## PROJECT REPORT

## ON

**BRAIN TUMOR DETECTION USING PYTHON**

**(CSE IV Semester Mini project) Session: 2023-24**

**Submitted to: Submitted by:**

Mr. Chandradeep Bhatt Garv

(CC-CSE-C2-4th-Sem) Roll. No:2218750

**(**CSE-C2-4th-Sem)

**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING**

**GRAPHIC ERA HILL UNIVERSITY, DEHRADUN**

## CERTIFICATE

**UNIVERSITY ROLL NO. :** 2218750 **CLASS ROLL NO. :**28

I hereby certify that the work which is being presented in the project report entitled “Brain Tumor Detection using Python” in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering of the Graphic Era Hill University, Dehradun has been carried out by me under my own guidance.

# ACKNOWLEDGMENT

Special thanks to the Graphic Era Hill University for providing a conducive academic environment, state-of-the-art facilities, and resources essential for the successful execution of this project. The support staff and administrative personnel have been instrumental in ensuring a smooth academic journey.

I would like to thank particularly my project guide Mr. Chandradeep Bhatt for his patience, support and encouragement throughout the completion of this project and having faith in me. Lastly, I want to thank my family for their unwavering support, understanding, and encouragement. Their love and belief in my abilities have been a constant source of motivation.

**Garv**

Roll No.- 2218750

(CSE-F1-4th-Sem)

Session: 2023-2024 GEHU, Dehradun

**TABLE OF CONTENTS**

**SR NO. TITLE**

1. EXECUTIVE SUMMARY

2. INTRODUCTION

3. OBJECTIVES

4. TECHNOLOGIES USED

5. SYSTEM ARCHITECTURE

6. USER INTERFACE

7. TESTING AND EVALUATION

8. CHALLENGES AND SOLUTION

9. CONCLUSION

10. FUTURE ENHANCEMENTS

11. REFERENCES

**EXECUTIVE SUMMARY**

The provided script is an elegant implementation of a machine learning pipeline designed for brain tumor classification. Leveraging the power of Python libraries such as NumPy, Pandas, and scikit-learn, the script meticulously loads and preprocesses MRI images, transforming them into a suitable format for model training.

It employs both Logistic Regression and Support Vector Classifier (SVC) algorithms to distinguish between four types of brain conditions: glioma, meningioma, no tumor, and pituitary tumor.

The dataset is efficiently split into training and testing sets, ensuring robust model validation. With impressive visualization capabilities, the script not only evaluates the models' performance by printing accuracy scores but also graphically displays the classification results on sample images.

This comprehensive approach highlights the script's proficiency in handling image data, model training, and performance evaluation, culminating in a powerful tool for medical image classification.

**INTRODUCTION**

1. Objective :

- The primary goal of this project is to develop a machine learning pipeline for classifying brain tumors using MRI images.

2. Significance:

- Accurate classification of brain tumors is crucial for medical diagnostics, enabling timely and effective treatment plans.

3. Categories:

- The project focuses on distinguishing between four types of brain conditions: glioma, meningioma, no tumor, and pituitary tumor.

4. Methodology:

- Utilizing advanced Python libraries such as NumPy, Pandas, and scikit-learn, the project involves:

- Data loading and preprocessing

- Model training

- Model evaluation

- Visualization of classification results

5. Models Used:

- Logistic Regression

- Support Vector Classifier (SVC)

- These models are chosen for their robustness and effectiveness in classification tasks.

6. Dataset Handling:

- The dataset is split into training and testing sets to ensure comprehensive evaluation and validation of the models' performance.

7. Relevance:

- The project highlights the potential of machine learning in aiding medical professionals, enhancing the accuracy of brain tumor diagnoses.

8. Report Overview:

- This report details the methodology, implementation, and results of the machine learning approach to brain tumor classification, emphasizing its importance in the medical field.

### OBJECTIVES

1. Develop a Data Preprocessing Pipeline:

- Collect and preprocess MRI images of brain tumors, ensuring they are resized to a uniform dimension for consistent input to machine learning models.

2. Implement Label Encoding:

- Accurately label the images with the appropriate tumor type (glioma, meningioma, no tumor, pituitary tumor) and convert these categorical labels into numerical values for machine learning algorithms.

3. Dataset Segmentation:

- Divide the dataset into training and testing subsets to facilitate model training and robust evaluation.

