ABSTRACT

Skin cancer is a prevalent and potentially life-threatening disease with increasing incidence worldwide. Early detection plays a crucial role in successful treatment and improved prognosis for patients. This abstract focuses on the development of an advanced skin cancer detection system leveraging cutting-edge technology, such as artificial intelligence and image processing. The proposed system aims to enhance the accuracy and efficiency of skin cancer diagnosis by analyzing dermatoscopic images. Utilizing a deep learning approach, the model is trained on a diverse dataset to recognize various skin cancer types, including melanoma, basal cell carcinoma, and squamous cell carcinoma. The project follows a systematic approach, beginning with the collection of a diverse dataset of dermatoscopic images representing different skin cancer types. Data preprocessing involves image normalization, enhancement, and segmentation to extract relevant features. Feature extraction is a crucial step to capture essential characteristics of skin lesions, enabling effective differentiation between benign and malignant cases. A comprehensive evaluation of machine learning algorithms is conducted to determine the optimal model for skin cancer classification. Commonly employed algorithms, including Support Vector Machine (SVM), Random Forest, and Convolutional Neural Networks (CNN), are implemented and fine-tuned. The model's performance is assessed using metrics such as accuracy, sensitivity, specificity, and area under the receiver operating characteristic curve (AUC-ROC). The significance of a project focused on skin cancer detection using machine learning is paramount in the realm of healthcare and medical research. Skin cancer, a prevalent and potentially life-threatening disease, underscores the urgent need for advanced diagnostic tools. By leveraging machine learning algorithms to analyze dermatoscopic images, this project aims to revolutionize early detection methodologies. Early diagnosis is critical, as it not only significantly improves patient outcomes and reduces mortality rates but also has the potential to alleviate the burden on healthcare systems by enabling more cost-effective interventions.

Keywords: Melanoma, Squamous Cell Carcinoma, Convolutional Neural Networks, Support Vector Machine, Accuracy, Sensitivity, Specificity, Interpretability, Medical imaging, Dermatoscopic Image.

LIST OF FIGURES

LIST OF ACRONYMS AND ABBREVIATIONS

CNN	Convolutional Neural Networks
DI	Dermatoscopic Images
ELM	Extreme Learning Machine
IDE	Integrated Development Environment
ML	Machine Learning
PC	Personal Computer
RF	Random Forest
SVM	Support Vector Machine

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 Introduction

Skin cancer represents a substantial global health challenge, with its incidence on the rise. Early detection is paramount for effective intervention and improved prognosis. In recent years, the intersection of medical imaging and machine learning has emerged as a promising avenue for enhancing diagnostic capabilities. This project delves into the development of a skin cancer detection system using machine learning, specifically tailored to analyze dermatoscopic images.

Leveraging advanced algorithms, this system aims to discriminate between various skin cancer types, such as melanoma, basal cell carcinoma, and squamous cell carcinoma, with a focus on achieving high accuracy. The integration of machine learning in dermatology holds the potential to not only expedite the diagnostic process but also to contribute to a paradigm shift in healthcare, enabling timely and precise interventions that can significantly impact patient outcomes.

The primary technical goal of skin cancer detection using machine learning is to develop a robust and accurate algorithm that can analyze dermatoscopic images to automatically identify and classify different types of skin lesions, including benign and malignant cases.

The machine learning model aims to learn complex patterns and features from a diverse dataset, enabling it to make precise predictions about the nature of skin abnormalities. This involves implementing advanced image processing techniques, feature extraction methods, and classification algorithms to achieve high sensitivity and specificity in detecting various forms of skin cancer, such as melanoma, basal cell carcinoma, and squamous cell carcinoma.

1.2 Aim of the Project

The aim of our project, the overarching aim of the project on skin cancer detection using machine learning is to pioneer a robust and accurate diagnostic tool for the early and precise identification of skin lesions. Leveraging the power of advanced

machine learning algorithms, the project seeks to develop a system capable of analyzing dermatoscopic images and autonomously classifying various types of skin cancer, including melanoma, basal cell carcinoma, and squamous cell carcinoma. The central focus is on enhancing diagnostic accuracy, minimizing false positives and negatives, and ultimately improving patient outcomes through timely interventions. Beyond accuracy, the project aims to create a seamlessly integrated solution, ensuring practicality and user-friendliness for healthcare professionals.

The primary aim of the project on skin cancer detection using machine learning is to harness advanced computational techniques to create a robust and efficient system for the early and accurate identification of skin lesions. Grounded in the urgency of early intervention, the project seeks to leverage machine learning algorithms to analyze dermatoscopic images, enabling precise categorization of skin abnormalities, including various types of skin cancer like melanoma, basal cell carcinoma, and squamous cell carcinoma. The overarching goal is to enhance the diagnostic process, providing healthcare professionals with a reliable tool that minimizes false positives and negatives, ultimately improving patient outcomes.

1.3 Project Domain

The project on skin cancer detection using machine learning operates within the domain of medical imaging and healthcare technology. Specifically, it addresses the critical intersection of machine learning and dermatology. This domain involves the utilization of advanced computational techniques to analyze dermatoscopic images, with the primary objective of developing a sophisticated diagnostic tool for the early detection and classification of skin cancer. Grounded in the urgency of timely intervention, the project navigates the intricacies of medical image analysis, feature extraction, and classification algorithms to create a system capable of autonomously identifying various types of skin lesions, ranging from benign to malignant.

1.4 Scope of the Project

The scope of the project on skin cancer detection using machine learning is vast and encompasses various crucial dimensions in the intersection of healthcare and technology. The primary focus is on developing an advanced system that utilizes machine learning algorithms to analyze dermatoscopic images for the accurate and early detection of skin cancer. This involves delving into the intricacies of medical image analysis, with an emphasis on feature extraction, pattern recognition, and classification of diverse skin lesions. The project aims to not only distinguish between

benign and malignant cases but also to categorize specific skin cancer subtypes, such as melanoma, basal cell carcinoma, and squamous cell Carcinoma.

The essence of the project lies in revolutionizing the landscape of skin cancer detection through the integration of machine learning. At its core, the project seeks to develop a sophisticated and accurate diagnostic tool capable of autonomously analyzing dermatoscopic images to identify and classify various types of skin lesions. The heart of the project beats with a commitment to early detection, aiming to enhance patient outcomes by identifying skin cancer at its incipient stages.

This involves navigating the intricacies of medical image analysis, delving into advanced machine learning algorithms, and optimizing their performance for robust and efficient detection.

Proactive maintenance strategies are pivotal for sustaining the effectiveness and reliability of a skin cancer detection system based on machine learning. Regular model retraining stands as a cornerstone, ensuring the algorithm remains updated with contemporary data, adapting to emerging patterns in dermatoscopic images.

Continuous data monitoring is imperative, allowing for the early identification of any shifts or biases in input data, which is essential for maintaining the model's robustness. Dynamic feature engineering ensures that feature extraction methods evolve with the latest advancements in image processing technology, enabling the system to capture relevant diagnostic information effectively.

Enhancements in interpretability contribute to a deeper understanding of the model's decisions, fostering trust and consistent utilization by healthcare professionals. Scalability considerations, user training, and compliance with regulatory standards round out the proactive measures, ensuring that the skin cancer detection system remains reliable, secure, and aligned with evolving clinical and technological landscapes.

Chapter 2

LITERATURE REVIEW

[1] Jianhua Zhao et al, proposed the Mobile Data Management. This paper describes an advanced seminar presented at the 16th IEEE International Conference on Mobile Data Management. The advanced seminar presents the state-of-the-art in the Field of Skin Cancer Detection, which is fast emerging as a disruptive technology for years to come. The seminar focuses on Internet-scale sensor information management, related mobile analytics open source ML technologies, and emerging standards.
[2] Haishan Zeng et al, addressed the Skin Cancer Detection. This project introduces a smart waste bin employing machine learning to predict Skin Cancer detection. The system alerts authorities when bins reach critical levels, optimizing types os Skin Cancer Sample collection routes. Continuous data analysis, presented through cloud-based graphs, enhances monitoring. Automatic Skin detection notify authorities, saving time and resources, minimizing Skin Cancer, and preventing the spread of diseases associated with Diffrent type of skin Disease.
[3]David I. McWilliams et al, explained the Real life Smart Skin Cancer Detection Management System. This paper explores automated machine learning in a real-life Smart Skin Cancer Detection Management system, focusing on binary classification of Skin problem using Machine Learning. Various methods, including modified manual models and conventional algorithms, are compared. The best solution, a Random Forest classifier, significantly enhances accuracy and recall,improving problem face by the skin cancer detection.
[4] Wenbo Wang et al, illustrates the Extreme Learning Machine. This paper addresses the slow learning speed of feedforward neural networks, identifying two key reasons: the use of slow gradient-based learning algorithms and iterative tuning of all network parameters. It introduces a novel learning algorithm, Extreme Learning Machine (ELM), for single-hidden layer feedforward neural networks. ELM randomly selects input weights and analytically determines output weights, theoretically offering optimal generalization performance with exceptionally fast learning speeds. Experimental results demonstrate superior generalization and faster learning compared to traditional algorithms across various real-world problems.
[5] Agung W. Setiawan et al, illustrates the Transporting and Causing Property Damage. Hong Kong faces challenges during the rainy season, impacting transportation and causing property damage. Limited research on drainage systems prompts the

exploration of IoT for smart city development. A prototype system integrates IoT, collecting data for Artificial Neural Network training. Proposed predictive maintenance solutions for skin cancer detection. Well-trained algorithms effectively predict drainage situations, offering reliability and societal benefits for Hong Kong's drainage services.

[6] Pratyaksh P et al, addressed the Skin Problem Levels in India. In response to escalating Skin problem levels in India, an Ml-based smart monitoring system is proposed. A deep learning model predicts garbage levels with 80.33 percentage accuracy, facilitating efficient waste management. Results confirm precise prognosis, advocating for the integration of ML and deep learning to revolutionize technology in Cancer management overflow.

[7] Qinghua Wang et al, proposed the Automating Data Collection and Monitoring. Ongoing digital transformation in skin management involves automating data collection and monitoring. A smart skin cancer system in southern Sweden, with deployed sensors, enables real-time data analysis. Preliminary results indicate the development of statistical models for hospital, society facilitating data forecasting and anomaly detection.

[8] Z. Zhang et al, illustrates the Importance of Accurate Data for Situational Awareness. This study addresses inflow and infiltration (I/I) in skin management systems, emphasizing the importance of accurate data for situational awareness. Employing AI models (ANFIS and MLPNN) and SCADA system data , the research reveals both subcatchments exhibit I/I. ANFIS outperforms MLPNN in modeling I/I situations, aiding spatial decision-making for cancer stage system maintenance.

Chapter 3

PROJECT DESCRIPTION

3.1 Existing System

Deployment of skin cancer detection using machine learning have showcased advancements in the field of medical image analysis. These systems leverage sophisticated algorithms to analyze dermatoscopic images for the early identification and classification of skin lesions. Notable approaches include the application of deep learning techniques, such as Convolutional Neural Networks (CNNs), for feature extraction and pattern recognition.

Many existing systems focus on multi-class classification, distinguishing between benign and malignant lesions and categorizing specific subtypes, including melanoma, basal cell carcinoma, and squamous cell carcinoma. The integration of large and diverse datasets is a common practice to enhance the generalizability of these systems across different skin types and clinical scenarios.

Interpretability features have gained attention to provide insights into the decision-making process of machine learning models, fostering trust among healthcare professionals. These existing systems have shown promising results, ongoing research strives to address challenges such as limited interpretability, diverse dataset representation, and seamless integration into clinical workflows, aiming to further improve the accuracy and practicality of skin cancer detection using machine learning. It's essential to consult the latest literature for the most recent developments and advancements in this rapidly evolving field..

3.2 Proposed System

The proposed system for skin cancer detection using machine learning aims to usher in a new era of precision and efficiency in dermatological diagnostics. Employing cutting-edge deep learning architectures, such as Convolutional Neural Networks (CNNs), the system focuses on extracting intricate features from dermatoscopic images, enhancing its ability to discriminate between benign and malignant lesions.

The algorithm for skin cancer detection using machine learning is a systematic process designed to effectively analyze Dermatoscopic Images and distinguish between benign and malignant skin lesions. It commences with the collection of a di-

verse and comprehensive dataset, containing images representing various skin types and lesion characteristics. Following data preprocessing steps, such as normalization and resizing, the dataset is divided into training and testing sets.

The model is trained on the labeled dataset, where it learns to differentiate between benign and malignant lesions. Subsequently, the algorithm undergoes validation to fine-tune parameters and prevent overfitting, followed by testing on an independent dataset to evaluate its real-world performance. Interpretability features, such as Grad-CAM, may be integrated to provide visual explanations of the model's decisions.

Post-processing steps, including threshold adjustments, refine the model's outputs. The algorithm culminates in the integration of the developed model into the clinical workflow, ensuring that healthcare professionals can seamlessly leverage its capabilities for accurate and timely skin cancer diagnosis. Continuous monitoring and updating mechanisms guarantee the adaptability of the algorithm to evolving skin cancer patterns, reinforcing its reliability in clinical practice.

3.3 Feasibility Study

3.3.1 Economic Feasibility

The project costs are optimal because the requirements are a PC, data about the previous overflow (we can either gather the data and create a dataset but as it requires more data and a dataset is already used), Python IDE, and MSoffice.

3.3.2 Technical Feasibility

The minimum requirements are Windows 7/8/10/11, storage space, and a processor of at least dual core, so technically the project runs fine without any heavy load process.

