SRI SIDDHARTHA INSTITUTE OF TECHNOLOGY, MARLUR

TUMAKURU-572105

(A Constitute College of Sri Siddhartha Academy of Higher Education)

Mini-Project Report On:

"Detection of Autism Spectrum Disorder(ASD) using ML Techniques"

submitted in partial fulfillment of the requirement for the completion of VI semester of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE

Submitted by

NEHA ACHARYA (20CS051)

PRATHUASHA K B (20CS058)

Under the guidance of:

DR.RENUKALATHA S

Professor and HOD

Department of CSE

SSIT, Tumakuru

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

2022-23

SRI SIDDHARTHA INSTITUTE OF TECHNOLOGY,

TUMAKURU-572105

DEPARTMENT OF COMPUTER SCIENCE& ENGINEERING

CERTIFICATE

Certified that the mini project work entitled ""Detection of Autism Spectrum Disorder(ASD) using ML Techniques" is a bonafide work being carried out by NEHA ACHARYA (20CS051), PRATHUASHA K B (20CS058) in partial fulfillment for the completion of VI Semester of Bachelor of Engineering in Department of Computer Science & Engineering from Sri Siddhartha Institute of Technology,A Constitute College of Sri Siddhartha Academy of Higher Education during the academic year 2022-23. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report deposited in the department library. The mini Project report has been approved as it satisfies the academic requirements in respect of mini project work prescribed for the Bachelor of Engineering degree.

Signature of the Guide	Signature of the HOD
DR.RENUKALATHA S	**DR.RENUKALATHA S**
Professor & Head, Dept of CSE	Professor & Head, Dept of CSE

Names of the Examiners	**Signature with date**
1.	
2.	

DECLARATION

NEHA ACHARYA(20CS051) and **PRATHUASHA K B(20CS058)** of VI semester, Department of Computer Science and Engineering of Sri Siddhartha Institute of Technology, Tumakuru, hereby declare that this Mini project titled, "Detection of Autism Spectrum Disorder(ASD) using ML Techniques", has been carried out by us under the supervision of **DR.RENUKALATHA S** Professor and HOD Department of CSE, Department of Computer Science and Engineering, Sri Siddhartha Institute of Technology, Tumakuru in partial fulfilment of the requirement for the completion of VI semester in Computer Science and Engineering.

NEHA ACHARYA	(20CS051)
PRATHUASHA K B	(20CS058)

Contents

Abstract

Autism spectrum disorder (ASD) is a neuro-developmental disorder associated with brain development that subsequently affects the physical appearance of the face. This type of mental illness/disorder begins in early childhood and lasts throughout a person's life. Autistic children have different patterns of facial features, which set them distinctively apart from typically developed (TD) children. So specialists believe that these distinct facial features can be used to help diagnose their ASD and even correlate with ASD severity. Autism can be diagnosed at any stage in once life and is said to be a "behavioral disease" because in the first two years of life symptoms usually appear.

Thus this correlation can be used to train a model to detect ASD using those unique facial features. The goal is to provide a free preliminary diagnostic tool that can aid parents in their decision to pursue further ASD testing. Propelled with the rise in use of Machine Learning techniques in the reseaarch dimensions of medical diagnosis. Here is an attempt to explore the possibility to use Naive Bayes, Support Vector Machine, Logistic Regression, KNN, Neural Network and Convolutional Neural Network for predicting of ASD problems in children. It's important for models that have significant real-world impacts to represent their results responsibly and used Bayesian statistics to explain their meaning.

Chapter 1

Introduction

Autism spectrum disorder (ASD) is a type of mental illness that can be detected by using social media data and biomedical images. Autism spectrum disorder (ASD) is a neurological disease correlated with brain growth that later impacts the physical impression of the face. Children with ASD have dissimilar facial landmarks, which set them noticeably apart from typically developed (TD) children.