4. Train Machine Learning Models:

- Employ Logistic Regression and Support Vector Classifier (SVC) to train on the preprocessed dataset, aiming to achieve high classification accuracy for brain tumor detection.

5. Normalize Image Data:

- Apply normalization techniques to the image data, ensuring that pixel values are scaled appropriately to enhance model performance.

6. Apply Dimensionality Reduction:

- Utilize Principal Component Analysis (PCA) to reduce the dimensionality of the dataset, improving computational efficiency and potentially enhancing model performance.

7. Evaluate Model Performance:

- Assess the trained models using accuracy metrics on both the training and testing sets, identifying how well the models generalize to unseen data.

8. Analyze Misclassified Samples:

- Investigate instances of misclassified images to understand the limitations of the models and identify areas for potential improvement.

9. Visualize Results:

- Create visualizations of sample images along with their predicted labels to provide insight into the model's classification capabilities and real-world applicability.

10. Document the Process:

- Thoroughly document each step of the process, from data preparation to model evaluation, to ensure reproducibility and provide a comprehensive guide for future research and development in this area.

#### TECHNOLOGIES USED

1. NumPy:

- For numerical operations and array manipulations.

2. Pandas:

- For data manipulation and analysis.

3. Matplotlib:

- For data visualization and plotting images.

4. scikit-learn:

- For machine learning algorithms, including model training, evaluation, and PCA for dimensionality reduction.

- `train\_test\_split` for splitting the dataset.

- `accuracy\_score` for evaluating model performance.

- `LogisticRegression` for logistic regression model.

- `SVC` for Support Vector Classifier.

- `PCA` for Principal Component Analysis.

5. OpenCV (cv2):

- For image processing tasks such as reading, resizing, and visualizing images.

6. OS:

- For interacting with the file system, such as listing directory contents.

7. Warnings:

- To manage and suppress warning messages during execution.

These technologies collectively enable the development of a comprehensive machine learning pipeline for classifying brain tumor images, ensuring efficient data handling, model training, and performance visualization.

**SYSTEM ARCHITECTURE**

**Data Collection and Preparation:**

* **Input:**
  + MRI images of brain tumors stored in a directory structure.
* **Process:**
  + Use OpenCV (cv2) to read and resize images to a uniform size (200x200 pixels).
  + Assign labels to images based on their respective tumor types.

**Data Organization:**

* **Classes Definition:**
  + Define tumor classes: glioma, meningioma, no tumor, pituitary tumor.
* **Data Storage:**
  + Store images in list X and corresponding labels in list Y.

**Data Preprocessing:**

* **Conversion:**
  + Convert lists X and Y to NumPy arrays.
* **Reshaping:**
  + Reshape X to a 2D array for machine learning models.
* **Normalization:**
  + Normalize image data to have values between 0 and 1.

**Data Splitting:**

* **Process:**
  + Split data into training and testing sets using train\_test\_split from scikit-learn.
  + Allocate 80% of data for training and 20% for testing.

**Dimensionality Reduction:**

* **Principal Component Analysis (PCA):**
  + Apply PCA to reduce the dimensionality of the dataset, retaining 98% of variance.

**Model Training:**

* **Models:**
  + Train two machine learning models:
    - Logistic Regression (LogisticRegression from scikit-learn).
    - Support Vector Classifier (SVC from scikit-learn).
* **Training:**
  + Fit models on the training dataset.

**Model Evaluation:**

* **Accuracy Calculation:**
  + Evaluate models on both training and testing datasets.
  + Calculate and print training and testing accuracy scores.
* **Misclassification Analysis:**
  + Identify and count misclassified samples.

**Visualization:**

* **Display Images:**
  + Use Matplotlib to visualize sample images along with their predicted labels.

**Result Interpretation:**

* **Outputs:**
  + Training and testing accuracy scores for both models.
  + Number of misclassified samples.
  + Visualization of sample test images with predicted labels.

#### 

#### USER INTERFACE

The user interface (UI) described here is designed to interact with the machine learning pipeline for brain tumor classification. While the provided code itself does not have a graphical user interface, it performs the following operations in a command-line environment:

1. Data Directory Configuration:

- Input: Directory paths for training and testing data.

- Description: The code reads MRI images from specified directories using OpenCV (`cv2`) and processes them for classification.

2. \*\*Data Processing and Visualization:

- Preprocessing:

- Resizes images to a uniform dimension (200x200 pixels).

