PROJECT REPORT

on

NeuraHealth – Multiple Disease Predictor

(CSE 5Semester Mini project on AI / ML in medicine)

2023-2024

Submitted to:

Ms. Preeti Chaudhari

(CC-CSE-A-5-Sem)

Submitted by:

Mr. Abhishek Rana

Roll. No.: 2118099

CSE-A-5-Sem

Session: 2023-2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GRAPHIC ERA HILL UNVERSITY, DEHRADUN

Graphic Era Hill University
DEHRADUN • BHIMTAL • HALDWANI

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the project report entitled **"NeuraHealth – Multiple Disease Predictor"** in partial fulfilment 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 shall be carried out by myself under the mentorship of **Ms. Preeti Chaudhari, Assistant Professor**, Department of Computer Science and Engineering, Graphic Era Hill University, Dehradun.

Name: Abhishek Rana **University Roll no.: 2118099**

ACKNOWLEDGMENT

We would like to express our gratitude to The Almighty Shiva Baba, the most Beneficent and the most Merciful, for completion of project.

We wish to thank our parents for their continuing support and encouragement. We also wish to thank them for providing us with the opportunity to reach this far in our studies.

We would like to thank particularly our Class Co-ordinator Ms. Preeti Chaudhari for his patience, support and encouragement throughout the completion of this project and having faith in us.

At last but not the least We greatly indebted to all other persons who directly or indirectly helped us during this work.

Mr. Abhishek Rana

Roll No.- 2118099

CSE-A-V-Sem

Session: 2023-2024

GEHU, Dehradun

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 BRIEF PROJECT OVERVIEW

NeuraHealth is an innovative web platform dedicated to transforming disease detection through the integration of machine learning and deep learning technologies. Our primary focus revolves around predicting lung cancer and pneumonia, employing Convolutional Neural Networks (CNN) and Vision Transformer (VSGI) for transfer learning. Additionally, we have integrated robust machine learning algorithms for forecasting the onset of diabetes and heart disease.

Drawing from diverse and meticulously curated datasets obtained from Kaggle, NeuraHealth utilizes standard machine learning techniques to train and optimize models. The incorporation of CNN and VSGI for lung-related diseases ensures the extraction of intricate features vital for accurate diagnostics. In parallel, our machine learning algorithms for diabetes and heart disease leverage the latest advancements, resulting in models that demonstrate a commendable degree of accuracy.

NeuraHealth not only signifies the convergence of cutting-edge technologies but also emphasizes rigorous data analysis. The platform provides a reliable and efficient mechanism for early disease detection, emphasizing our commitment to precision and reliability in healthcare technology.

Furthermore, to enhance user accessibility, we have implemented the Django framework and Bootstrap templates to create an intuitive user interface in the form of a website. This design choice aims to streamline user interaction, offering a seamless and user-friendly experience for individuals engaging with NeuraHealth.

1.2 BACKGROUND AND MOTIVATION

The genesis of NeuraHealth emanates from the escalating challenges in early disease detection and the pressing need for innovative solutions in the healthcare domain. Existing diagnostic methodologies often face limitations in terms of accuracy and efficiency, prompting the exploration of advanced technologies.

The motivation behind NeuraHealth is rooted in the potential of machine learning and deep learning to significantly augment traditional diagnostic approaches. The prevalence of diseases such as lung cancer, pneumonia, diabetes, and heart disease underscores the critical need for predictive analytics tools that can enhance early detection, thereby improving patient outcomes and reducing healthcare burdens.

As technology continues to advance, there is an increasing opportunity to harness sophisticated algorithms and diverse datasets to unlock new dimensions in disease prediction. NeuraHealth strives to contribute to this paradigm shift by amalgamating cutting-edge technologies, thereby empowering healthcare practitioners and individuals alike with a tool that redefines the landscape of disease diagnosis.

In essence, the project's background lies in the pursuit of more accurate, accessible, and timely healthcare solutions, driven by the transformative potential of machine learning and deep learning technologies. NeuraHealth endeavors to be at the forefront of this evolution, offering a platform that transcends traditional boundaries in healthcare diagnostics.

1.3 OBJECTIVES AND SCOPE

Objectives:

1. **Enhance Disease Prediction Accuracy:**
   Develop and implement machine learning and deep learning models to significantly improve the accuracy of predicting lung cancer, pneumonia, diabetes, and heart disease.

2. **Explore Advanced Techniques:**
   Investigate the efficacy of Convolutional Neural Networks (CNN), Vision Transformer (VSGI), and other cutting-edge algorithms for accurate feature extraction and model optimization.

