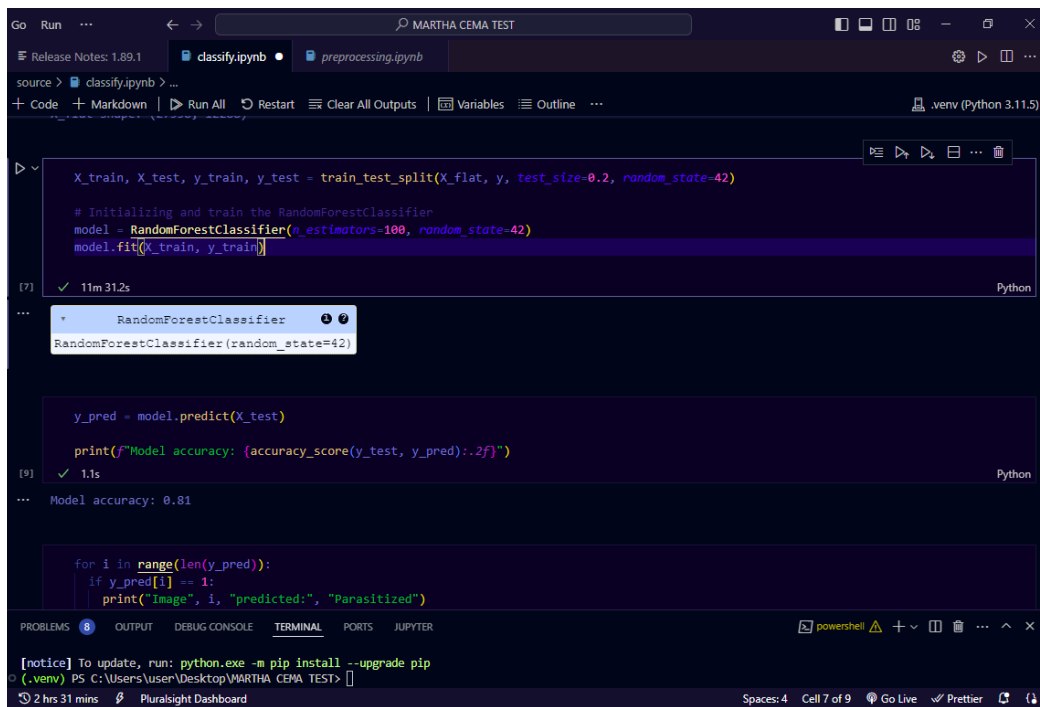
The bottom of the notebook shows a terminal window with the following text:

```
[notice] To update, run: python.exe -m pip install --upgrade pip
(.venv) PS C:\Users\User\Desktop\WARTH CEMA TEST>
```

The bottom status bar indicates "Cell 8 of 9" and "Go Live".