- Normalizes pixel values to range [0, 1].

- Visualization:

- Uses Matplotlib (`plt`) to display sample images and their predicted labels.

3. Model Training and Evaluation:

- Training:

- Trains Logistic Regression and Support Vector Classifier (SVC) models on preprocessed image data.

- Evaluation:

- Outputs model accuracy scores for training and testing datasets.

- Identifies and prints the number of misclassified samples.

4. Prediction and Result Display:

- Prediction:

- Predicts labels for test images using the trained SVC model.

- Display:

- Uses Matplotlib to show test images along with their predicted tumor types.

**User Interaction**

1. Configuration:

- Paths Setup:

- Users need to specify the correct paths to their image datasets in the `path` variable (e.g., `'D:/miniproj/Training'`).

2. Execution:

- Running the Script:

- Execute the script in a command-line interface or terminal.

- The script processes the data, trains the models, evaluates performance, and generates visualizations.

3. Output:

- Command Line:

- Displays accuracy scores and misclassification details in the console.

- Visual Output:

- Shows images with predicted labels in separate Matplotlib plots.

**Future Enhancements**

- Graphical User Interface (GUI):

- Implement a GUI using libraries such as Tkinter or PyQt to allow users to interact with the system more intuitively.

- Interactive Features:

- Add options for users to select directories, adjust model parameters, and view results interactively.

This UI description outlines how users interact with the code in its current form and suggests future enhancements to improve user experience.

#### TESTING AND EVALUATION

The **testing** phase of the machine learning pipeline involves evaluating the performance of the trained models on unseen data. This process ensures that the models generalize well and accurately classify new MRI images of brain tumors.

1. Dataset Preparation:

- Splitting: The dataset is divided into training and testing sets using `train\_test\_split` from scikit-learn. Specifically, 80% of the data is used for training, and 20% is reserved for testing.

- Normalization: The image pixel values in both training and testing sets are normalized to a range of [0, 1] by dividing by 255, ensuring consistent input for the models.

2. Model Training:

- Logistic Regression and SVC: Two classification models—Logistic Regression and Support Vector Classifier (SVC)—are trained on the normalized training data. Hyperparameters for Logistic Regression are set to `C=0.1`, and SVC uses default settings.

3. Model Evaluation:

- Accuracy Scores: The performance of both models is assessed using accuracy scores calculated on both the training and testing sets. Accuracy metrics provide an indication of how well each model classifies the images.

- Misclassification Analysis: The number of misclassified samples is identified by comparing predicted labels against true labels. This analysis helps in understanding the model's performance and limitations.

- Confusion with Predictions: Specific misclassified examples are examined to gain insights into errors and the accuracy of predictions.

**Evaluation**

1. Performance Metrics:

- Training and Testing Accuracy: Accuracy scores are printed for both training and testing datasets for Logistic Regression and SVC models. These metrics indicate how well the models perform in terms of correctly classifying images.

- Misclassified Samples: The number of misclassified samples is reported, providing insight into how often the models make errors. This count helps in evaluating the robustness and reliability of the models.

2. Visualization:

- Sample Predictions: Matplotlib is used to visualize sample images from the testing set. Each image is displayed with its predicted label, allowing for a qualitative assessment of the model's performance. This visual inspection helps in understanding how well the models differentiate between tumor types.

3. Results Interpretation:

- Accuracy Comparison: By comparing accuracy scores and analyzing misclassifications, the relative performance of Logistic Regression and SVC is evaluated. Higher accuracy and fewer misclassifications indicate better model performance.

- Model Assessment: The visual examination of predicted labels and misclassified samples provides a more detailed understanding of model behavior and potential areas for improvement.

Overall, this testing and evaluation process ensures that the machine learning models are assessed comprehensively, providing valuable insights into their accuracy and effectiveness in classifying brain tumor images.

#### CHALLENGES AND SOLUTION

**Challenges**

1. Data Preprocessing:

- Challenge: High computational cost and potential for inconsistent image quality.

- Impact: Affects model accuracy.

2. Model Performance:

- Challenge: Logistic Regression and SVC may require fine-tuning.

- Impact: Potentially low accuracy.

3. Misclassification:

- Challenge: High misclassification rates.

- Impact: Decreased model reliability.

4. Computational Load:

- Challenge: Resource-intensive processing of high-resolution images.