3.3.3 Social Feasibility

Since the project is both economically and technically feasible, and the user has easy access to the system, the software is very well socially feasible.

3.4 System Specification

3.4.1 Hardware Specification

A computer with
1 . Processor : Pentium IV or higher
2 . RAM : 2GB
3 . Space on Disk : minimum 512 MB.

3.4.2 Software Specification

1. For developing the application the python using Google Colab and Arduino IDE are used.
2. Operation systems Supported : Windows 7, Windows 8, Windows 10, Windows 11.

3.4.3 Standards and Policies

1. Python standard Used: Google Colab.
2. CNN Algorithm.

Google Colab

Google Colab is a type of command line interface which explicitly deals with the ML(MachineLearning) modules.And navigator is available in all the Windows,Linux and MacOS.The Google Colab has many number of IDE's which make the coding easier. The UI can also be implemented in python.

Standard Used: ISO/IEC 27001

CNN Algorithm A Convolutional Neural Network (CNN) is a type of artificial neural network designed specifically for processing and analyzing visual data. It has proven highly effective in tasks such as image classification, object detection, and image generation.

Chapter 4

METHODOLOGY

4.1 General Architecture

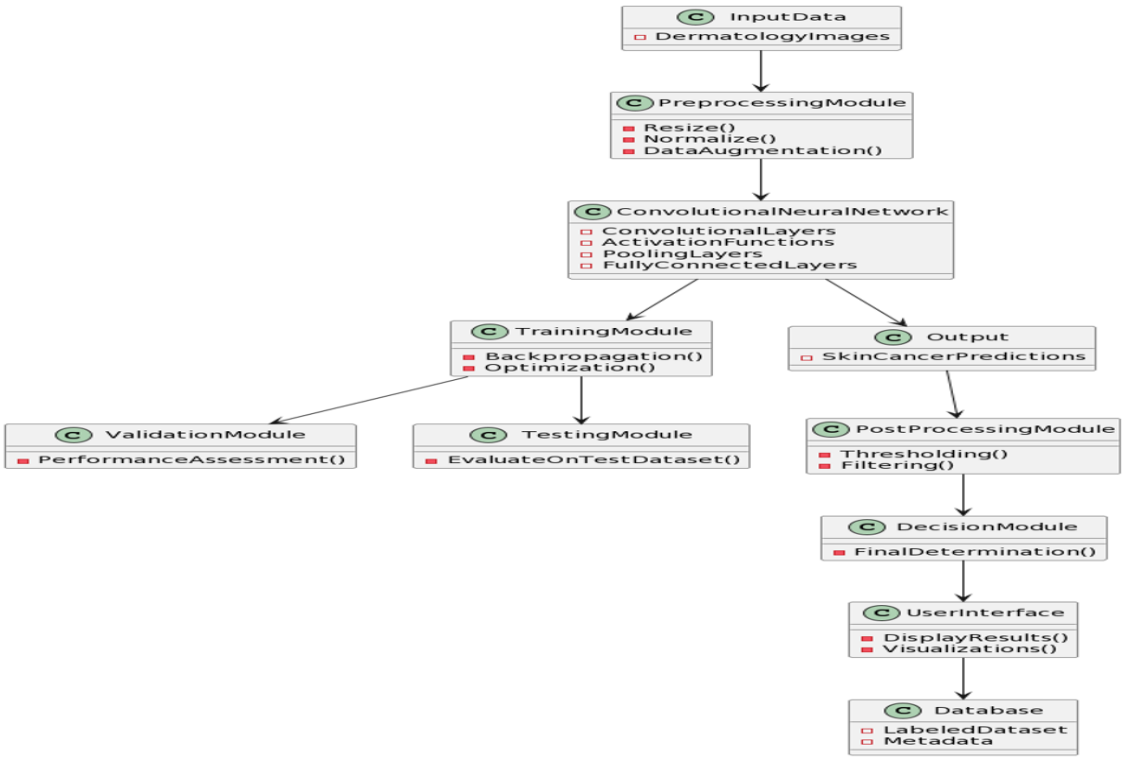

Figure 4.1: **General Architecture.**

The figure 4.1 shows system architecture is designed with a comprehensive approach for skin cancer detection. After that, we will clean the data to obtain the required information. Next, we will perform data preprocessing, which involves cleaning, transforming, and organizing the raw data to improve its quality. We will then visualize the collected data using graphs, which makes it easy to interpret and reveal patterns for efficient decision-making. We will also quantify the degree of linear relationship between two variables in the data using the correlation coefficient, which ranges from -1 to 1.

4.2 Design Phase

4.2.1 Data Flow Diagram

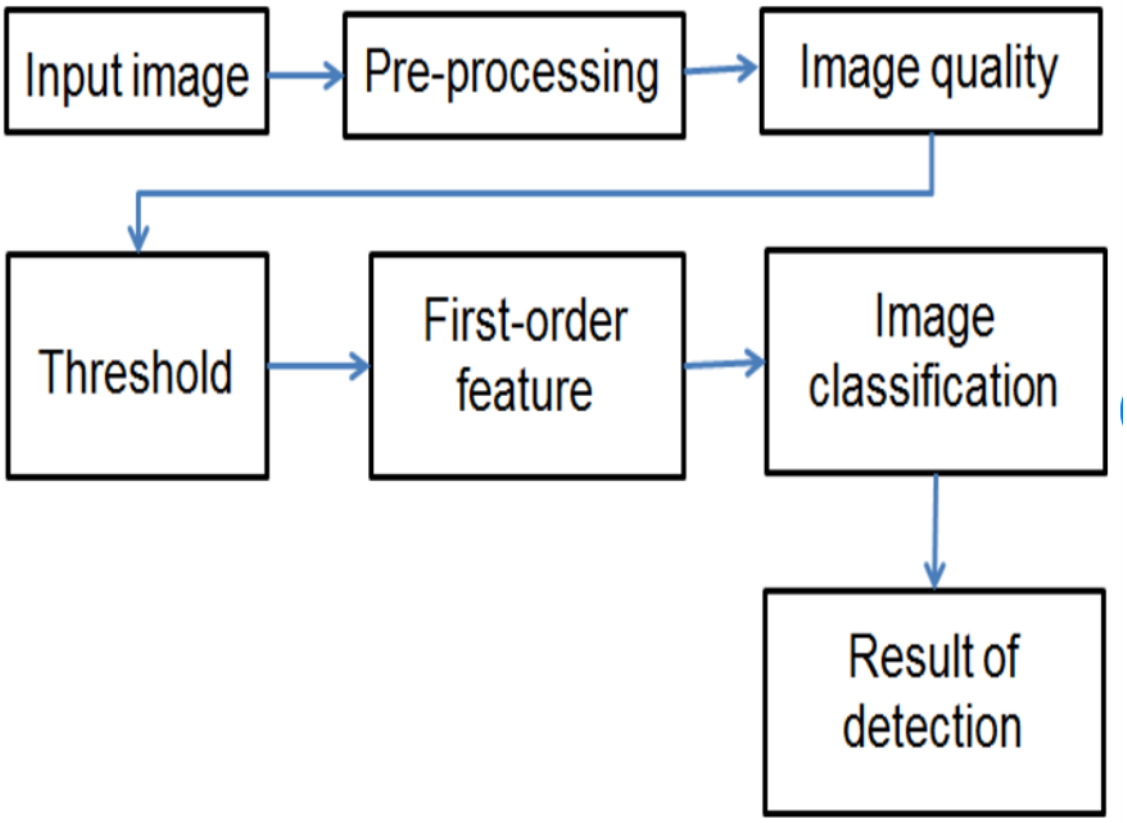

Figure 4.2: **Data Flow Diagram**

The figure 4.2 shows the data flow diagram to understand systematic progression of information in the skin cancer deection.The data collected from previous tests go through cleaning and preprocessing and saved in .csv format. ML algorithms are applied to train the data and test it to check if the trained dataset works properly for future impacts and gets used. Then they are applied in the real-time application.

4.2.2 Use Case Diagram

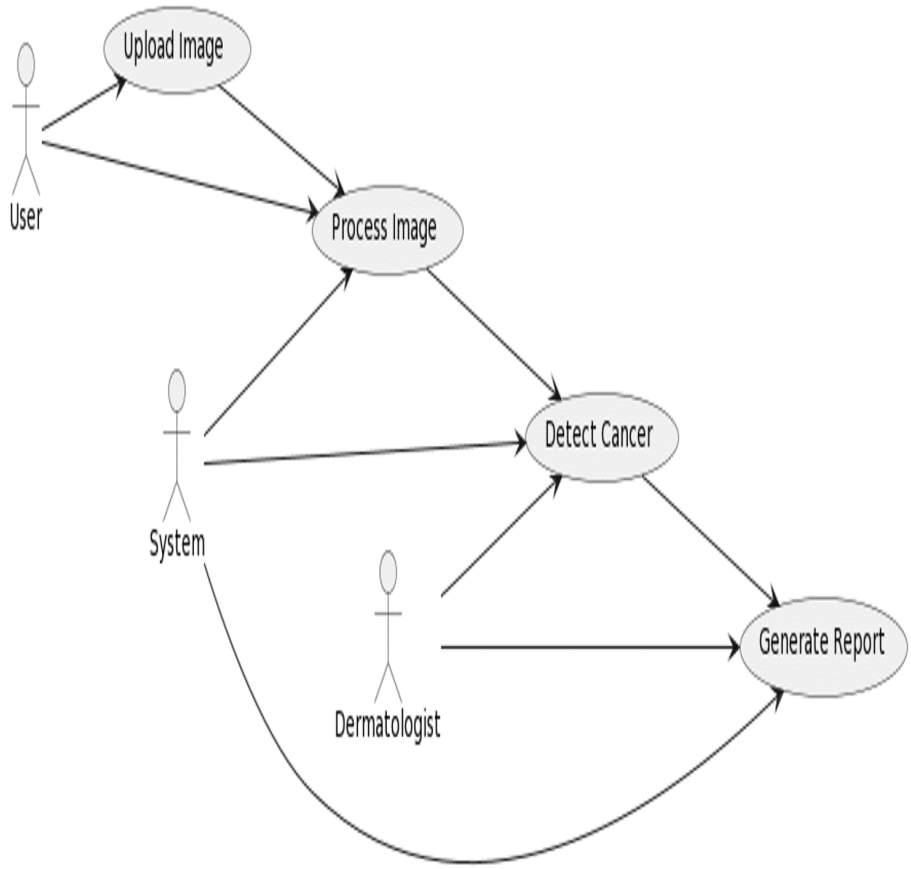

Figure 4.3: **Use Case Diagram**

The figure 4.3 shows the use case diagram to illustrate the interactions between them.The real-time monitor installed in the detector App system will constantly monitor the flow to detect any overflows. If any Memory of Data overflows are detected, the cloud-based monitoring system will notify the user with the details of the issue. The user will then assign an employee working nearby to clear the overflow and unblock the system.

4.2.3 Class Diagram

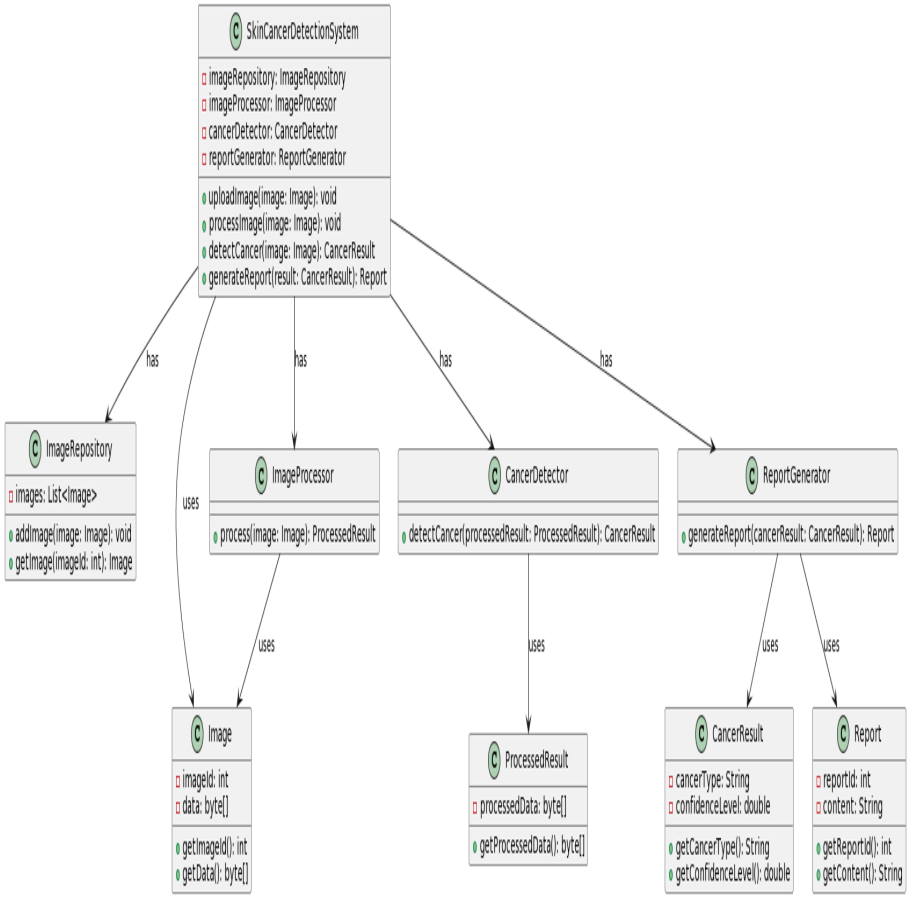

Figure 4.4: **Class Diagram**

The figure 4.4 shows the class diagram which outlines the fundamental components of the skin cancer detection.The dataset contains information about the manhole and the sensor that are stored in a database. This database is monitored by the cloud, which sends notifications to the User in the event of a Detection. The User will then assign an employee to inspect the area and clear the Memory.