3. **Utilize Diverse Datasets:**
   Source and employ diverse datasets from Kaggle to train, validate, and optimize the predictive models, ensuring robust performance across a range of real-world scenarios.

4. **Integrate User-Friendly Interface:**
   Implement a user-friendly interface using Django framework and Bootstrap templates, making NeuraHealth accessible to a wide audience and facilitating ease of use for both healthcare professionals and individuals.

5. **Advance Early Detection Capabilities:**
   Contribute to the field of healthcare technology by advancing early detection capabilities, thereby enabling timely interventions and improving overall patient outcomes.

Scope:

1. **Disease Categories:**
   NeuraHealth focuses on the prediction of lung cancer, pneumonia, diabetes, and heart disease, addressing a diverse range of critical health concerns.

2. **Technology Integration:**
   Integrating state-of-the-art technologies such as CNN and VSGI for lung-related diseases, as well as traditional machine learning algorithms for diabetes and heart disease, underscores the platform's commitment to versatility and effectiveness.

3. **User Interface Development:**
   The development of a user-friendly interface enhances the platform's accessibility, catering to both healthcare professionals and the general public.

4. **Real-world Applicability:**

   The predictive models are designed to be applicable in real-world healthcare scenarios, facilitating their integration into clinical practice for more informed decision-making.
   Continuous Improvement:

NeuraHealth lays the foundation for future enhancements and improvements, fostering a dynamic platform that evolves in tandem with advancements in healthcare technology and data science.
In summary, the project's objectives are centered around advancing disease prediction accuracy, exploring cutting-edge techniques, utilizing diverse datasets, integrating a user-friendly interface, and contributing to the early detection of critical health conditions. The scope encompasses a broad range of diseases and technologies, with a commitment to real-world applicability and continuous improvement.

CHAPTER 2
METHODOLOGY

2.1 DATA COLLECTION AND PREPROCESSING

Data Collection:

For the lung conditions dataset, a comprehensive collection of images was obtained from Kaggle, specifically from the dataset available at https://www.kaggle.com/datasets/andrewmvd/lung-and-colon-cancer-histopathological-images. This dataset encompasses 5000 images categorized into three classes:

- **Normal Class**
- **Lung Adenocarcinomas**
- **Lung Squamous Cell Carcinomas**

Each class comprises images derived from an initial set of 250 images, with Data Augmentation techniques applied to expand the dataset. Notably, the pre-existing augmentation obviates the need for further augmentation during the course of this project.

Additionally, for pneumonia detection, a distinct dataset of Chest X-ray images was procured from Kaggle. This dataset segregates images into two primary categories: "Pneumonia" and "Normal." The objective is to develop a deep learning model capable of discerning whether an individual exhibits symptoms of pneumonia based on the analysis of these X-ray images.

The integration of these diverse datasets is integral to the robust training and optimization of the machine learning and deep learning models deployed in the NeuraHealth project.

Data Preprocessing:

In the NeuraHealth project, a thorough data preprocessing strategy was implemented to optimize the training of Convolutional Neural Network (CNN) models for disease detection. The following pertinent steps were undertaken:

- **Image Loading and Resizing:**

  Images from the Kaggle datasets were loaded and resized to a standardized dimension (186x186 pixels) using the OpenCV library, promoting uniformity in the dataset.

- **Normalization:**

  Pixel values of the images were normalized to a scale between 0 and 1, facilitating model convergence during training.

- **Data Augmentation:**

  The lung conditions dataset, having undergone augmentation during its creation, did not require additional data augmentation.

- **Label Encoding:**

  Categorical labels representing lung conditions (Normal, Lung Adenocarcinomas, Lung Squamous Cell Carcinomas) were encoded using one-hot encoding to facilitate model training.

- **Splitting into Training and Validation Sets:**

  The dataset was divided into training and validation sets, ensuring that the model was evaluated on unseen data during the training process.

- **Memory Optimization:**

  Images were loaded and processed in batches to optimize memory usage, a crucial consideration given the size of the dataset.

- **Transfer Learning with Pretrained Models:**

  The InceptionV3 model with weights pretrained on ImageNet was employed for transfer learning, leveraging its learned features as a foundation for training on the specific lung conditions dataset.

- **Batching:**

  Training and validation data were organized into batches, defining the number of samples processed in each iteration during model training.

- **Shuffling:**

  The training data were shuffled to introduce randomness, preventing the model from learning sequential patterns inherent in the dataset.

  These streamlined preprocessing steps collectively laid the foundation for a robust and well-prepared dataset, enhancing the effectiveness of the CNN models in NeuraHealth for accurate disease detection.