1.1 Scope and Importance

Recently in the research field, there is a high demand for automating processes to reduce the cost and time of any industry. Health care is one of the most important fields that would benefit from reducing processing time. The speed and efficiency of human health issues diagnostics is significant. The current diagnosing time is a huge challenge in many health conditions, especially Autism.

It takes up to six months to firmly diagnose a child with autism due the long process, and a child must see many different specialists to diagnose autism, starting from developmental pediatricians, neurologists, psychiatrists or psychologists. The time consumed to finalize Autism diagnoses is relatively long in the current traditional way. Therefore, Machine Learning methods can make a significant change to accelerate the process. It is known that Early Intervention is the key for improving Autistic children. Clearly speeding the diagnosing time is even more crucial in Autism cases. Big data and machine learning technologies can make enormous progress to predict and speed up the complex and time-consuming processes of diagnosis and treatment. A machine learning system can be developed to utilize a massive amount of health and medical data available towards predictive modeling and predictive analysis.

This is important because approximately 25 percent of children with autism are undiagnosed. Diagnosing ASD is a complicated and expensive process that not every family can go through. Getting the right kids to the right specialists is crucial to reducing the number of undiagnosed children and the burden on families.

1.2 Problem Statement

The problem of autism spectrum disorder (ASD) have been mounting swiftly nowadays among all ages of the human population. Early detection of this neurological disease can greatly assist in the maintenance of the subject's mental and physical health. With the rise of application of machine learning based models in the predictions of various human diseases, their early detection based on various health and physiological parameter now seems possible.

The main problem of this research is to expedite Autism diagnoses by providing a machine learning system that uses different machine learning algorithms that lead to the make Autism predictive model with most possible accuracy. The solution is proposing a predictive model with high accuracy that can predict if a child has Autism or not using Autism Quotient questionnaire (AQ) test. The aim is to use a traditional Autism diagnosis method and transform it to a machine learning model that can utilize the massive amount of data collected to make predictions, observations and lead to better solutions in the future of discovering Autism at the earliest age possible. Ideally more observations and data analysis in the field will lead to improvements suggesting new methods of improving Autistic population lives.

1.3 Objective

Early detection and treatment are most important steps to be taken to decrease the symptoms of autism spectrum disorder problem and to improve the quality of life of ASD suffering people. It is not possible to completely treat the patient suffering from this disease, however its effects can be reduced for some time if the symptoms are early detected.The goal is to provide a free preliminary diagnostic tool that can aid parents in their decision to pursue further ASD testing. It's important for models that have significant real-world impacts to represent their results responsibly and used Bayesian statistics to explain their meaning.

Chapter 2

Requirements

2.1 System Hardware:

- Computer or Server:Sufficient computing power and memory capacity.
- Processor: Intel-core i5
- 2 GB free disk space

2.2 Software Requirements:

1. **Programming Languages:**

- **Python**

2. **Frameworks and Libraries:**

- **Python Libraries:** Numpy, Pandas, Matplotlib, Seaborn, Sklearn libraries for text processing, tokenization, and feature extraction.
- **Deep Learning Frameworks:** TensorFlow, Keras for implementing deep learning models
- **Computer Vision Libraries:** OpenCV or other computer vision libraries.

Chapter 3

Design and Architecture

3.1 Data flow Diagram

Below Figure 3.1 shows Data-flow Diagram of ASD detection model:

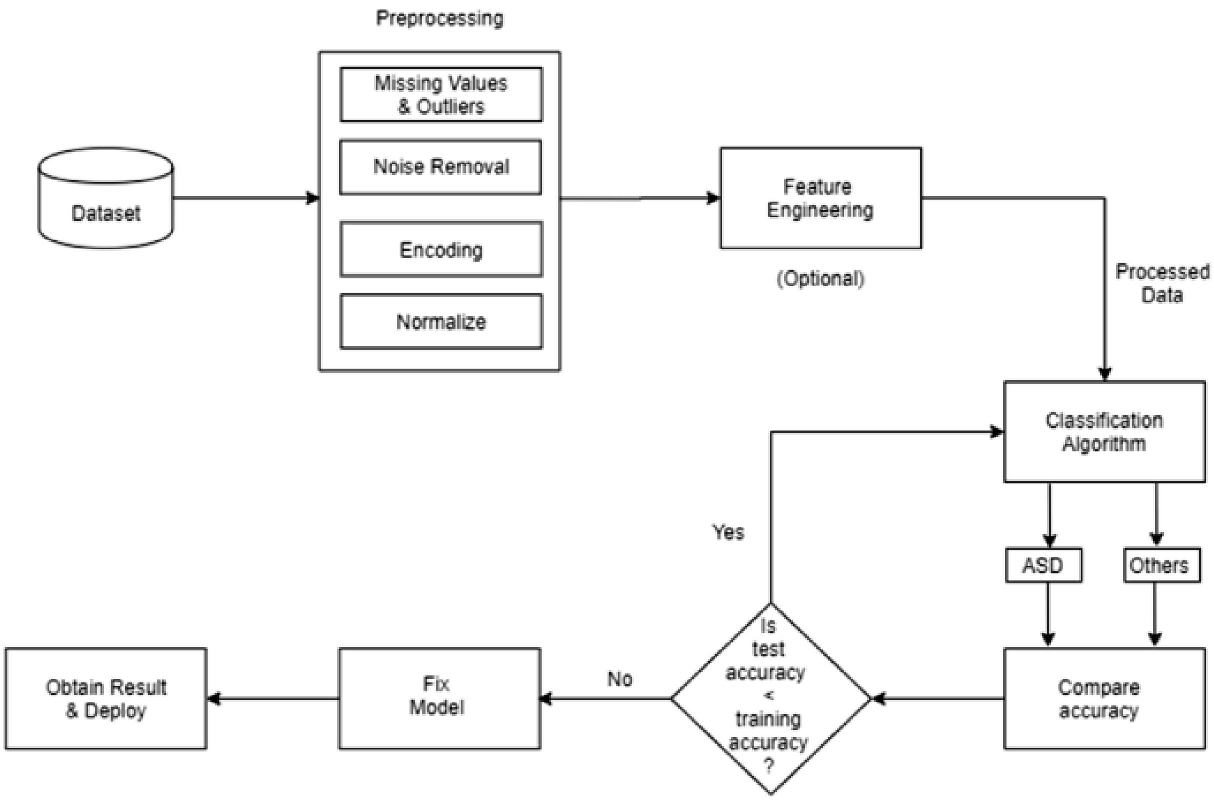


Figure 3.1: Data-flow Diagram of ASD detection model

3.2 System Architecture

Below Figure 3.2 shows System Architecture of ASD detection model:

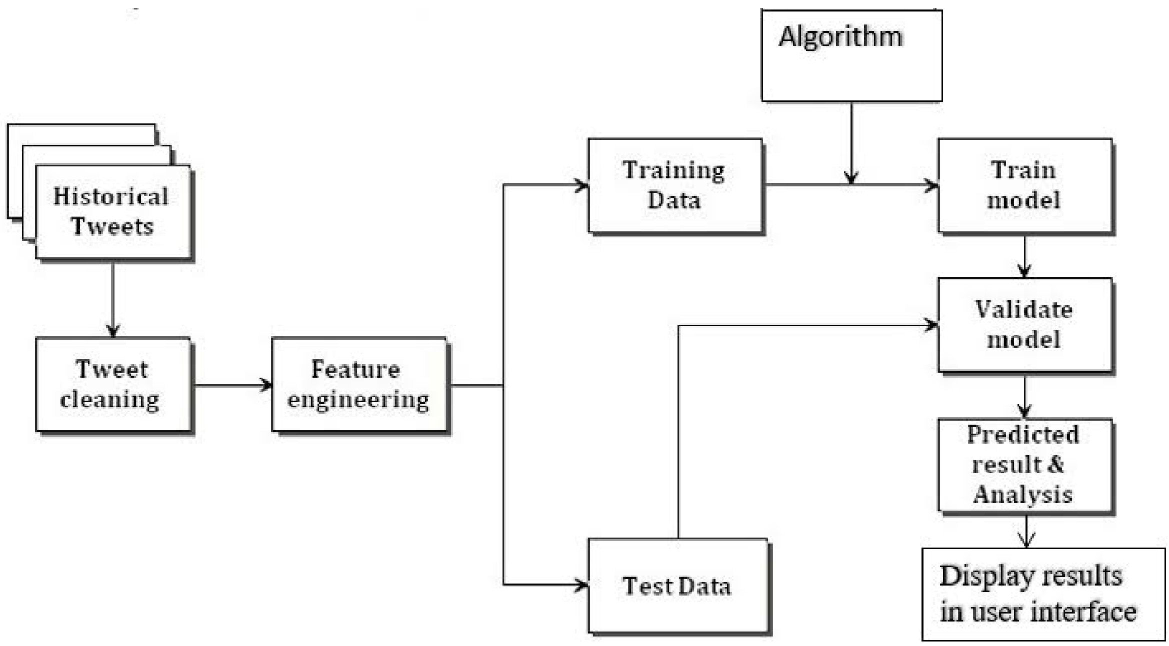


Figure 3.2: System Architecture of ASD detection model

Chapter 4

Implementation

4.1 Detection of ASD using Image Dataset

```
#Importing necessary libraries
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

plt.style.use('classic')

from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, BatchNormalization
from keras.layers import Activation, Dropout, Flatten, Dense
#from keras import backend as K

import os
import cv2
from PIL import Image
import numpy as np

#Mounting colab to drive to load data
from google.colab import drive
drive.mount('/content/drive')

#Assigning Autistic data to Autistic_images
SIZE = 150
dataset = []
label = []
```

```
Autistic_images = os.listdir('/content/drive/MyDrive/ASD Detection/
Autism_data/test/Autistic/')
for image_name in (Autistic_images):
  image = cv2.imread('/content/drive/MyDrive/ASD Detection/
  Autism_data/test/Autistic/'+image_name)
  image = Image.fromarray(image, 'RGB')
  image = image.resize((SIZE,SIZE))
  dataset.append(np.array(image))
  label.append(0)

#Assigning Non-autistic data to Nonautistic_images
Nonautistic_images = os.listdir('/content/drive/MyDrive/ASD Detection/
Autism_data/test/Non_Autistic/')
for image_name in (Autistic_images):
  image = cv2.imread('/content/drive/MyDrive/ASD Detection/Autism_data/
  test/Non_Autistic/'+image_name)
  image = Image.fromarray(image, 'RGB')
  image = image.resize((SIZE,SIZE))
  dataset.append(np.array(image))
  label.append(1)

dataset = np.array(dataset)
label = np.array(label)

#Spliting the dataset
from sklearn.model_selection import train_test_split
import tensorflow as tf
#from keras.utils import to_categorical
x_train, x_test, y_train, y_test = train_test_split(dataset,
label,test_size = 0.20,
random_state = 0)
```

```
#without scaling (normalize) the training may not converge.
#Normalization is a rescaling of the data from the original range
#so that all values are within the range of 0 and 1.
x_train = tf.keras.utils.normalize(x_train,axis=1)
x_test = tf.keras.utils.normalize(x_test,axis=1)

# Define the CNN model
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(SIZE,
    SIZE, 3)),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

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

# Train the model
history = model.fit(x_train, y_train, epochs=10, batch_size=32)

# Extract accuracy values and epochs from the history
accuracy = history.history['accuracy']
epochs = range(1, len(accuracy) + 1)

# Plotting the accuracy vs epochs
```

```
plt.figure(facecolor='w')
plt.plot(epochs, accuracy, marker='o', color='#FF8C00')

# Adding labels and title
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Accuracy vs Epochs')

# Displaying the plot
plt.show()

# Evaluate the model
loss, accuracy = model.evaluate(x_test, y_test)
print("Test loss:", loss)
print("Test accuracy:", accuracy)

#Make predictions
n=24
img = x_test[n]
plt.imshow(img)
input_img = np.expand_dims(img, axis=0)
print("The prediction for this image is: ", model.predict(input_img))
print("The actual label for this image is: ", y_test[n])

#confusion matrix
import numpy as np
from sklearn.metrics import confusion_matrix
import seaborn as sns

# Evaluate the model on test data
y_pred = model.predict(x_test)
y_pred_classes = np.round(y_pred).flatten().astype(int)
```

```
# Compute the confusion matrix
cm = confusion_matrix(y_test, y_pred_classes)

# Plotting the confusion matrix
plt.figure(facecolor='w')
sns.heatmap(cm, annot=True, fmt='d', cmap='Oranges')

# Adding labels and title
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.title('Confusion Matrix')

# Displaying the plot
plt.show()
```