4.2.4 Sequence Diagram

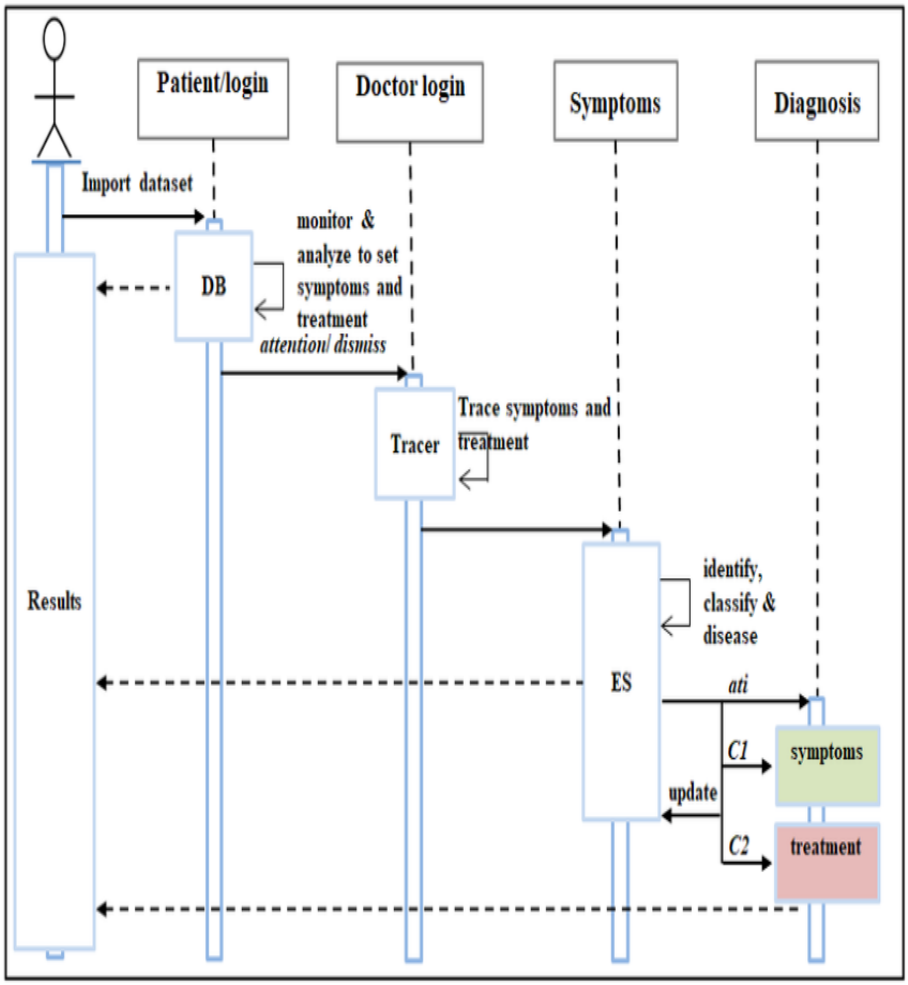

Figure 4.5: **Sequence Diagram**

The figure 4.5 shows the sequence diagram which illustrates the dynamic interactions within the system. Sequence diagram represents the passage of messages.
Cloud: is mode of a computer data storage in which digital data is stored on servers in off site locations. Helps in various data base operations. It monitors the data periodically and alert the database host provider in cases of any anomalies detected.
ML algorithms: Machine Learning algorithms are used to calculate the accuracy of readings taken out. Machine Learning Library contain many algorithms that are used to derive the best accuracy from the read data.
Accuracy: states the Accuracy of reading, that will be calculated by using the appropriate machine learning algorithms such as Random Forest Classifier, Logistic regression, etc.
User: Cloud will present the calculated Accuracy to the authorized members of the User. User receive alert of the Skin problem.

4.2.5 Collaboration Diagram

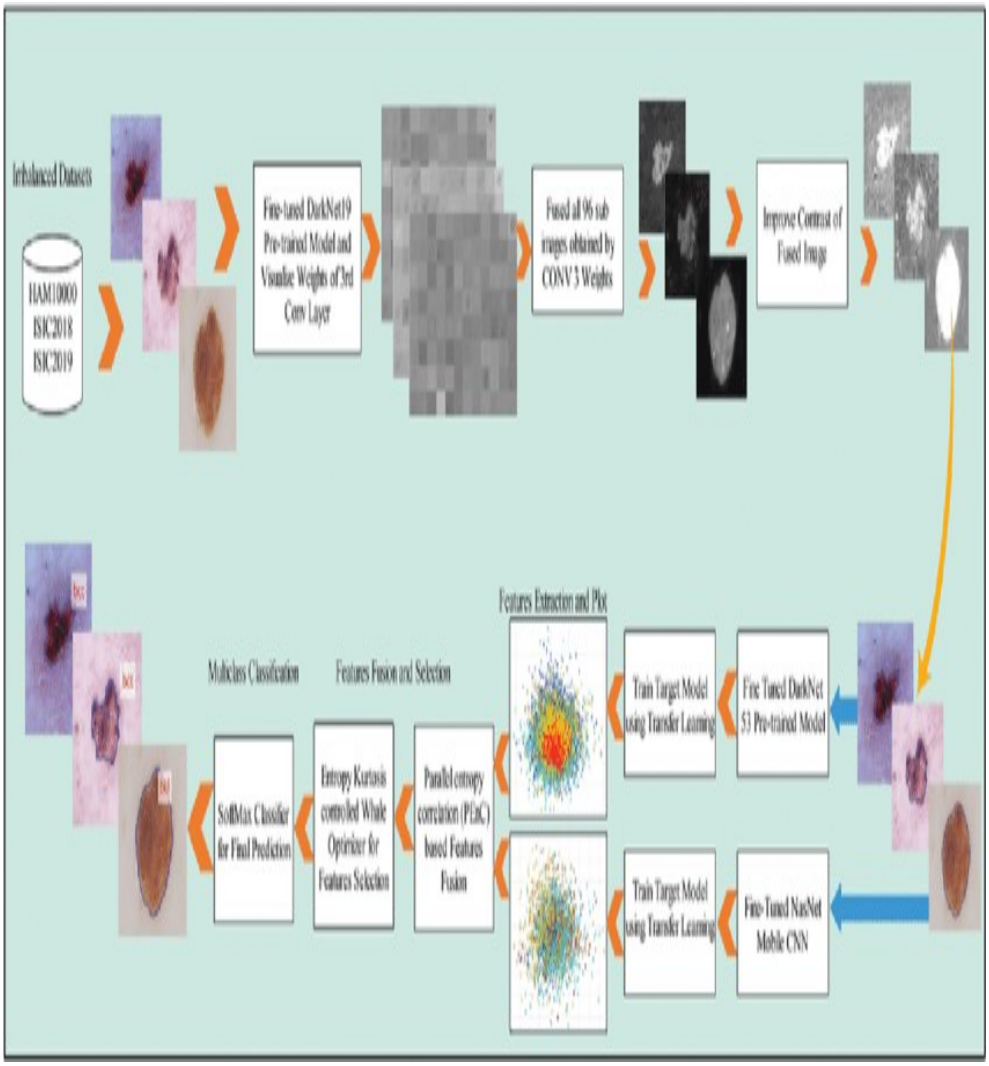

Figure 4.6: **Collaboration Diagram**

The figure 4.6 shows the collabration diagram which depicts the structure of the project.The logistic regression algorithm is mainly used for classification tasks where the goal is to predict the probability that an instance belongs to a given class or not. A decision tree is one of the most powerful tools used for both classification and regression tasks. It builds a flowchart-like tree structure where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label. The K-Nearest Neighbors (KNN) algorithm is a robust and intuitive machine learning method employed to tackle classification and regression problems. By capitalizing on the concept of similarity, KNN predicts the label or value of a new data point by considering its K closest neighbors in the training dataset. The SVM method draws a graph and takes the plane with the most area.

4.2.6 Activity Diagram

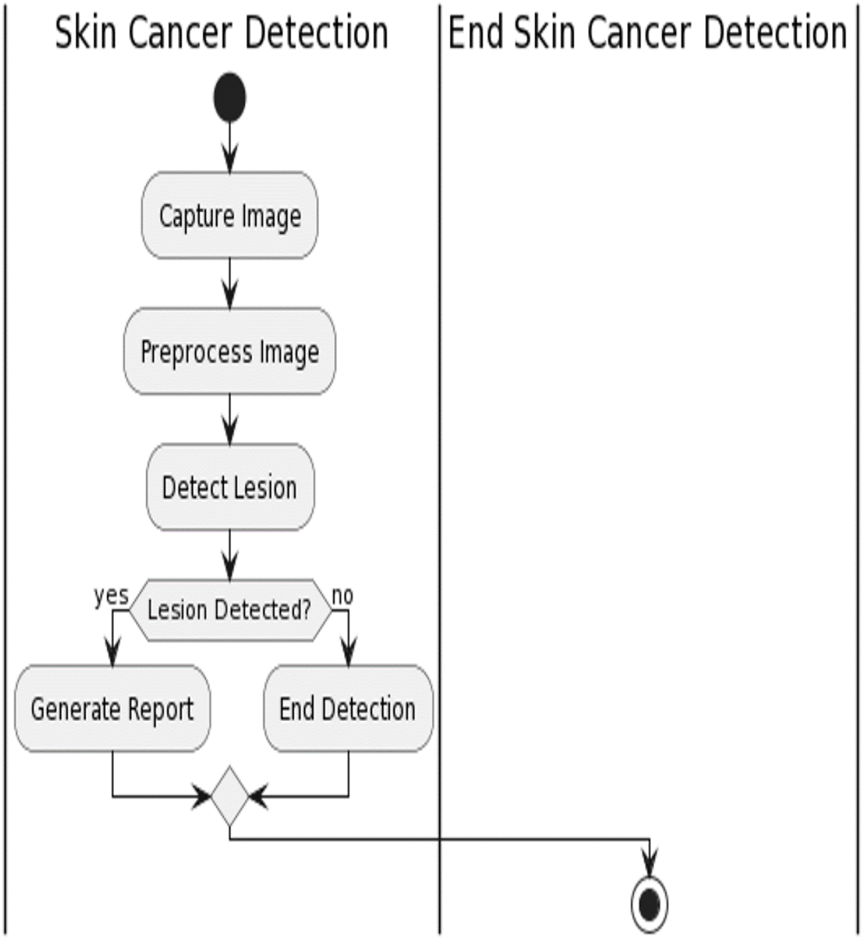

Figure 4.7: **Activity Diagram**

The figure 4.7 shows the activity diagram which depicts the flow of events of skin cancer detection.First, data is collected and performed preprocessing. The data is trained info is tested and checked whether blocking has occurred or not. If the blocking, the idea is to inform the user so that they will assign to user to skin problem, if not there would be no problem.

4.3 Algorithm & Pseudo Code

4.3.1 Algorithm

Step 1: Start.
Step 2: Clean and load the dataset into the code.
Step 3: Generate and train the dataset using Machine Learning Algorithms.
Step 4: Run the unit code.
Step 5: Visualize the efficiency and accuracy of the graph.
Step 6: Stop.

4.3.2 Pseudo Code

```

# Import necessary libraries
import pandas as pd
import numpy as np
import seaborn as sns
import plotly.express as px
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model selection import train test split
from sklearn.metrics import accuracy score, confusion matrix
from sklearn.linear model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier

# Load the dataset
df = pd.read csv("/content/drive/Colab Notebooks/Minor 1/waterpotability.csv")

# Data preprocessing and visualization
# ...

# Handle missing values
# ...

# Feature scaling
# ...

# Split the dataset into training and testing sets
# ...

# Model training and evaluation

```

```
# Logistic Regression
model lr = LogisticRegression()
model lr.fit(x-train, y-train)
predlr = model-lr.predict(x-test)
accuracylr = accuracyscore(y-test, predlr)

# Decision Tree
model-dt = DecisionTreeClassifier(max-depth=4)
model-dt.fit(x-train, y-train)
pred-dt = model-dt.predict(x-test)
accuracy-dt = accuracy-score(y-test, pred-dt)

# Random Forest
modelrf = RandomForestClassifier()
modelrf.fit(xtrain, ytrain)
predrf = modelrf.predict(xtest)
accuracyrf = accuracyscore(ytest, predrf)

# K-Nearest Neighour
modelknn = K-NearestClassifier(nneighbors=11)
modelknn.fit(xtrain, ytrain)
predknn = modelknn.predict(xtest)
accuracyknn = accuracyscore(ytest, predknn)

# Support Vector Machine (SVM)
modelsvm = SVC(kernel="rbf")
modelsvm.fit(xtrain, ytrain)
predsvm = modelsvm.predict(xtest)
accuracysvm = accuracyscore(ytest, predsvm)

# AdaBoost
modelada = AdaBoostClassifier(nestimators=200, learningrate=0.03)
modelada.fit(xtrain, ytrain)
predda = modelada.predict(xtest)
accuracyada = accuracyscore(ytest, predada)

# XGBoost
modelxgb = XGBClassifier(nestimators=100, learningrate=0.04)
modelxgb.fit(xtrain, ytrain)
predxgb = modelxgb.predict(xtest)
accuracyxgb = accuracyscore(ytest, predxgb)

# Create a dataframe to store model accuracies
models = pd.DataFrame({
    "Model": ["Logistic Regression", "Decision Tree", "Random Forest",
              "KNN", "SVM", "AdaBoost", "XGBoost"],
    "Accuracy": [accuracylr, accuracydt, accuracyrf,
                 accuracyknn, accuracysvm, accuracyada, accuracyxgb]
})
# Logistic Regression
```

```
model lr = LogisticRegression()
model lr.fit(x-train, y-train)
predlr = model-lr.predict(x-test)
accuracylr = accuracyscore(y-test, predlr)

# Decision Tree
model-dt = DecisionTreeClassifier(max-depth=4)
model-dt.fit(x-train, y-train)
pred-dt = model-dt.predict(x-test)
accuracy-dt = accuracy-score(y-test, pred-dt)

# Random Forest
modelrf = RandomForestClassifier()
modelrf.fit(xtrain, ytrain)
predrf = modelrf.predict(xtest)
accuracyrf = accuracyscore(ytest, predrf)

# K-Nearest Neighour
modelknn = K-NearestClassifier(nneighbors=11)
modelknn.fit(xtrain, ytrain)
predknn = modelknn.predict(xtest)
accuracyknn = accuracyscore(ytest, predknn)

# Visualize model accuracies
# ...

# Display sorted model accuracies
# ...
```