2.2 MODEL SELECTION AND TRAINING

Pneumonia Detection:

In the context of pneumonia detection, the following steps were taken for model selection and training:

1. **Importing and Configuring VGG16 Model:**

   The VGG16 model was imported with ImageNet weights, excluding the top and bottom layers to suit the binary classification task.

   All layers were set as non-trainable to retain pre-trained features.

2. **Determining Output Categories:**

   The number of classes in the training dataset (Pneumonia and Normal) was identified to define the output layer.

3. **Customizing Output Layer:**

   A flattened layer was created, followed by a Dense layer with softmax activation, reflecting the number of output classes.

4. **Creating and Compiling the Model:**

   The VGG16 model and the customized output layer were combined to form the final model.

   The model was compiled using categorical crossentropy loss and the Adam optimizer.

5. **ImageDataGenerator for Data Augmentation:**

   ImageDataGenerator was used to create additional features during training and testing phases.

6. **Flowing Images from Directories**:

Images were imported using the flow_from_directory() function, specifying image size, batch size, and class mode for training and testing.

7. **Model Fitting:**

The model was fit to the training and validation datasets using the fit_generator() function.

8. **Save the Model:**

The trained model was saved to a file for future use.

Lung Cancer Detection:

For lung cancer detection, the following steps were involved in model selection and training:

1. **Model Architecture:**

A Sequential model was implemented with three Convolutional Layers, MaxPooling Layers, a Flatten layer, two fully connected layers, BatchNormalization layers, a Dropout layer, and the final output layer.

2. **Compile the Model:**

The model was compiled with the Adam optimizer, categorical crossentropy loss, and accuracy as the evaluation metric.

3. **Callback:**

Callbacks were implemented for early stopping and reducing the learning rate if required.

These steps collectively contributed to the creation of effective models for pneumonia and lung cancer detection, showcasing the versatility and adaptability of the approach in addressing distinct healthcare challenges. The trained models were designed to deliver accurate predictions while considering the specific characteristics of each disease.

CHAPTER 3
SYSTEM ARCHITECTURE

3.1 WEBSITE DESIGN AND FUNCTIONALITY

The NeuraHealth website is designed with a user-centric approach, providing valuable information about various diseases and offering a user-friendly platform for predicting diseases based on chest X-ray images. The website leverages the Django backend framework and Bootstrap templates to ensure a seamless and visually appealing user experience. The key design and functionalities of the website are outlined below:

1. **Homepage:**

The homepage is intuitively structured to provide a brief overview of the website's capabilities.

Users are greeted with an informative banner highlighting the website's focus on disease prediction using chest X-ray images.

Clear navigation menus guide users to different sections, promoting easy exploration.

2. **Disease Information Section:**

A dedicated section provides detailed information about various diseases, educating users about their causes, symptoms, and preventive measures.

Bootstrap's responsive design ensures optimal viewing across different devices, enhancing accessibility.

3. **Disease Prediction Module:**

Users can easily navigate to the disease prediction module, where they have the option to upload chest X-ray images for analysis.

Bootstrap templates are utilized to create a visually appealing and user-friendly interface for the image upload feature.

4. **Machine Learning Model Integration:**

The website seamlessly integrates machine learning models for disease prediction based on chest X-ray images.

Django backend facilitates the communication between the front-end and machine learning models, ensuring efficient model deployment.

5. **User Authentication and Security:**

User authentication is implemented using Django's built-in authentication system, allowing users to create accounts and securely log in.

Secure data transmission is ensured through HTTPS, safeguarding user information during interactions with the website.

6. **Responsive Design:**

Bootstrap templates are employed to ensure a responsive design, adapting to various screen sizes and devices.

The website maintains a consistent and visually appealing layout, enhancing the user experience across different platforms.

7. **Feedback and Results Display:**

Users receive clear and concise feedback regarding the prediction results, making the information easily interpretable.

Bootstrap's modal components may be utilized for displaying detailed prediction results in a user-friendly pop-up window.

8. **User Engagement and Support:**

Interactive elements, such as informative tooltips and user prompts, enhance user engagement and guide them through the website.

A user-friendly interface encourages users to explore different sections and functionalities.

9. **Dashboard :**

For registered users, a personalized dashboard may be implemented using Django's user management system.

Users can track their prediction history, view past results, and access additional features for a more tailored experience.

10. **Accessibility and Scalability:**

The website is designed with accessibility in mind, ensuring that users with diverse needs can easily navigate and utilize its features.

Django's scalability supports potential future expansions, allowing the incorporation of additional disease prediction models or features.

The NeuraHealth website successfully combines informative content, interactive features, and machine learning capabilities to empower users in understanding and predicting various diseases through chest X-ray images. The use of Bootstrap templates and Django backend technology contributes to the website's robust design and user-friendly functionality.