# Classification Model & Training

## I. RandomForest Classifier



```
Go Run ... MARTHA CEMA TEST
Release Notes: 1.89.1
classify.ipynb preprocessing.ipynb
source > classify.ipynb > ...
+ Code + Markdown | Run All Restart Clear All Outputs Variables Outline ...
.venv (Python 3.11.5)

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

# Initializing and train the RandomForestClassifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

[7] ✓ 11m 31.2s Python

RandomForestClassifier
RandomForestClassifier(random_state=42)

y_pred = model.predict(X_test)

print(f"Model accuracy: {accuracy_score(y_test, y_pred):.2f}")

[9] ✓ 1.1s Python

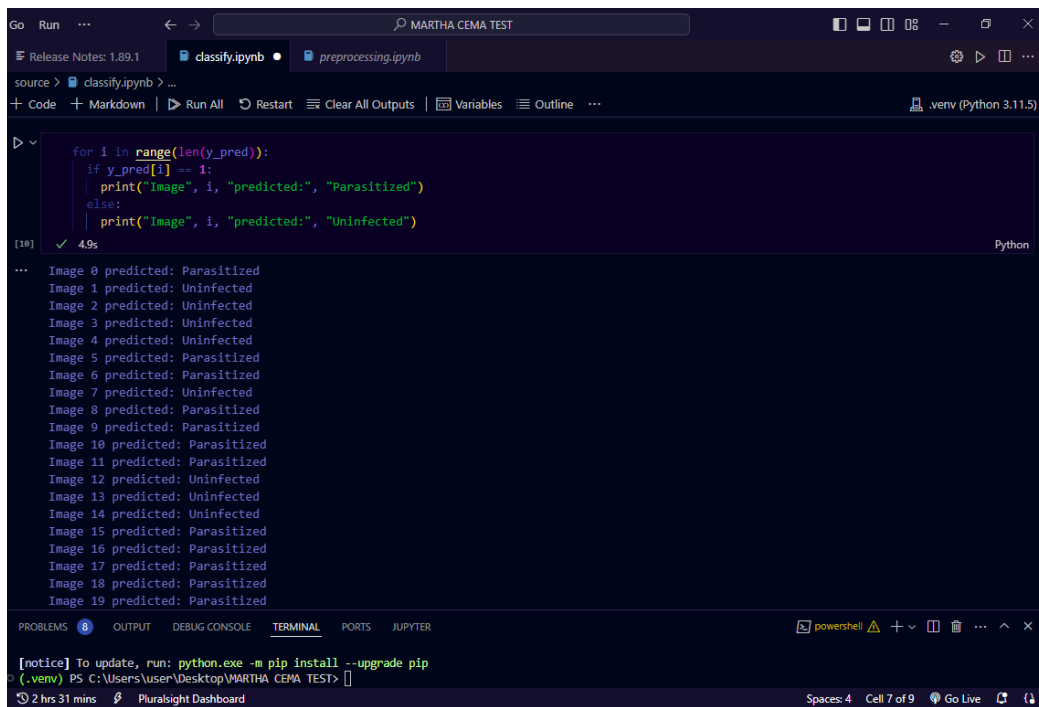
Model accuracy: 0.81

for i in range(len(y_pred)):
    if y_pred[i] == 1:
        print("Image", i, "predicted:", "Parasitized")

PROBLEMS 8 OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER
[notice] To update, run: python.exe -m pip install --upgrade pip
(.venv) PS C:\Users\User\Desktop\WARTH CEMA TEST>
2 hrs 31 mins Pluralsight Dashboard Spaces: 4 Cell 7 of 9 Go Live Prettier
```

I used RandomForest Classifier for its ease of use and it served as a strong baseline model. Upon training, the model achieved 0.81 model accuracy.

## II. Prediction Outcomes



```
for i in range(len(y_pred)):
    if y_pred[i] == 1:
        print("Image", i, "predicted:", "Parasitized")
    else:
        print("Image", i, "predicted:", "Uninfected")
```

Image 0 predicted: Parasitized  
Image 1 predicted: Uninfected  
Image 2 predicted: Uninfected  
Image 3 predicted: Uninfected  
Image 4 predicted: Uninfected  
Image 5 predicted: Parasitized  
Image 6 predicted: Parasitized  
Image 7 predicted: Uninfected  
Image 8 predicted: Parasitized  
Image 9 predicted: Parasitized  
Image 10 predicted: Parasitized  
Image 11 predicted: Parasitized  
Image 12 predicted: Uninfected  
Image 13 predicted: Uninfected  
Image 14 predicted: Uninfected  
Image 15 predicted: Parasitized  
Image 16 predicted: Parasitized  
Image 17 predicted: Parasitized  
Image 18 predicted: Parasitized  
Image 19 predicted: Parasitized

[notice] To update, run: `python.exe -m pip install --upgrade pip`  
(.venv) PS C:\Users\user\Desktop\MARTHA CEMA TEST>

### III. Suggestions and Recommendations

To reveal specific types of errors a Confusion Matrix would do best as it shows how many instances from each class were predicted correctly (diagonal) and incorrectly classified as other classes (off-diagonal elements).

## Conclusion

This study investigated the application of a machine learning model for malaria parasite detection in blood images. A Random Forest classifier model was trained on a dataset of preprocessed blood cell images labeled as parasitized or uninfected. The trained model achieved an accuracy of 81% on a separate testing set.