4.4 Module Description

4.4.1 Collection Of Data:

```
C: > Android-App-Skin-Cancer-Detector > Training > Training_CNN_Model.ipynb > from google.colab import drive
+ Code + Markdown ...                                                                    Select Kernel
    else:
      feature_extractor.trainable = False
[101]                                                                                          Python

print("Building model with", MODULE_HANDLE)
model = tf.keras.Sequential([
    feature_extractor,
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dropout(rate=0.2),
    tf.keras.layers.Dense(train_generator.num_classes, activation='softmax',
                          kernel_regularizer=tf.keras.regularizers.l2(0.0001))
])
#model.build((None,)+IMAGE_SIZE+(3,))

model.summary()
[102]                                                                                          Python

... Building model with https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/2
    Model: "sequential_4"
    _________________________________________________________________
    Layer (type)                 Output Shape              Param #
    =================================================================
    keras_layer_10 (KerasLayer)  (None, 1280)              2257984
    _________________________________________________________________
    flatten_11 (Flatten)         (None, 1280)              0
    _________________________________________________________________
    dense_22 (Dense)             (None, 512)               655872
    _________________________________________________________________
    dropout_11 (Dropout)         (None, 512)               0
    _________________________________________________________________
    dense_23 (Dense)             (None, 2)                 1026
    =================================================================
    Total params: 2,914,882
    Trainable params: 656,898
```

Figure 4.8: **Collection of Data**

The figure 4.8 shows the Collection of Data .Data collection is the systematic process of gathering and recording information from various sources, such as surveys, observations, or sensing algorithm. It forms the foundation for meaningful analysis and insights in research, decision-making, and problem-solving.

4.4.2 Processing Of Data:

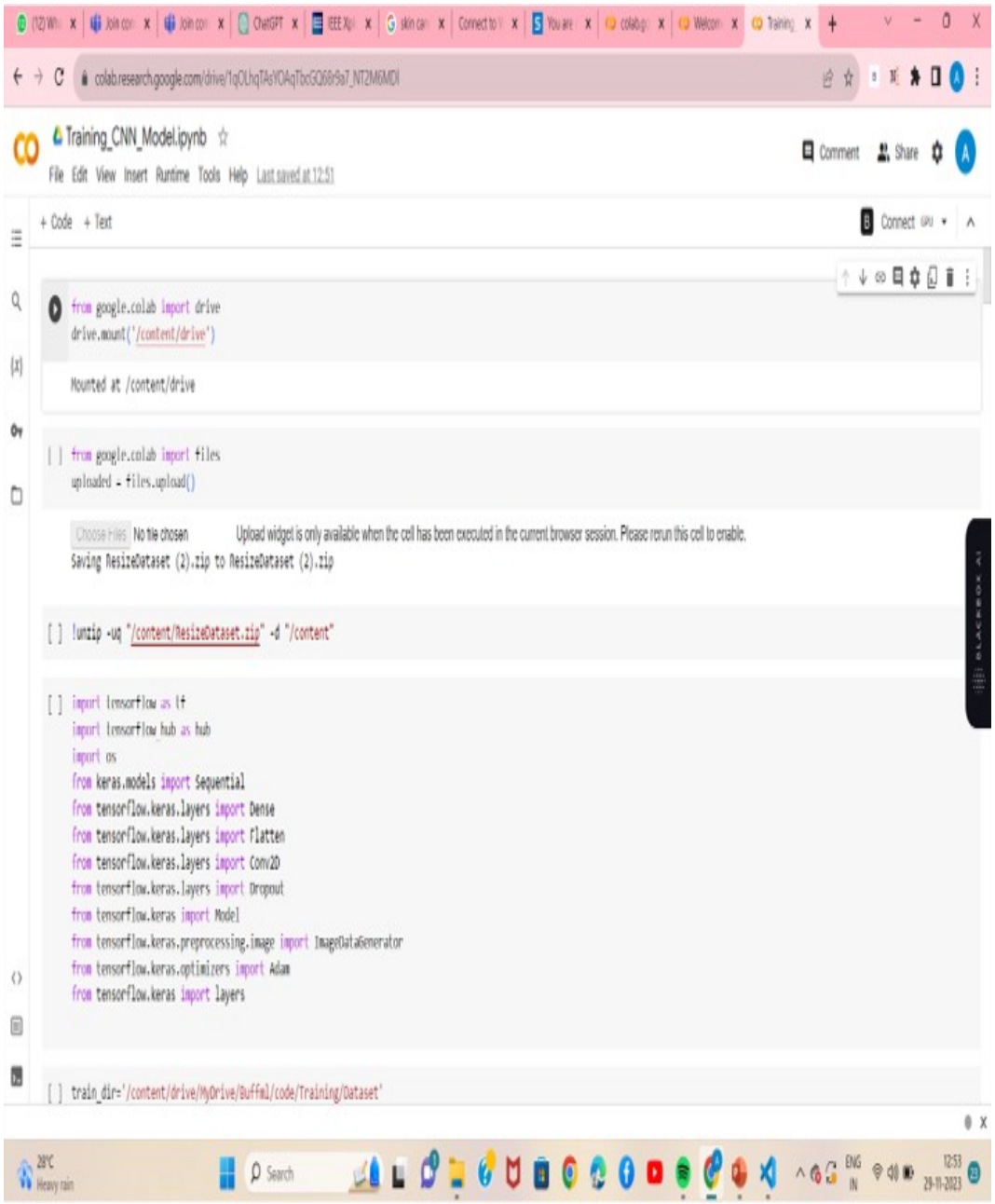

Figure 4.9: **Processing of Data**

The figure 4.9 shows the Proccesing of Data .Data preprocessing involves cleaning, transforming, and organizing raw data to enhance its quality and usability. Common tasks include handling missing values, scaling features, and encoding categorical variables, ensuring the data is suitable for analysis or machine learning applications.

4.4.3 Implementing Algorithms:

Decision Tree:

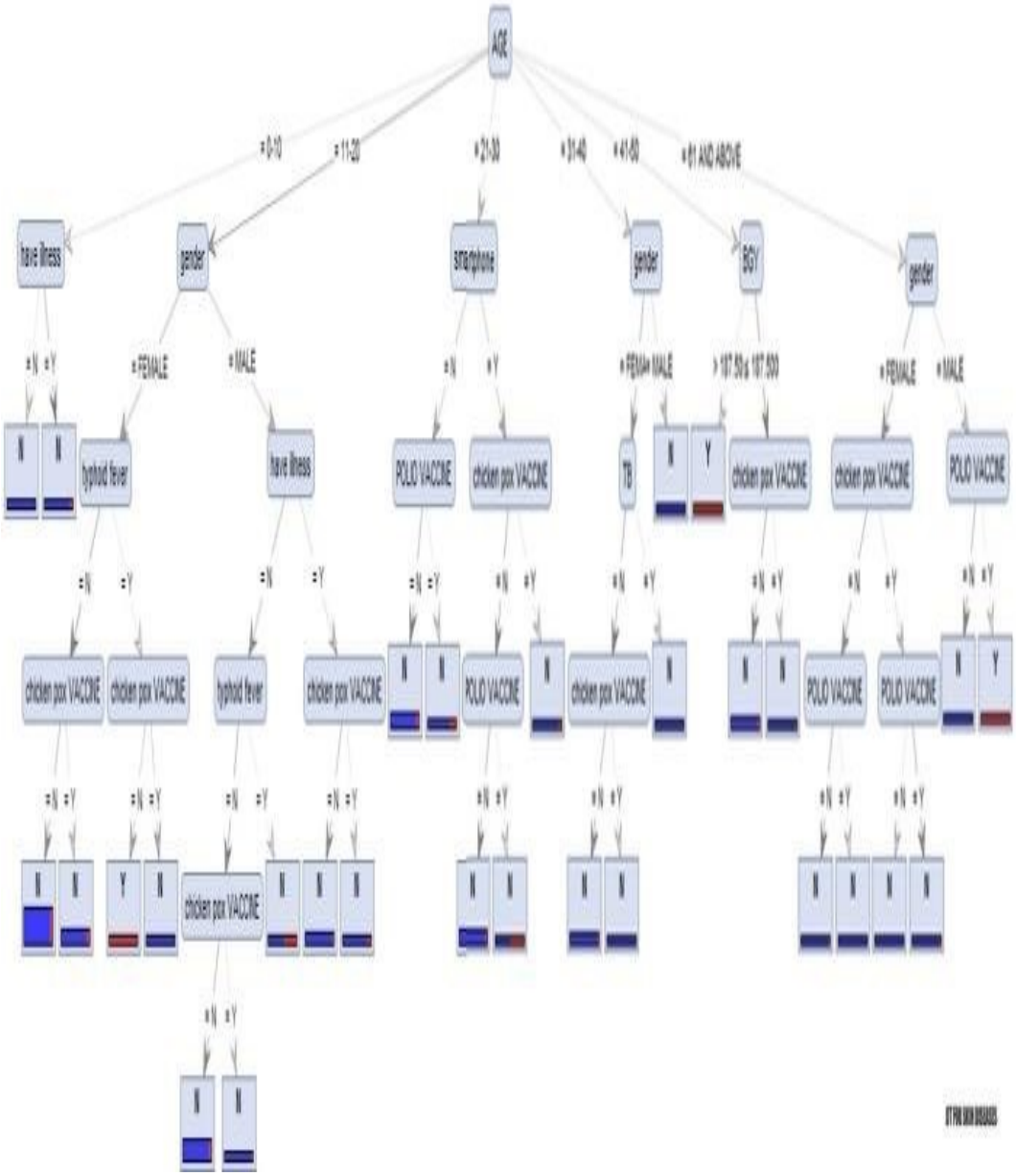

Figure 4.10: **Decision Tree Algorithm**

The figure 4.10 depicts the Decision Tree Algorithm.A decision tree is a supervised machine learning algorithm that models decisions as a tree-like structure, consisting of nodes representing features, branches as decision rules, and leaves as outcomes. It recursively partitions data based on feature values, making it intuitive for classification and regression tasks.

Logistic Regression:

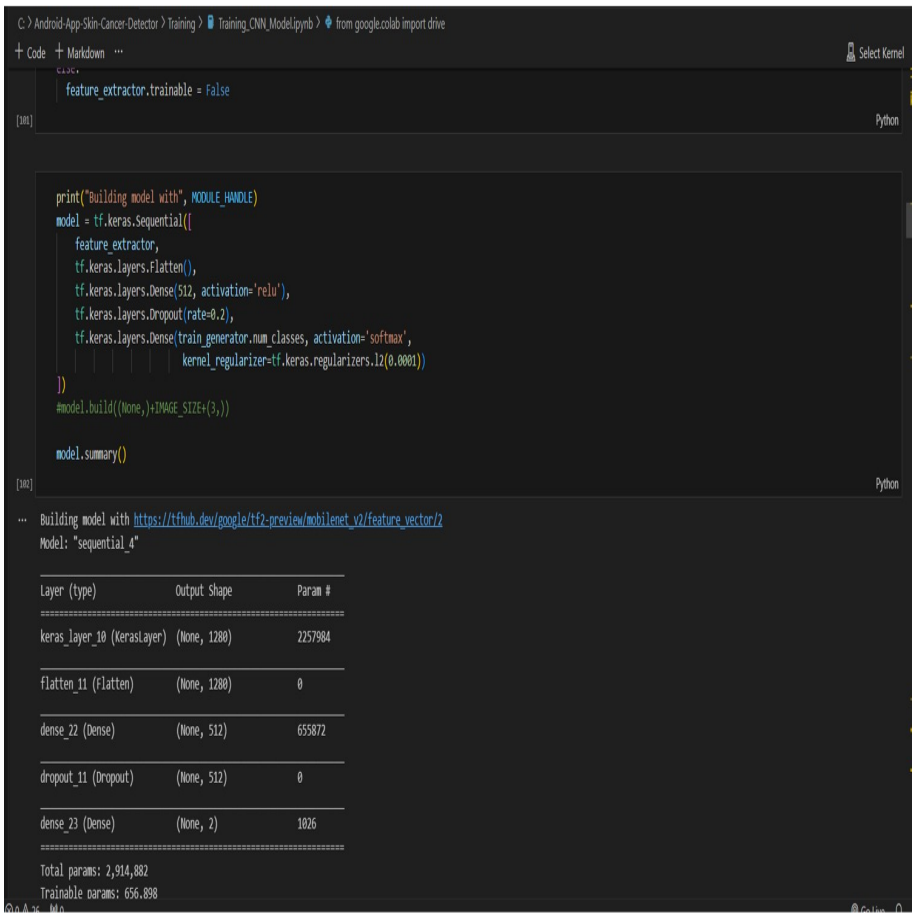

Figure 4.11: **Logistic Regression**

The figure 4.11 depicts the Logistic Regression.Logistic Regression is a statistical method used for binary classification problems, predicting outcomes between two possible categories. It models the probability of an instance belonging to a particular class through a logistic function. The algorithm is widely employed in fields such as medicine and finance for its simplicity and interpretability, making it a fundamental tool in machine learning.