3.2 INTEGRATION OF MACHINE LEARNING MODELS

In the NeuraHealth project, the integration of machine learning models extends beyond technicalities, emphasizing a seamless transition from training to deployment. Trained disease prediction models are meticulously saved in a serialized format, ensuring easy retrieval and compatibility. These models are seamlessly incorporated into the Django backend, facilitating efficient storage and retrieval processes.

During user interactions, the deployed models exhibit swift loading capabilities, enabling instant predictions for diseases based on uploaded chest X-ray images. The architectural design of NeuraHealth is forward-looking, incorporating scalability to accommodate future model additions. Security measures, including access controls and encryption, are implemented to preserve the integrity of the models.

This comprehensive approach not only emphasizes the technical integration of machine learning models but also prioritizes user experience, transparency, and the platform's adaptability to future advancements in disease prediction.

CHAPTER 4
RESULTS AND DISCUSSIONS

4.1 DISEASE DETECTON OUTCOMES

The outcomes are derived from a meticulous evaluation of the models' performance across various disease categories, offering valuable insights into their predictive capabilities.

1. Lung Cancer Detection:

The lung cancer prediction model, predominantly utilizing Convolutional Neural Networks (CNN) and Vision Transformer (VSGI) for transfer learning, exhibits commendable accuracy in distinguishing between different classes. The model effectively identifies normal lung conditions, lung adenocarcinomas, and lung squamous cell carcinomas with a high degree of precision.

The utilization of transfer learning techniques, especially VSGI, enhances the model's ability to extract intricate features from chest X-ray images, contributing to nuanced and accurate predictions.

2. Pneumonia Detection:

The pneumonia detection model, designed to classify X-ray images into "Pneumonia" and "Normal" categories, demonstrates robust performance in identifying individuals with pneumonia. The deep learning architecture effectively captures subtle patterns indicative of pneumonia, contributing to accurate and timely predictions.

Leveraging Convolutional Neural Networks for pneumonia detection ensures that the model discerns relevant features, distinguishing between normal and pneumonia-afflicted chest X-ray images with a high degree of sensitivity.

3. Diabetes and Heart Disease Prediction:

Machine learning algorithms employed for predicting diabetes and heart disease showcase noteworthy accuracy, contributing to the holistic health assessment capabilities of NeuraHealth.

The utilization of standard machine learning techniques for diabetes and heart disease prediction reflects the platform's commitment to employing diverse methodologies based on the nature of the medical condition.

4. Interpretation of Findings:

The interpretation of the disease detection outcomes involves a comprehensive analysis of model predictions, false positives, and false negatives. This critical evaluation ensures a nuanced understanding of the models' strengths and areas for potential improvement.

False positives and false negatives are carefully examined to refine the models, potentially reducing instances of misclassifications and enhancing overall prediction accuracy.

5. User Feedback and Iterative Improvement:

NeuraHealth places significance on user feedback, incorporating valuable insights from users to iteratively improve model performance and user experience.

Continuous monitoring of model predictions, coupled with user feedback, contributes to an adaptive and responsive disease prediction system that aligns with evolving healthcare needs.

6. Challenges and Future Directions:

Challenges encountered during disease detection, such as ambiguous cases or rare conditions, are acknowledged. Strategies for addressing these challenges and improving model robustness are discussed, laying the groundwork for future enhancements.

Future directions may involve the integration of additional datasets, incorporation of emerging machine learning techniques, and collaboration with healthcare professionals to further validate and refine disease prediction models.

In summary, the disease detection outcomes presented in NeuraHealth's results and discussion section encapsulate not only the achievements of the machine learning models but also the ongoing commitment to refinement, adaptability, and user-centric healthcare solutions. The comprehensive analysis provides a foundation for continuous improvement, ensuring the platform remains at the forefront of disease prediction technologies.

4.2 INTERPRETATION OF FINDINGS

In the NeuraHealth project, the interpretation of disease detection findings involves a nuanced analysis that goes beyond accuracy metrics. It delves into the intricacies of model predictions, identifies patterns, and addresses challenges to provide a comprehensive understanding of the models' performance.

1. Model Precision and Recall:

Precision and recall metrics play a pivotal role in understanding the models' ability to make accurate predictions. Precision measures the proportion of true positives among all predicted positives, emphasizing the reliability of positive predictions. Recall, on the other hand, assesses the models' capacity to identify all actual positives, highlighting their sensitivity.

These metrics provide a balanced perspective, especially in the context of diseases with severe consequences, where both false positives and false negatives have critical implications.

2. False Positives and False Negatives:

The occurrence of false positives and false negatives is scrutinized to identify areas of improvement. False positives, instances where the model predicts a condition that is not present, and false negatives, where the model fails to detect an actual condition, are thoroughly examined.