- Impact: Increased training time.

5. Validation:

- Challenge: Lack of explicit validation.

- Impact: Risk of inaccurate performance assessment.

**Solutions**

1. Optimize Preprocessing:

- Solution: Streamline resizing and normalization.

- Impact: Improves data quality.

2. Tune Models:

- Solution: Adjust hyperparameters and apply dimensionality reduction.

- Impact: Enhances accuracy.

3. Analyze Misclassifications:

- Solution: Investigate and address errors.

- Impact: Improves reliability.

4. Manage Resources:

- Solution: Optimize data handling and use lower resolution if possible.

- Impact: Reduces training time.

5. Implement Validation:

- Solution: Include a validation set for accurate performance evaluation.

- Impact: Ensures reliable results.

#### CONCLUSION

The implemented code successfully demonstrates the application of machine learning techniques for classifying brain tumor images. The process involves data collection, preprocessing, model training, and evaluation. Initially, images are read, resized, and normalized, then reshaped for model input. The dataset is split into training and testing sets to assess the performance of Logistic Regression and Support Vector Classifier (SVC) models.

Key findings include that both models were trained and tested, with their performance measured through accuracy scores. The SVC model, in particular, showed promising results with its classification accuracy, though some misclassification was observed. The use of Principal Component Analysis (PCA) to reduce dimensionality helped manage computational load and potentially improved model performance.

In practice, the model's accuracy can be further improved by addressing identified challenges, such as enhancing data preprocessing and fine-tuning model parameters. The code's visualization of predictions on test images provides a practical tool for evaluating model outputs and validating classification results. Overall, this project lays a solid foundation for deploying image classification models in medical imaging contexts.

#### FUTURE ENHANCEMENTS

To advance the image classification system, several key improvements can be applied. Switching to Convolutional Neural Networks (CNNs) could greatly enhance accuracy, as they are better suited for image data. Data augmentation techniques like rotation, scaling, and flipping can diversify the training set, leading to better model robustness. Hyperparameter optimization through grid or random search can fine-tune model performance. Implementing k-fold cross-validation will provide a more reliable measure of model efficacy. Employing dimensionality reduction techniques such as PCA, t-SNE, or UMAP can improve data representation and manage overfitting. Addressing class imbalance with methods like SMOTE will improve performance across underrepresented classes.

Additionally, creating a dedicated validation set and conducting thorough error analysis will help in refining the model. Finally, developing an intuitive user interface for real-time predictions and integrating the model into practical applications will make it more accessible and effective in real-world scenarios.

### REFERENCES

1. NumPy:

- Harris, C. R., Millman, K. J., van der Walt, S. J., et al. (2020). Array programming with NumPy. Nature, 585, 357–362. [DOI: 10.1038/s41586-020-2649-2](https://doi.org/10.1038/s41586-020-2649-2)

2. Pandas:

- McKinney, W. (2010). Data structures for statistical computing in Python. In Proceedings of the 9th Python in Science Conference (pp. 51-56). [URL](https://conference.scipy.org/proceedings/scipy2010/pdfs/mckinney.pdf)

3. Matplotlib:

- Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90-95. [DOI: 10.1109/MCSE.2007.55](https://doi.org/10.1109/MCSE.2007.55)

4. Scikit-Learn:

- Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825-2830. [URL](http://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf)

5. OpenCV:

- Bradski, G., & Kaehler, A. (2008). Learning OpenCV: Computer vision with the OpenCV library. O'Reilly Media, Inc.

6. PCA (Principal Component Analysis):

- Jolliffe, I. T. (2011). Principal Component Analysis (2nd ed.). Springer.

7. Logistic Regression:

- Cox, D. R. (1958). The regression analysis of binary sequences. Journal of the Royal Statistical Society, Series B (Methodological), 20(2), 215-242. [URL](https://www.jstor.org/stable/2983795)

8. Support Vector Machines (SVM):

- Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273-297. [DOI:10.1007/BF00994018](<https://doi.org/10.1007/BF00994018>)

9. Data Handling and Preprocessing:

- Zhang, H., & Chen, X. (2014). Data preprocessing in machine learning: A survey. Computational Intelligence and Neuroscience, 2014, 1-13. [DOI: 10.1155/2014/179641](<https://doi.org/10.1155/2014/179641>)

These references provide a foundation for understanding the technologies and methods used in the code.