Random Forest Classifier:

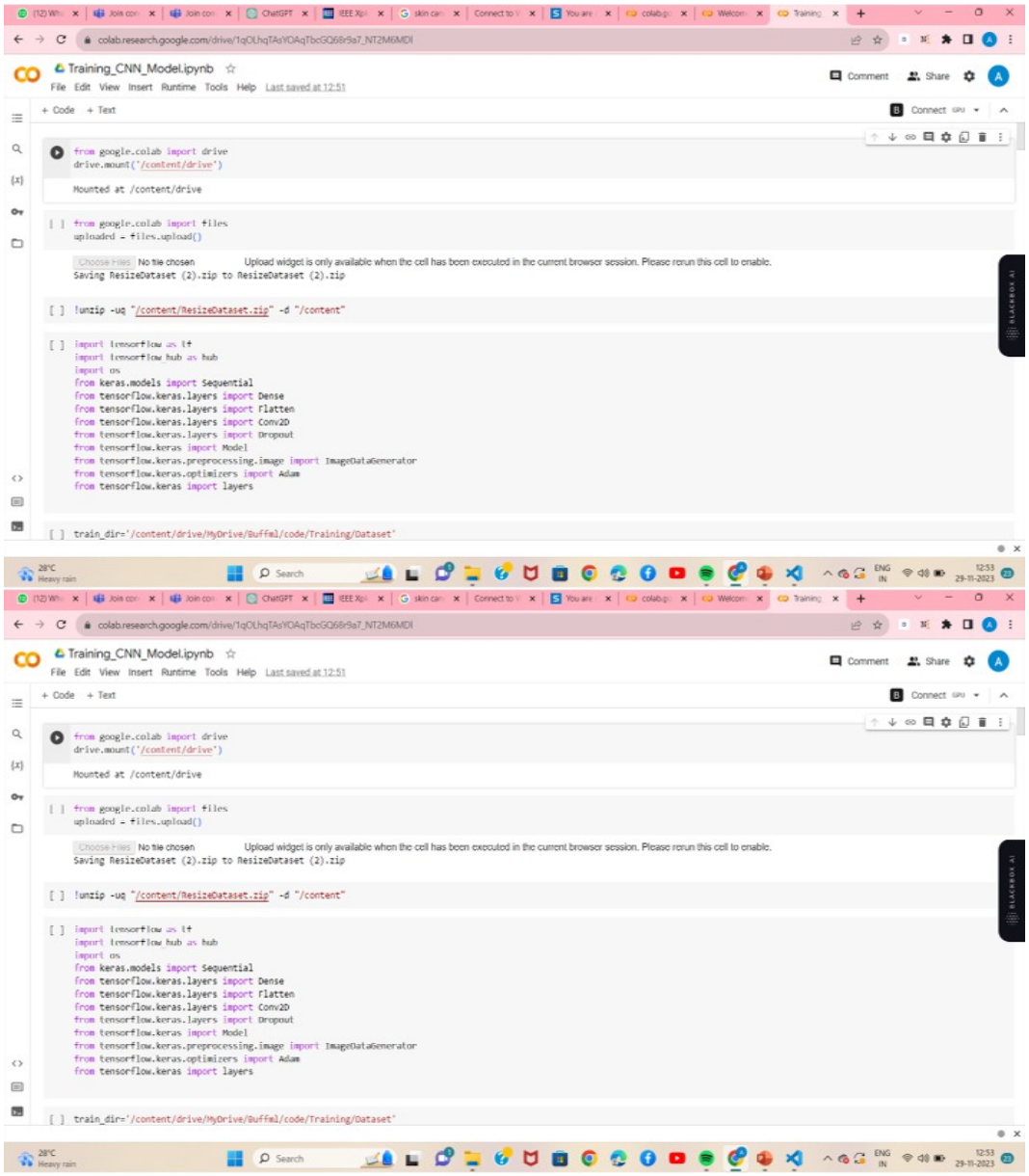

Figure 4.12: **Random Forest Classifier**

The figure 4.12 depicts the Random Forest Classifier.Random Forest is an ensemble learning algorithm that constructs multiple decision trees during training and outputs the mode of the classes for classification or the mean prediction for regression. It introduces randomness by using bootstrapped samples and random subsets of features for each tree, enhancing model diversity. Known for high accuracy and robustness, Random Forest mitigates overfitting and is suitable for various applications, from image classification to financial forecasting.

k-Nearest Neighbour:

```
C: > Android-App-Skin-Cancer-Detector > Training > Training_CNN_Model.ipynb > from google.colab import drive
+ Code  + Markdown  ...                                                                    Select Kernel
    else:
      feature_extractor.trainable = False
[101]                                                                                          Python

print("Building model with", MODULE_HANDLE)
model = tf.keras.Sequential([
    feature_extractor,
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dropout(rate=0.2),
    tf.keras.layers.Dense(train_generator.num_classes, activation='softmax',
                          kernel_regularizer=tf.keras.regularizers.l2(0.0001))
])
#model.build((None,)+IMAGE_SIZE+(3,))

model.summary()
[102]                                                                                          Python

... Building model with https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/2
    Model: "sequential_4"
    _________________________________________________________________
    Layer (type)                 Output Shape              Param #
    =================================================================
    keras_layer_10 (KerasLayer)  (None, 1280)              2257984
    _________________________________________________________________
    flatten_11 (Flatten)         (None, 1280)              0
    _________________________________________________________________
    dense_22 (Dense)             (None, 512)               655872
    _________________________________________________________________
    dropout_11 (Dropout)         (None, 512)               0
    _________________________________________________________________
    dense_23 (Dense)             (None, 2)                 1026
    =================================================================
    Total params: 2,914,882
    Trainable params: 656,898
```

Figure 4.13: **K-Nearest Neighbours**

The figure 4.13 depicts the K-Nearest Neighbour.K Nearest Neighbors (KNN) is a simple and versatile supervised machine learning algorithm used for both classification and regression tasks. It classifies a new data point based on the majority class of its k nearest neighbors in the feature space. KNN's effectiveness depends on the choice of the distance metric and the value of k, and it is non-parametric, making it suitable for various data distributions and patterns.

4.5 Steps to execute/run/implement the project

4.5.1 Run Python :

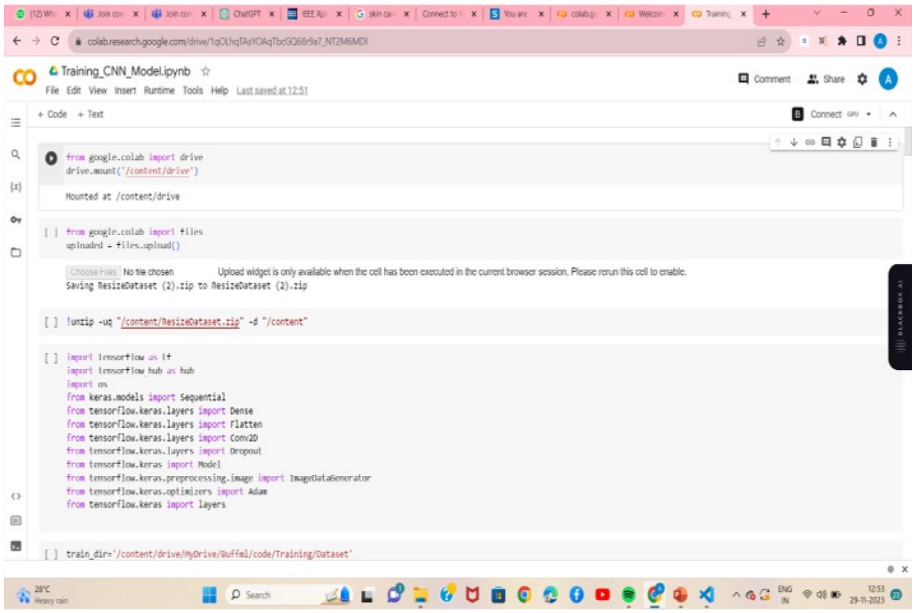

Figure 4.14: **Google Colab Environment**

The figure 4.14 depicts the Google Colab Environment. Open Google Colab and run it using the configured interpreter, which runs on a virtual environment.

4.5.2 Loading Data Set:

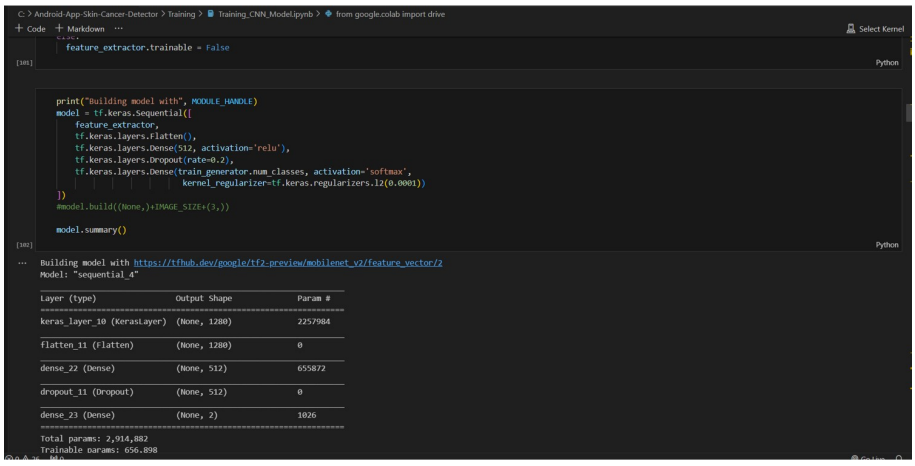

Figure 4.15: **Loading Data Set**

The figure 4.15 represents the Loading Data Set.Load the dataset and perform preprocessing, train, and test it.

4.5.3 Decision Tree:

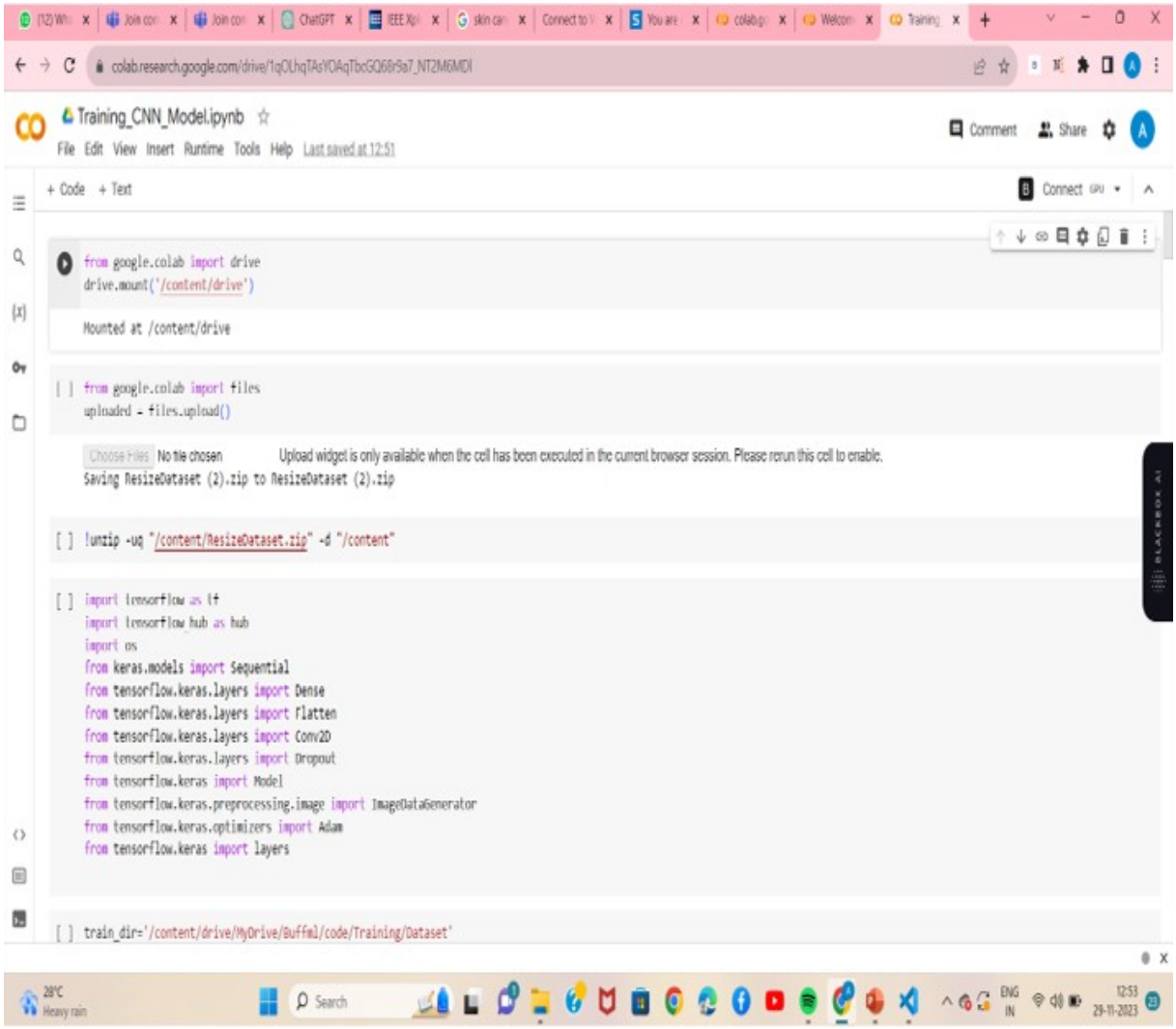

Figure 4.16: **Decision Tree**

The figure 4.16 depicts the Decision Tree.A decision tree is a flowchart-like structure in which each internal node represents a decision or test on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label or a decision taken after evaluating all the attributes.