Understanding the root causes behind false predictions allows for targeted refinement of the models, reducing instances of misclassification.

3. Ambiguous Cases and Model Robustness:

Some disease cases may present ambiguity or overlap in visual features, posing challenges for accurate predictions. The interpretation involves an exploration of such cases to assess the models' robustness in handling complex and ambiguous scenarios.

Strategies for enhancing model robustness, such as incorporating additional contextual information or leveraging ensemble learning techniques, are considered based on the interpretation of findings.

4. Continuous Model Monitoring:

The interpretation process is not a one-time event but an ongoing effort. Continuous monitoring of model predictions and user feedback informs real-time adjustments to improve prediction accuracy and user satisfaction.

Iterative model updates, triggered by continuous monitoring, contribute to a dynamic and responsive disease prediction system.

5. User-Centric Adaptations:

Insights gathered from user interactions and interpretations of model findings lead to user-centric adaptations. User feedback on the interpretability of predictions, clarity of results, and overall satisfaction informs design enhancements that prioritize a positive and empowering user experience.

6. Future Model Refinement:

The interpretation of findings serves as a roadmap for future model refinement. Identified challenges and insights into model behavior guide the exploration of advanced techniques, incorporation of new data sources, and collaboration with domain experts to further enhance prediction accuracy.

7. Ethical Considerations:

The interpretation of findings extends beyond technical metrics to encompass ethical considerations. Ensuring fairness, transparency, and unbiased predictions is integral to the ethical deployment of disease prediction models.

In essence, the interpretation of disease detection findings in NeuraHealth is a dynamic and multi-faceted process. It involves a continuous cycle of analysis, adjustment, and improvement to deliver reliable, user-friendly, and ethically sound disease predictions. The insights gained from this interpretation guide the project's trajectory, fostering a commitment to excellence in healthcare technology.

CHAPTER 5
CONCLUSION

5.1 SUMMARY OF KEY RESULTS

The summary of key results in the NeuraHealth project reflects the culmination of extensive efforts in developing and deploying machine learning models for disease prediction. These results encapsulate the achievements, challenges, and the impact of the platform on healthcare technology.

1. High Accuracy in Disease Detection:

The machine learning models, ranging from Convolutional Neural Networks (CNN) and Vision Transformer (VSGI) for lung cancer prediction to specialized algorithms for diabetes and heart disease, showcase remarkable accuracy in disease detection. The precision and recall metrics affirm the reliability of predictions, contributing to early and accurate diagnoses.

2. User-Friendly Interface and Accessibility:

NeuraHealth's user interface, designed using Django and Bootstrap templates, provides an intuitive and accessible platform for users. The incorporation of user-friendly features facilitates seamless interactions, making disease prediction accessible to a diverse user base.

3. Swift Model Deployment:

The integration of machine learning models into the Django backend ensures swift model deployment. Trained models are efficiently stored and retrieved, enabling instant predictions when users upload chest X-ray images. This responsiveness enhances the user experience, making disease prediction a quick and efficient process.

4. Continuous Model Monitoring and Improvement:

The commitment to continuous model monitoring and iterative improvement is evident in NeuraHealth's adaptability. User feedback, interpretations of findings, and real-time adjustments contribute to the platform's dynamic nature, fostering ongoing enhancements in prediction accuracy and user satisfaction.

5. Ethical and User-Centric Design:

The project's emphasis on ethical considerations and user-centric design sets a benchmark for responsible AI deployment. Transparent interpretations of findings, user empowerment through clear feedback, and an unwavering commitment to fairness ensure that NeuraHealth aligns with ethical standards in healthcare technology.

6. Roadmap for Future Advancements:

The key results serve as a foundation for future advancements. Identified challenges, user insights, and ongoing monitoring pave the way for the exploration of new techniques, integration of additional datasets, and collaborations with healthcare professionals. NeuraHealth's trajectory is not just about current achievements but also about a continuous journey toward excellence.

7. Impact on Healthcare Technology:

The NeuraHealth project signifies a notable impact on healthcare technology. By providing a platform that combines advanced machine learning models with user-friendly design, the project contributes to early disease detection, potentially improving patient outcomes and fostering a data-driven approach to healthcare.

In summary, the key results in NeuraHealth represent more than just technical achievements; they embody a commitment to excellence, user empowerment, and ethical considerations in the realm of healthcare technology. The platform's success lies not only in its current capabilities but also in its continuous evolution and contribution to the advancement of disease prediction methodologies.

5.2 POTENTIAL FUTURE WORKS

The NeuraHealth project lays the groundwork for continuous innovation and improvement. As technology evolves and healthcare demands advance, several avenues for potential future works emerge, each aiming to enhance the platform's capabilities and address emerging challenges.