Logistic Regression:

Features	Values
Standard vector	20.8532
Diameter	2.1480
Asymmetry index	1
Color values of r, g, b	37.0471, 23.2337, 27.0009
Auto correlation	2.520931623931624e + 01
Contrast	1.228632478632479e-01
Correlation	9.894224944536026e-01
Energy	1.669194389655928e-01
Entropy	2.156049329513495e + 00
Homogeneity	9.411574074074074e-01

Figure 4.17: **Logistic Regression**

The figure 4.17 represents the logistic Regression.Logistic Regression is a statistical method used for modeling the probability of a binary outcome. It is a type of regression analysis that is well-suited for predicting the probability of an event occurring as a function of one or more predictor variables. Despite its name, logistic regression is used for classification problems, not regression problems.Binary Classification: Logistic regression is primarily used for binary classification problems where the dependent variable (response) has two possible outcomes, often coded as 0 and 1.

Random Forest Classifier:

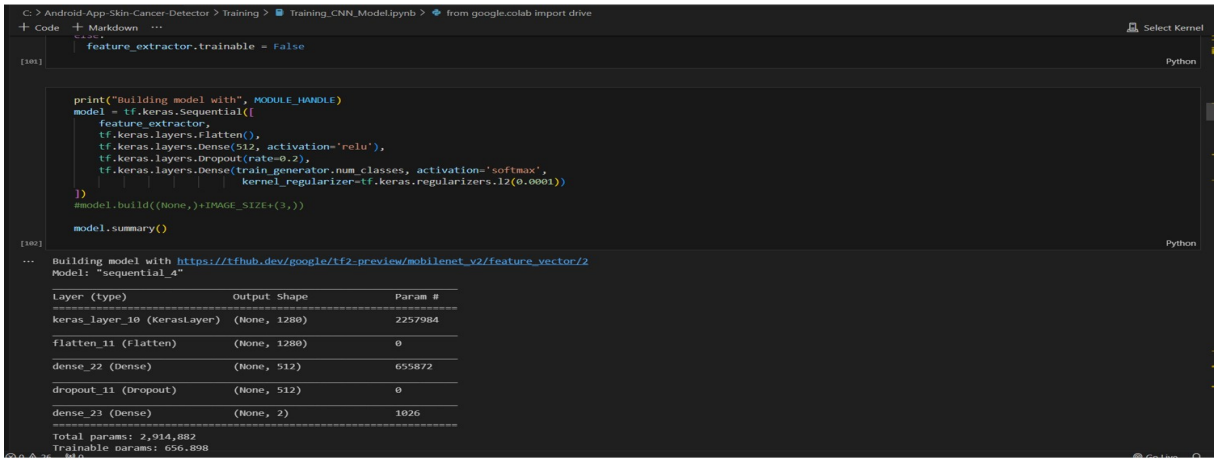

Figure 4.18: **Random Forest Classifier**

The figure 4.18 represents the Random Forest Classifier . Random Forest is an ensemble learning method that operates by constructing a multitude of decision trees at training time and outputs the class that is the mode of the classes (classification)

or mean prediction (regression) of the individual trees. It's a popular and powerful machine learning algorithm known for its versatility and effectiveness.The basic building block of a Random Forest is a decision tree. A decision tree is a flowchart-like structure where each internal node represents a decision based on an attribute, each branch represents the outcome of the decision, and each leaf node represents the class label (in the case of classification) or the predicted value (in the case of regression).The "Random" in Random Forest comes from the fact that it introduces randomness during the tree-building process. At each node of a tree, a random subset of features is considered for splitting. This helps in decorrelating the trees, making the ensemble more robust and less prone to overfitting.Another source of randomness is introduced by using bootstrapped samples. Instead of training each tree on the entire dataset, each tree is trained on a random subset of the data with replacement (bootstrap sample)

K-Nearest Neighbour:

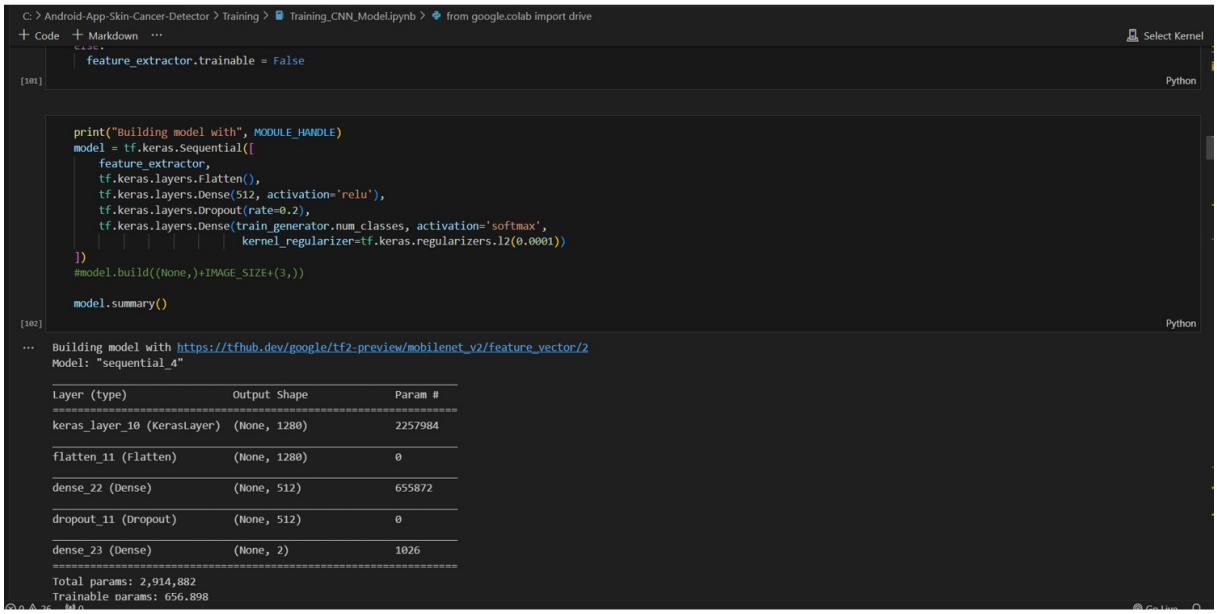

Figure 4.19: **K-Nearest Neighbour**

The figure 4.19 represents the K-Nearest Neighbour.K-Nearest Neighbors (KNN) is a simple and popular supervised machine learning algorithm used for both classification and regression tasks.

Chapter 5

IMPLEMENTATION AND TESTING

5.1 Input and Output

5.1.1 Permissions to Dataset

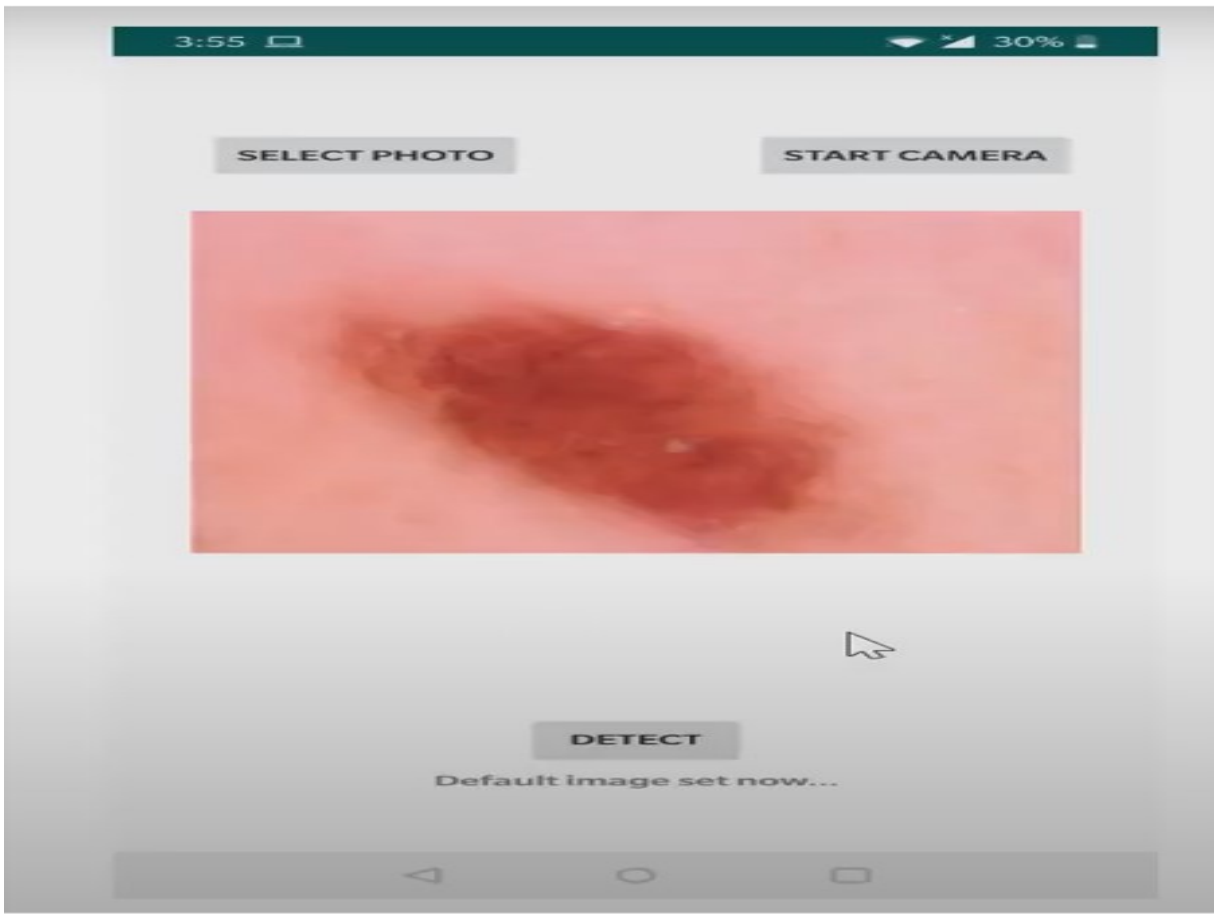

Figure 5.1: **Permissions to Dataset**

The figure 5.1 depicts the Permissions to Dataset.In this project the dataset consists of 1500 records in the form of excel sheet in csv format as input to train algorithm.

5.1.2 Processing of the Dermal

Here the output is in the form of accuracy where the machine learning algorithms give the accuracy by using the data from the dataset and finally, as an output the accuracies will be obtained so that efficient machine learning algorithm can be taken from the set of algorithms.

5.2 Testing

5.3 Types of Testing

5.3.1 Unit Testing

Unit test is used to ensure that each modular component of the project is working. The smallest unit of the software design is the subject of unit testing. The mentioned project underwent a progression examination of unit testing

Input

Logicstic Regression:

```
    from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Initialize the Logistic Regression model
model_lr = LogisticRegression()

# Training the model
model_lr.fit(x_train, y_train)

# Make predictions on the test set
pred_lr = model_lr.predict(x_test)

# Calculate accuracy score
accuracy_score_lr = accuracy_score(y_test, pred_lr)
accuracy_score_lr *= 100  # Multiply by 100 to get percentage

print(f"Accuracy of Logistic Regression Model: {accuracy_score_lr:.2f}%")
```

Random Forest Classifier:

```
from sklearn.ensemble import RandomForestClassifier

# Creating model object
model_rf = RandomForestClassifier()
```

```
#Training Model RF
model_rf.fit(x_train,y_train)
#Making Prediction
pred_rf = model_rf.predict(x_test)
accuracy_score_rf = accuracy_score(y_test,pred_rf)
accuracy_score_rf*100
cm3 = confusion_matrix(y_test,pred_rf)
cm3
```

Decision Tree :

```
from sklearn.tree import DecisionTreeClassifier

# creating the model object
model_dt = DecisionTreeClassifier(max_depth = 4)
#Training of decision tree
model_dt.fit(x_train,y_train)
#Making prediction using Decision Tree
pred_dt = model_dt.predict(x_test)
accuracy_score_dt = accuracy_score(y_test,pred_dt)
accuracy_score_dt*100
cm2 = confusion_matrix(y_test,pred_dt)
cm2
```

K-Nearest Neighbour:

```
from sklearn.neighbors import KNeighborsClassifier

#Creating Model object
model_knn = KNeighborsClassifier()

for i in range(4,15):
  model_knn = KNeighborsClassifier(n_neighbors=i)
  model_knn.fit(x_train,y_train)
  pred_knn = model_knn.predict(x_test)
  accuracy_score_knn = accuracy_score(y_test,pred_knn)
  print(i,accuracy_score_knn)

model_knn = KNeighborsClassifier(n_neighbors=11)
model_knn.fit(x_train,y_train)
pred_knn = model_knn.predict(x_test)
accuracy_score_knn = accuracy_score(y_test,pred_knn)
print(accuracy_score_knn*100)
```

Test result

The unit testing findings were encouraging.