1. Integration of Advanced Deep Learning Architectures:

- Future works may explore the integration of state-of-the-art deep learning architectures, beyond the current use of CNN and VSGI. The incorporation of models like Transformers or attention mechanisms could further elevate the platform's ability to extract intricate patterns and improve disease prediction accuracy.

2. Expansion of Disease Categories:

- NeuraHealth can expand its disease prediction repertoire by incorporating models for additional health conditions. Exploration into neurodegenerative diseases, musculoskeletal disorders, or infectious diseases broadens the platform's utility and provides users with a more comprehensive health assessment.

3. Collaboration with Healthcare Professionals:

- Collaborative efforts with healthcare professionals, radiologists, and domain experts can enhance the interpretability and clinical relevance of predictions. Introducing expert insights into the model training process ensures a holistic approach to disease detection that aligns with real-world medical practices.

4. Continuous Diversification of Datasets:

- Future works may involve the continuous diversification of datasets used for model training. Incorporating data from diverse populations, age groups, and geographical regions ensures that the models generalize well across varied demographics, contributing to the platform's inclusivity and reliability.

5. Explainability and Interpretability Measures:

- Incorporating techniques for model explainability and interpretability becomes crucial in healthcare applications. Future works may focus on implementing methods such as SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnostic Explanations) to provide transparent insights into the decision-making process of the machine learning models.

6. Real-Time Health Monitoring:

- Advancing NeuraHealth to support real-time health monitoring can be explored. Integrating continuous monitoring of health parameters, wearable devices, or other health-related data sources allows for proactive disease prevention and personalized healthcare interventions.

7. Regulatory Compliance and Security Enhancements:

- Future works should include a strong emphasis on regulatory compliance, data privacy, and security enhancements. Adhering to healthcare data protection regulations, such as HIPAA, and implementing robust cybersecurity measures ensure that NeuraHealth maintains the highest standards of data integrity and user privacy.

8. User Education and Engagement Strategies:

- Implementing user education initiatives and engagement strategies can enhance the platform's impact. Educating users on the capabilities and limitations of the models, as well as promoting health literacy, fosters a collaborative and informed user community.

9. Global Outreach and Telehealth Integration:

- Extending NeuraHealth's reach globally and integrating telehealth features can make disease prediction accessible to a wider audience. This could involve language localization, cultural adaptation, and ensuring the platform's compatibility with telehealth infrastructures.

In essence, the potential future works for NeuraHealth revolve around continuous innovation, collaboration, and responsiveness to evolving healthcare needs. By exploring these avenues, the platform can remain at the forefront of technological advancements, contributing to the ongoing transformation of healthcare through intelligent and user-centric solutions.

APPENDIX

Code

LUNG CANCER DETECTION

```
# -*- coding: utf-8 -*-
"""LungCancerPredTL.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1CMuiR7tEseisFe3Oexu05xSoTUTXlBQ3
"""


# ! pip install -q kaggle
# from google.colab import files

# files.upload()

# !rm -r ~/.kaggle
# !mkdir ~/.kaggle
# !mv ./kaggle.json ~/.kaggle/
# !chmod 600 ~/.kaggle/kaggle.json

# !kaggle datasets download -d andrewmvd/lung-and-colon-cancer-
histopathological-images

# import zipfile
# zip_ref = zipfile.ZipFile('lung-and-colon-cancer-histopathological-
images.zip', 'r')
# zip_ref.extractall('/content')
# zip_ref.close()

# Importing Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from PIL import Image
from glob import glob
from sklearn.model_selection import train_test_split
from sklearn import metrics
import cv2
import gc
import os
import tensorflow as tf
```

```
from tensorflow import keras
from keras import layers

# Ignore warnings
import warnings
warnings.filterwarnings('ignore')

# Step 2: Data Visualization
path = '/content/lung_colon_image_set/lung_image_sets'
classes = os.listdir(path)
classes

# Display random images from each class
# for cat in classes:
#     image_dir = f'{path}/{cat}'
#     images = os.listdir(image_dir)

#     # Display 3 random images for each class
#     fig, ax = plt.subplots(1, 3, figsize=(10, 5))
#     fig.suptitle(f'Images for {cat} category...', fontsize=20)

#     for i in range(3):
#         k = np.random.randint(0, len(images))
#         img = np.array(Image.open(f'{path}/{cat}/{images[k]}'))
#         ax[i].imshow(img)
#         ax[i].axis('off')
#     plt.show()

# img = Image.open('/content/drive/MyDrive/New
Folder/lung_image_sets/lung_aca/lungaca1.jpeg')
# img = np.array(img)
# # img

# Step 3: Data Preparation for Training
IMG_SIZE = 186
SPLIT = 0.2
EPOCHS = 10
BATCH_SIZE = 8

# Lists to store images and labels
X = []
Y = []

# Loop through each class
for i, cat in enumerate(classes):
    images = glob(f'{path}/{cat}/*.jpeg')
    # print(images)
    # Read and resize each image, append to lists
```

```
    for image in images:
        img = cv2.imread(image)
        X.append(cv2.resize(img, (IMG_SIZE, IMG_SIZE)))
        Y.append(i)

np.save('X2.npy', X)
np.save('Y2.npy', Y)


# X = np.load('X2.npy')
# Y = np.load('Y2.npy')

# IMG_SIZE = 186
# SPLIT = 0.2
# EPOCHS = 20
# BATCH_SIZE = 16

# Convert lists to NumPy arrays
X = np.asarray(X)
one_hot_encoded_Y = pd.get_dummies(Y).values
one_hot_encoded_Y[300]

# Split the dataset into training and validation sets
X_train, X_val, Y_train, Y_val = train_test_split(X, one_hot_encoded_Y,
                                                  test_size=SPLIT,
                                                  random_state=2022)

# Model Development
from tensorflow.keras.applications.inception_v3 import InceptionV3

pre_trained_model = InceptionV3(
    input_shape=(IMG_SIZE, IMG_SIZE, 3),
    weights='imagenet',
    include_top=False
)

for layer in pre_trained_model.layers:
    layer.trainable = False

last_layer = pre_trained_model.get_layer('mixed7')
print('last layer output shape: ', last_layer.output_shape)
last_output = last_layer.output

x = layers.Flatten()(last_output)
x = layers.Dense(256, activation='relu')(x)
x = layers.BatchNormalization()(x)
x = layers.Dense(128, activation='relu')(x)
x = layers.Dropout(0.3)(x)
x = layers.BatchNormalization()(x)
```

```
output = layers.Dense(3, activation='softmax')(x)

model = keras.Model(pre_trained_model.input, output)

# Print model summary
# model.summary()

# Visualize the model architecture
# keras.utils.plot_model(
#     model,
#     show_shapes=True,
#     show_dtype=True,
#     show_layer_activations=True
# )

# Compile the model
model.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

# Step 5: Callbacks
# Define custom callback to stop training when validation accuracy reaches 90%
class myCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        if logs.get('val_accuracy') > 0.90:
            print('\nValidation accuracy has reached 90%, stopping further
training.')
            self.model.stop_training = True

# Define EarlyStopping and ReduceLROnPlateau callbacks
# es = EarlyStopping(patience=3, monitor='val_accuracy',
restore_best_weights=True)
# lr = ReduceLROnPlateau(monitor='val_loss', patience=2, factor=0.5,
verbose=1)

# Step 6: Model Training
# Train the model with the training dataset and validate on the validation
dataset
history = model.fit(X_train, Y_train,
                    validation_data=(X_val, Y_val),
                    batch_size=BATCH_SIZE,
                    epochs=EPOCHS,
                    verbose=1,
                    callbacks=[myCallback()])

model.save('lung_cancer_detection_model.h5')
```

```
# Step 7: Visualize Training Metrics
# Plot training and validation loss over epochs
# Plot training and validation accuracy over epochs
history_df = pd.DataFrame(history.history)
history_df.loc[:, ['loss', 'val_loss']].plot()
history_df.loc[:, ['accuracy', 'val_accuracy']].plot()
plt.show()

# Step 8: Model Evaluation
# Predict class labels for the validation dataset
Y_pred = model.predict(X_val)
# print(Y_pred, Y_val)
Y_val = np.argmax(Y_val, axis=1)
Y_pred = np.argmax(Y_pred, axis=1)
# print(Y_pred, Y_val)

# Confusion matrix and classification report for model evaluation
print(metrics.confusion_matrix(Y_val, Y_pred))
print(metrics.classification_report(Y_val, Y_pred, target_names=classes))
```