5.3.2 Integration Testing

Integration testing is a methodical methodology for building the software architecture while also running tests to detect faults related to the interface. Integration testing, in other words, is the comprehensive testing the products collection of modules.

Input

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
plt.figure(figsize=(10,6))
sns.heatmap(df.isnull())
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(),annot=True)
fig, ax=plt.subplots(ncols=5, nrows=2, figsize= (20,10))
ax=ax.flatten()
index=0
for col,value in df.items():
  sns.boxplot(y=col,data=df,ax=ax[index])
  index +=1
df["ph"] = df["ph"].fillna(df["ph"].mean())
df["Sulfate"] = df["Sulfate"].fillna(df["Sulfate"].mean())
df["Trihalomethanes"] =
x = df.drop("Potability",axis=1)
y = df["Potability"]
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
accuracy_score_lr = accuracy_score(y_test,pred_lr)
accuracy_score_lr*100
models = pd.DataFrame({
    "Model": ["Logistic Regression",
              "Decision Tree",
              "Random Forest",
              "KNN"] ,
    "Accuracy Score" : [accuracy_score_lr,accuracy_score_dt,accuracy_score_rf,
                        accuracy_score_knn]
```

```
})
```

Test result

For the described project, a sequential analysis of integration testing was undertaken. The findings of the integration testing were positive and encouraging.

Figure 5.2: **Testing ML Models**

5.3.3 System Testing

System testing is a sort of software testing that is carried on a whole integrated system in order to assess the system's compliance with the requirements. Passed components are used as input in system testing. Integration testing is used to find any discrepancies between the unit that are linked together. System testing looks for flaws in both the individual components and the entire system. The behaviour of a component or a system when it is tested is the result of system testing.

Input

```
from sklearn import metrices
# Initialize the Logistic Regression model
model_lr = LogisticRegression()

# Training the model
model_lr.fit(x_train, y_train)

# Make predictions on the test set
pred_lr = model_lr.predict(x_test)

# Calculate accuracy score
accuracy_score_lr = accuracy_score(y_test, pred_lr)
accuracy_score_lr *= 100  # Multiply by 100 to get percentage

model_rf = RandomForestClassifier()
#Training Model RF
model_rf.fit(x_train,y_train)
#Making Prediction
pred_rf = model_rf.predict(x_test)
accuracy_score_rf = accuracy_score(y_test,pred_rf)
accuracy_score_rf*100
cm3 = confusion_matrix(y_test,pred_rf)
```

```
cm3
from sklearn.tree import DecisionTreeClassifier

# creating the model object
model_dt = DecisionTreeClassifier(max_depth = 4)
#Training of decision tree
model_dt.fit(x_train,y_train)
#Making prediction using Decision Tree
pred_dt = model_dt.predict(x_test)
accuracy_score_dt = accuracy_score(y_test,pred_dt)
accuracy_score_dt*100
cm2 = confusion_matrix(y_test,pred_dt)
cm2
df["ph"] = df["ph"].fillna(df["ph"].mean())
df["Sulfate"] = df["Sulfate"].fillna(df["Sulfate"].mean())
df["Trihalomethanes"] =
x = df.drop("Potability",axis=1)
y = df["Potability"]
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
accuracy_score_lr = accuracy_score(y_test,pred_lr)
accuracy_score_lr*100
models = pd.DataFrame({
    "Model": ["Logistic Regression",
              "Decision Tree",
              "Random Forest",
              "KNN"] ,

    "Accuracy Score" : [accuracy_score_lr,accuracy_score_dt,accuracy_score_rf,
                        accuracy_score_knn]
})
```

5.3.4 Test Result

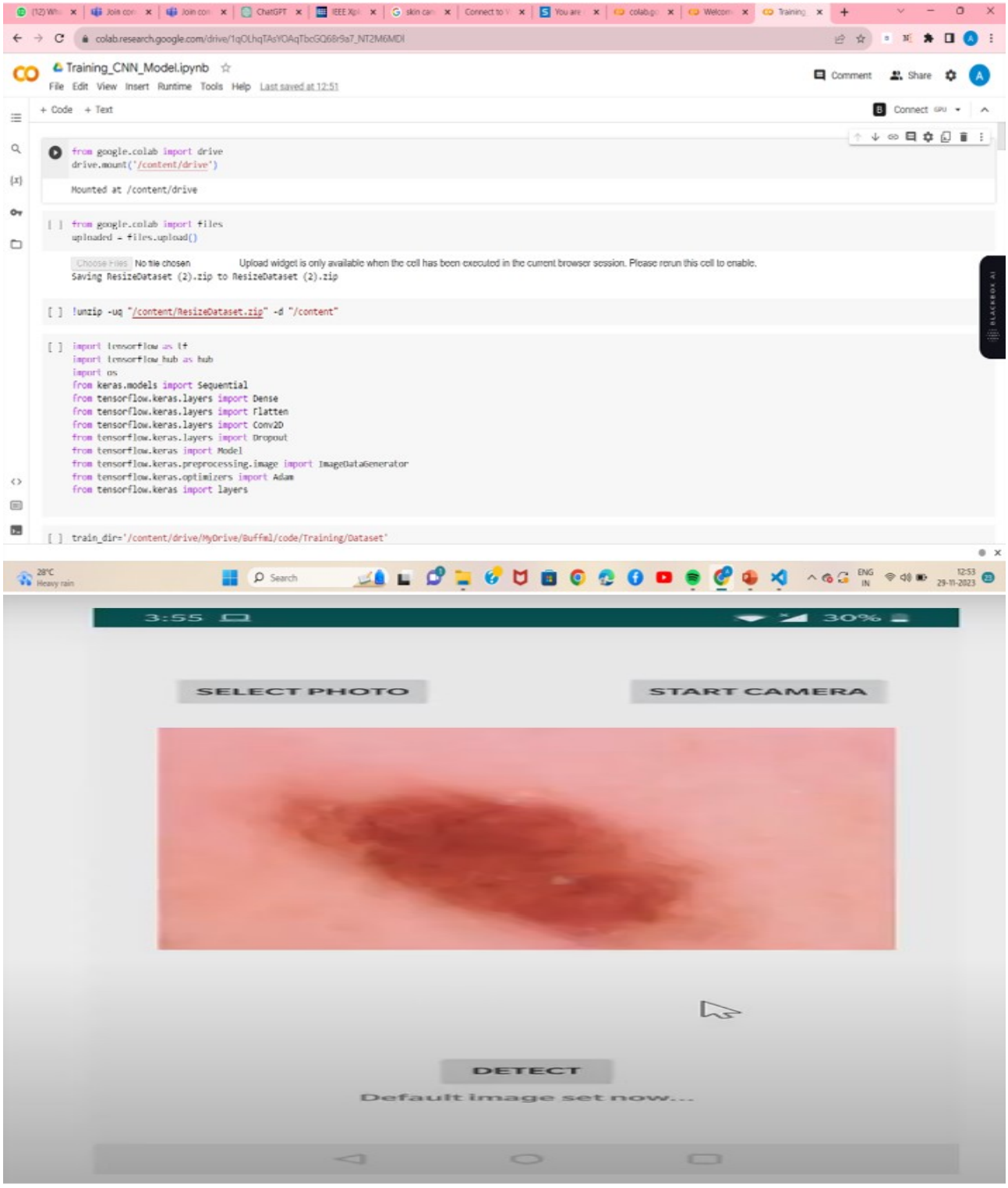

Figure 5.3: **Test Image**

The figure 5.3 depicts the Permissions to Dataset.In this project the dataset con- sists of 1500 records in the form of excel sheet in csv format as input to train algorithm.

Chapter 6

RESULTS AND DISCUSSIONS

6.1 Efficiency of the Proposed System

The proposed system is based on the Random forest Algorithm that creates many decision trees. Accuracy of proposed system is done by using random forest gives the ouput approximately 76 to 78 percent. Random forest implements many decision trees and also gives the most accurate output when compared to the decision tree. Random Forest algorithm is used in the two phases.

Firstly, the RF algorithm extracts subsamples from the original samples by using the bootstrap resampling method and creates the decision trees for each testing sample and then the algorithm classifies the decision trees and implements a vote with the help of the largest vote of the classification as a final result of the classification.

The random Forest algorithm always includes some of the steps as follows: Selecting the training dataset:Using the bootstrap random sampling method we can derive the K training sets from the original dataset properties using the size of all training set the same as that of original training dataset. Building the random forest algorithm: Creating a classification regression tree each of the bootstrap training set will generate the K decision trees to form a random forest model, uses the trees that are not pruned. Looking at the growth of the tree, 31 this approach is not chosen the best feature as the internal nodes for the branches but rather the branching process is a random selection of all the trees gives the best features.

6.2 Comparison of Existing and Proposed System

Existing system:(Decision tree)

In the Existing system, we implemented a decision tree algorithm that predicts whether to grant the loan or not. When using a decision tree model, it gives the training dataset the accuracy keeps improving with splits. We can easily overfit the dataset and doesn't know when it crossed the line unless we are using the cross validation. The advantages of the decision tree are model is very easy to interpret we can know that the variables and the value of the variable is used to split the data. But the accuracy of decision tree in existing system gives less accurate output that is less

when compared to proposed system.

Proposed system:(Random forest algorithm)

Random forest algorithm generates more trees when compared to the decision tree and other algorithms. We can specify the number of trees we want in the forest and also we also can specify maximum of features to be used in the each of the tree. But, we cannot control the randomness of the forest in which the feature is a part of the algorithm. Accuracy keeps increasing as we increase the number of trees but it becomes static at one certain point. Unlike the decision tree it won't create more biased and decreases variance. Proposed system is implemented using the Random forest algorithm so that the accuracy is more when compared to the existing system.

6.3 Sample Code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
df = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/Minor 1/water_potability.csv")
df.head()
plt.figure(figsize=(10,6))
sns.heatmap(df.isnull())
fig, ax=plt.subplots(ncols=5, nrows=2, figsize= (20,10))
ax=ax.flatten()
index=0
for col,value in df.items():
  sns.boxplot(y=col,data=df,ax=ax[index])

  index +=1

fig = px.pie(df,names = "Potability",hole = 0.4,template ="plotly_dark")
fig.show()
df["ph"] = df["ph"].fillna(df["ph"].mean())
df["Sulfate"] = df["Sulfate"].fillna(df["Sulfate"].mean())
df["Trihalomethanes"] = df["Trihalomethanes"].fillna(df["Trihalomethanes"].mean())

#Logistic Regression
from sklearn.linear_model import LogisticRegression

#object of LR
```

```
model_lr = LogisticRegression()
#Training Model
model_lr.fit(x_train,y_train)
#Making Prediction
pred_lr = model_lr.predict(x_test)
# accuracy score
accuracy_score_lr = accuracy_score(y_test,pred_lr)
accuracy_score_lr*100

from sklearn.tree import DecisionTreeClassifier

# creating the model object
model_dt = DecisionTreeClassifier(max_depth = 4)
#Training of decision tree
model_dt.fit(x_train,y_train)
#Making prediction using Decision Tree
pred_dt = model_dt.predict(x_test)
accuracy_score_dt = accuracy_score(y_test,pred_dt)
accuracy_score_dt*100
#confusion matrix
cm2 = confusion_matrix(y_test,pred_dt)
cm2


from sklearn.ensemble import RandomForestClassifier

# Creating model object
model_rf = RandomForestClassifier()
#Training Model RF
model_rf.fit(x_train,y_train)
#Making Prediction
pred_rf = model_rf.predict(x_test)
accuracy_score_rf = accuracy_score(y_test,pred_rf)
accuracy_score_rf*100
cm3 = confusion_matrix(y_test,pred_rf)
cm3


from sklearn.neighbors import KNeighborsClassifier

#Creating Model object
model_knn = KNeighborsClassifier()
for i in range(4,15):
  model_knn = KNeighborsClassifier(n_neighbors=i)
  model_knn.fit(x_train,y_train)
  pred_knn = model_knn.predict(x_test)
  accuracy_score_knn = accuracy_score(y_test,pred_knn)
  print(i,accuracy_score_knn)
model_knn = KNeighborsClassifier(n_neighbors=11)
model_knn.fit(x_train,y_train)
```

```
pred_knn = model_knn.predict(x_test)
accuracy_score_knn = accuracy_score(y_test,pred_knn)
print(accuracy_score_knn*100)

#outputs
models = pd.DataFrame({
    "Model": ["Logistic Regression",
              "Decision Tree",
              "Random Forest",
              "KNN"] ,

    "Accuracy Score" : [accuracy_score_lr,accuracy_score_dt,accuracy_score_rf,
                        accuracy_score_knn]
})
models

#accuracy
sns.barplot(x="Accuracy Score",y= "Model",data=models)
models.sort_values(by="Accuracy Score",ascending= False)
```