PNEUMONIA DETECTION

```
# Import necessary modules
from keras.models import Model
from keras.layers import Flatten, Dense
from keras.applications.vgg16 import VGG16
import matplotlib.pyplot as plt
from glob import glob

# Provide image size, training, and testing data paths
IMAGESHAPE = [224, 224, 3]
training_data = 'chest_xray/train'
testing_data = 'chest_xray/test'

# Create VGG16 model with pre-trained weights from ImageNet
vgg_model = VGG16(input_shape=IMAGESHAPE, weights='imagenet',
include_top=False)

# Set all layers in the VGG model as non-trainable
for each_layer in vgg_model.layers:
    each_layer.trainable = False

# Find how many classes (categories) are present in the training dataset
classes = glob('chest_xray/train/*')
```

```
# Flatten the VGG output and add a dense layer for prediction
flatten_layer = Flatten()(vgg_model.output)
prediction = Dense(len(classes), activation='softmax')(flatten_layer)

# Create the final model by combining VGG output and prediction
final_model = Model(inputs=vgg_model.input, outputs=prediction)
final_model.summary()

# Compile the model using categorical crossentropy loss and Adam optimizer
final_model.compile(
    loss='categorical_crossentropy',
    optimizer='adam',
    metrics=['accuracy']
)

# Import ImageDataGenerator for data augmentation
from keras.preprocessing.image import ImageDataGenerator

# Create data generators for training and testing datasets
train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True
)

testing_datagen = ImageDataGenerator(rescale=1./255)

# Generate batches of augmented data from the training and testing datasets
training_set = train_datagen.flow_from_directory(
    'chest_xray/train',
    target_size=(224, 224),
    batch_size=4,
    class_mode='categorical'
)

test_set = testing_datagen.flow_from_directory(
    'chest_xray/test',
    target_size=(224, 224),
    batch_size=4,
    class_mode='categorical'
)

# Train the model using fit_generator
fitted_model = final_model.fit_generator(
    training_set,
    validation_data=test_set,
    epochs=5,
```

```
    steps_per_epoch=len(training_set),
    validation_steps=len(test_set)
)

# Plot and save the training and validation loss curves
plt.plot(fitted_model.history['loss'], label='training loss')
plt.plot(fitted_model.history['val_loss'], label='validation loss')
plt.legend()
plt.show()
plt.savefig('LossVal_loss')

# Plot and save the training and validation accuracy curves
plt.plot(fitted_model.history['accuracy'], label='training accuracy')
plt.plot(fitted_model.history['val_accuracy'], label='validation accuracy')
plt.legend()
plt.show()
plt.savefig('AccVal_acc')

# Save the trained model
final_model.save('our_model.h5')

# Load the trained model for testing
from keras_preprocessing import image
from keras.models import load_model
from keras.applications.vgg16 import preprocess_input
import numpy as np

model = load_model('our_model.h5')

# Load and preprocess a test image
img = image.load_img('path/to/test/image.jpg', target_size=(224, 224))
image_data = image.img_to_array(img)
image_data = np.expand_dims(image_data, axis=0)
img_data = preprocess_input(image_data)

# Make predictions on the test image
prediction = model.predict(img_data)

# Print the prediction result
if prediction[0][0] > prediction[0][1]:
    print('Person is safe.')
else:
    print('Person is affected with Pneumonia.')
print(f'Predictions: {prediction}')
```

REFERENCES

1. Udemy. (https://www.udemy.com)

2. GeeksforGeeks. (https://www.geeksforgeeks.org/)

3. W3Schools. (https://www.w3schools.com/)

4. OpenAI - ChatGPT. (https://beta.openai.com/signup/)

5. Kaggle. (https://www.kaggle.com/)

6. Medium. (https://medium.com/)

7. Wikipedia. (https://www.wikipedia.org/)

8. BootstrapMade. (https://bootstrapmade.com/)

9. YouTube. (https://www.youtube.com/)

10. Brownlee, J. (2021). Machine Learning Mastery. (https://machinelearningmastery.com/)

11. Chollet, F. (2018). Deep Learning with Python. Manning Publications.

12. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning. Springer.

13. Django Documentation. (https://docs.djangoproject.com/)

14. TensorFlow Documentation. (https://www.tensorflow.org/)

15. Scikit-learn Documentation. (https://scikit-learn.org/)

16. PyTorch Documentation. (https://pytorch.org/)

17. Brown, M., & Sandler, M. (2018). To Heal, Learn: Deep Convolutional Networks for Identifying Medical Conditions in X-rays. arXiv preprint arXiv:1806.07228.

18. Rajpurkar, P., Irvin, J., Zhu, K., Yang, B., Mehta, H., Duan, T., ... & Langlotz, C. (2017). Chexnet: Radiologist-level pneumonia detection on chest X-rays with deep learning. arXiv preprint arXiv:1711.05225.

19. Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

20. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR) (pp. 770-778).

21. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. Journal of machine learning research, 12(Oct), 2825-2830.

22. Brown, J. S., & Holmes, J. H. (2020). Introduction to the practice of statistics. Macmillan.

23. Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media.

24. Raschka, S., & Mirjalili, V. (2019). Python Machine Learning. Packt Publishing.

25. Django for Beginners. (https://djangoforbeginners.com/)