Output

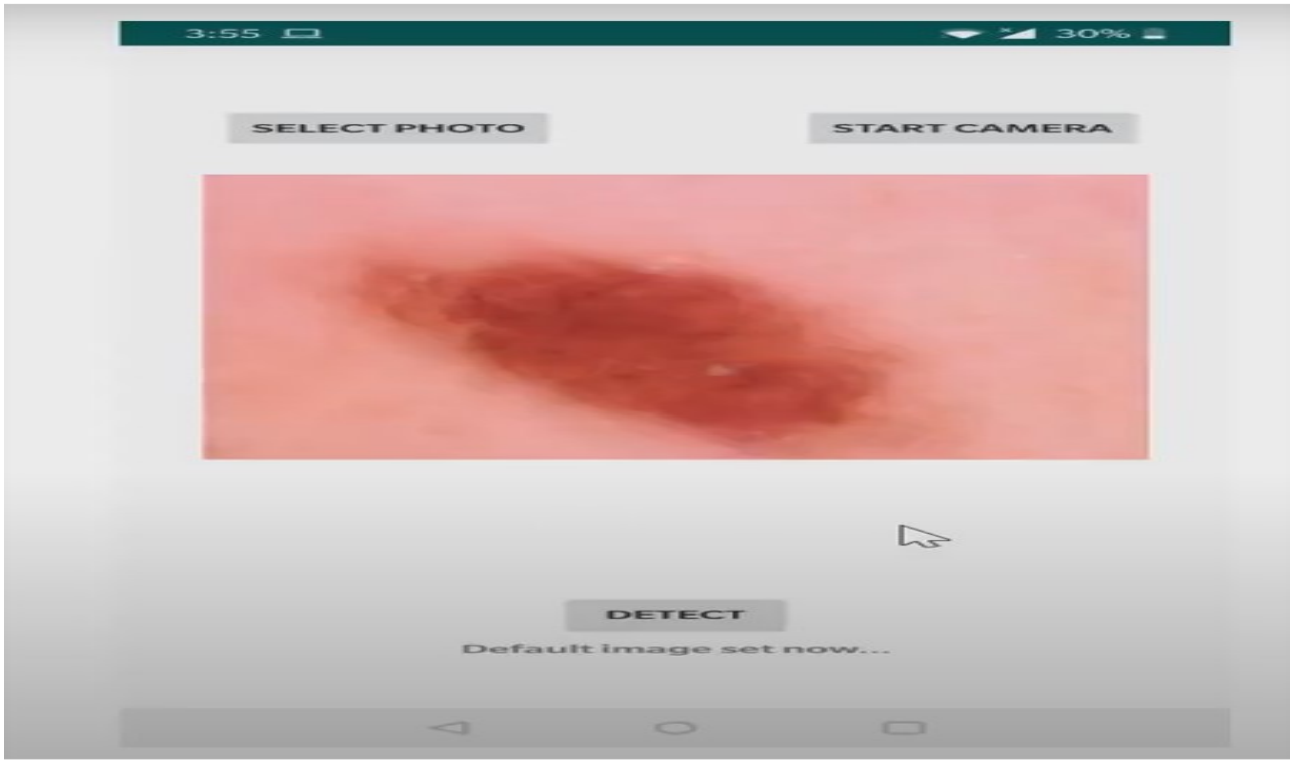

Figure 6.1: **Processed Skin Tissue**

The Figure 6.1 represents the Processed skin Tissue. The term ”processed skin tissue” generally refers to skin samples that have undergone some form of treatment, preparation, or modification for various purposes. These processed tissues are often used in medical, research, or therapeutic contexts. Here are a few scenarios where processed skin tissue might be involved.In dermatology and plastic surgery, processed skin tissue may be used in grafting procedures. This involves taking skin from one area of the body (donor site) and placing it on another area (recipient site) that requires skin replacement, such as in the case of burns or wounds.Skin tissue samples can be processed for research purposes. This may involve fixation, embedding, and sectioning for histological analysis. Researchers may study processed skin tissue to understand skin structure, diseases, or the effects of certain treatments.Skin tissue may be processed for cryopreservation, a technique that involves freezing the tissue at extremely low temperatures to preserve it for future use.

Chapter 7

CONCLUSION AND FUTURE ENHANCEMENTS

7.1 Conclusion

The integration of machine learning in skin cancer detection represents a transformative leap forward in the realm of dermatological diagnostics. The development and implementation of advanced algorithms, particularly Convolutional Neural Networks (CNNs), have showcased remarkable capabilities in accurately identifying and classifying skin lesions. The economic feasibility of such systems is underscored by their potential to revolutionize patient outcomes and healthcare practices.

The initial investments in infrastructure, data acquisition, and algorithm development are counterbalanced by the long-term benefits, including enhanced diagnostic accuracy, reduced treatment costs through early detection, and improved overall efficiency in clinical workflows.

The interpretability features integrated into these systems foster trust among healthcare professionals, a critical element in the adoption of cutting-edge technologies in clinical settings. As the field continues to evolve, embracing continuous monitoring, updates, and scalability, the economic feasibility of skin cancer detection using machine learning becomes increasingly evident, paving the way for a future where technology plays a pivotal role in advancing dermatological care.

The incorporation of interpretability features enhances transparency, addressing concerns in the medical community regarding the ”black-box” nature of machine learning models. As research and development continue to progress, the collaborative efforts of technologists, healthcare professionals, and researchers pave the way for a future where machine learning significantly contributes to the early detection and treatment of skin cancer, ultimately improving patient care and outcomes in dermatology.

7.2 Future Enhancements

The future enhancements of skin cancer detection using machine learning hold tremendous promise for revolutionizing dermatological diagnostics. Anticipated developments in this domain span across various facets, with a focus on improving accuracy, interpretability, and overall clinical applicability. Advanced multi-modal integration, combining various imaging techniques, is poised to provide a more holistic understanding of skin lesions.

Explainable AI (XAI) will take center stage, enhancing transparency in machine learning models and fostering trust among healthcare professionals. Real-time monitoring and feedback mechanisms are expected to become integral, allowing for dynamic adaptations to evolving skin conditions. Automated lesion segmentation techniques are likely to see refinement, contributing to more precise boundary delineation.

Innovations in data augmentation and synthesis will address challenges related to dataset diversity. Personalized risk stratification, considering individual patient characteristics, may become a standard feature, tailoring diagnostic recommendations for each patient. Edge computing technologies will bring skin cancer detection closer to the point of care, facilitating quicker diagnoses in diverse settings. Ethical considerations and regulatory compliance will be paramount, ensuring responsible deployment of these technologies.

Collaborative learning approaches, such as federated models, will promote knowledge sharing without compromising patient privacy. Furthermore, the integration of skin cancer detection into telemedicine platforms will enhance accessibility, allowing for remote assessments and reaching underserved populations. Collectively, these anticipated advancements signify a future where machine learning-driven skin cancer detection stands at the forefront of precision medicine, offering enhanced diagnostic capabilities with a focus on patient-centric care.

Chapter 8

SOURCE CODE & POSTER PRESENTATION

8.1 Source Code

```python

import pandas as pd
import numpy as np
import seaborn as sns
import plotly.express as px
import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
df = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/Minor 1/water_potability.csv")
df.head()
df.isnull().sum()
plt.figure(figsize=(12,8))
sns.heatmap(df.isnull())
plt.figure(figsize= (12,8))
sns.heatmap(df.corr(),annot=True)

df["Potability"].value_counts()
#Visulaization dataset also checking for outliers

fig, ax = plt.subplots(ncols=5, nrows=2, figsize = (20,10))

ax = ax.flatten()

index = 0

for col,values in df.items():
  sns.boxplot(y=col,data=df,ax=ax[index])

  index +=1
sns.pairplot(df)
fig = px.pie(df,names = "Potability",hole = 0.4,template ="plotly_dark")
fig.show()
```

```
fig = px.scatter(df,x = "ph",y="Sulfate",color = "Potability",template= "plotly_dark")
fig.show()
fig = px.scatter(df,x = "Organic_carbon",y="Hardness",color = "Potability",template= "plotly_dark")
fig.show()
df.isnull().mean().plot.bar(figsize = (10,6))
plt.xlabel("Features")
plt.ylabel("Percentage of missing values")
df["ph"] = df["ph"].fillna(df["ph"].mean())
df["Sulfate"] = df["Sulfate"].fillna(df["Sulfate"].mean())
df["Trihalomethanes"] = df["Trihalomethanes"].fillna(df["Trihalomethanes"].mean())
df.isnull().sum()
sns.heatmap(df.isnull())
df.head()
x = df.drop("Potability",axis=1)
y = df["Potability"]
x.shape , y.shape
scaler = StandardScaler()
x = scaler.fit_transform(x)
x
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
x_train.shape , x_test.shape


#Logistic Regression
from sklearn.linear_model import LogisticRegression

#object of LR
model_lr = LogisticRegression()
#Training Model
model_lr.fit(x_train,y_train)
#Making Prediction
pred_lr = model_lr.predict(x_test)
# accuracy score
accuracy_score_lr = accuracy_score(y_test,pred_lr)
accuracy_score_lr*100

from sklearn.tree import DecisionTreeClassifier

# creating the model object
model_dt = DecisionTreeClassifier(max_depth = 4)
#Training of decision tree
model_dt.fit(x_train,y_train)
#Making prediction using Decision Tree
pred_dt = model_dt.predict(x_test)
accuracy_score_dt = accuracy_score(y_test,pred_dt)
accuracy_score_dt*100

#confusion matrix
cm2 = confusion_matrix(y_test,pred_dt)
cm2
```

```python

from sklearn.ensemble import RandomForestClassifier

# Creating model object
model_rf = RandomForestClassifier()
#Training Model RF
model_rf.fit(x_train,y_train)
#Making Prediction
pred_rf = model_rf.predict(x_test)
accuracy_score_rf = accuracy_score(y_test,pred_rf)
accuracy_score_rf*100
cm3 = confusion_matrix(y_test,pred_rf)
cm3

from sklearn.neighbors import KNeighborsClassifier

#Creating Model object
#model_knn = KNeighborsClassifier()
for i in range(4,15):
  model_knn = KNeighborsClassifier(n_neighbors=i)
  model_knn.fit(x_train,y_train)
  pred_knn = model_knn.predict(x_test)
  accuracy_score_knn = accuracy_score(y_test,pred_knn)
  print(i,accuracy_score_knn)
model_knn = KNeighborsClassifier(n_neighbors=11)
model_knn.fit(x_train,y_train)
pred_knn = model_knn.predict(x_test)
accuracy_score_knn = accuracy_score(y_test,pred_knn)
print(accuracy_score_knn*100)

from sklearn.svm import SVC

#Creating object of Model
model_svm = SVC(kernel="rbf")
#Model training
model_svm.fit(x_train,y_train)
#Make prediction
pred_svm = model_svm.predict(x_test)
accuracy_score_svm = accuracy_score(y_test,pred_svm)
accuracy_score_svm*100

from pandas.core.arrays.interval import le
from sklearn.ensemble import AdaBoostClassifier

#Making object of Model
model_ada = AdaBoostClassifier(n_estimators=200,learning_rate=0.03)
#Training the model
model_ada.fit(x_train,y_train)
#Making prediction
pred_ada = model_ada.predict(x_test)
```

```
#accuracy check
accuracy_score_ada = accuracy_score(y_test,pred_ada)
accuracy_score_ada*100

from xgboost import XGBClassifier

#create model
model_xgb = XGBClassifier(n_estimators=100,learning_rate=0.04)
#Traning Model
model_xgb.fit(x_train,y_train)
#Prediction
pred_xgb = model_xgb.predict(x_test)
#accuracy
accuracy_score_xgb = accuracy_score(y_test,pred_xgb)
accuracy_score_xgb*100
models = pd.DataFrame({
    "Model": ["Logistic Regression",
              "Decision Tree",
              "Random Forest",
              "KNN",
              "SVM",
              "AdaBoost",
              "XGBoosT"] ,

    "Accuracy Score" : [accuracy_score_lr,accuracy_score_dt,accuracy_score_rf,
                        accuracy_score_knn,accuracy_score_svm,accuracy_score_ada,accuracy_score_xgb]
})
models
sns.barplot(x="Accuracy Score",y= "Model",data=models)
models.sort_values(by="Accuracy Score",ascending= False)
```

References

[1] A. Zaslavsky and D. Georgakopoulos, “Internet of Things: Challenges and State-of-the-Art Solutions in Internet-Scale Sensor Information Management and Mobile Analytics”, 2020 16th IEEE InternationalConference on Mobile Data Management, pp. 3-6, 2020.

[2] C. J. Baby, H. Singh, A. Srivastava, R. Dhawan and P. Mahalakshmi, “Skin detection: An intelligent cancer alert and prediction system using machine learning approach”, 2021 International Conference on Wireless Communications Signal Processing and Networking (WiSPNET), pp. 771-774, 2021.

[3] David Rutqvist; Fredrik Blomstedt; Denis Kleyko. An Automated Machine Learning Approach for Skin detector Management Systems(2020), IEEE Transactions on Industrial Informatics (Volume: 16, Issue: 1, January 2020).

[4] Guang-Bin Huang, Qin-Yu Zhu and Chee-Kheong Siew, “Extreme learning machine: a new learning scheme of feedforward neural networks”, 2022 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541), vol. 2, pp. 985-990, 2022.

[5] K. L. Keung, C. K. M. Lee, K. K. H. Ng and C. K. Yeung, “Smart City Application and Analysis: Real-time Skin disease problem: A Case Study of Hong Kong”, 2022 IEEE International Conference on Industrial Engineering and Engineering Management, pp. 521-525, 2022.

[6] Pratyaksh P. Rao; Siddhanth P. Rao; Rohit Ranjan, Deep Learning Based Smart Skin Monitoring System, IEEE Third International Conference on Multimedia Processing, Communication Information Technology (MPCIT) (2020).

[7] Qinghua Wang; Viktor Westlund; Jonas Johansson; Magnus Lindgren, Smart Skin-Disease Management and Data Forecast, IEEE Swedish Artificial Intelligence Society Workshop (SAIS) (2021).

[8] Z. Zhang, T. Laakso, Z. Wang et al., “Comparative Study of AI-Based Methods — Application of Analyzing Inflow and Infiltration in Skin problem Subcatchments”, Sustainability, vol. 12, no. 15, 2020.