4.2 Detection of ASD using CSV Dataset

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn import metrics
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
from imblearn.over_sampling import RandomOverSampler

import warnings
warnings.filterwarnings('ignore')
```

```
from google.colab import files
uploaded = files.upload()

import io

df = pd.read_csv(io.BytesIO(uploaded['Autism-Child-Data1.csv']))
print(df.head())

df.shape

df.info()

df.describe().T

df['ethnicity'].value_counts()

df['relation'].value_counts()

df = df.replace({'yes':1, 'no':0, '?':'Others', 'others':'Others'})

plt.pie(df['Class/ASD'].value_counts().values, autopct='%1.1f%%')
plt.show()

ints = []
objects = []

for col in df.columns:
  if df[col].dtype == int:
    ints.append(col)
  elif df[col].dtype == object:
    objects.append(col)
```

```
ints.remove('id')
ints.remove('Class/ASD')

plt.subplots(figsize=(15,15))

for i, col in enumerate(ints):
  plt.subplot(5,3,i+1)
  sb.countplot(df, x=df[col], hue=df['Class/ASD'])
plt.tight_layout()
plt.show()

plt.subplots(figsize=(15, 30))

for i, col in enumerate(objects):
plt.subplot(5, 3, i+1)
sb.countplot(df, x=df[col], hue=df['Class/ASD'])
plt.xticks(rotation=60)
plt.tight_layout()
plt.show()

plt.figure(figsize=(15,5))
sb.countplot(data=df, x=df['contry_of_res'], hue=df['Class/ASD'])
plt.xticks(rotation=90)
plt.show()

df = df[df['result']>-5]
df.shape

# This functions make groups by taking
# the age as a parameter
df['age'] = pd.to_numeric(df['age'], errors='coerce')
```

```
def convertAge(age):
if age < 4:
return 'Toddler'
elif age < 12:
return 'Kid'
elif age < 18:
return 'Teenager'
elif age < 40:
return 'Young'
else:
return 'Senior'

df['ageGroup'] = df['age'].apply(convertAge)

sb.countplot(x=df['ageGroup'], hue=df['Class/ASD'])
plt.show()

def add_feature(data):

# Creating a column with all values zero
  data['sum_score'] = 0
  for col in data.loc[:,'A1_Score':'A10_Score'].columns:

  # Updating the 'sum_score' value with scores
  # from A1 to A10
  data['sum_score'] += data[col]

  # Creating a random data using the below three columns
  data['ind'] = data['austim'] + data['used_app_before'] + data['jundice']

  return data
```

```
df = add_feature(df)

sb.countplot(x=df['sum_score'], hue=df['Class/ASD'])
plt.show()

# Applying log transformations to remove the skewness of the data.
df['age'] = df['age'].apply(lambda x: np.log(x))

sb.distplot(df['age'])
plt.show()

def encode_labels(data):
for col in data.columns:

# Here we will check if datatype
# is object then we will encode it
  if data[col].dtype == 'object':
  le = LabelEncoder()
  data[col] = le.fit_transform(data[col])

return data

df = encode_labels(df)

# Making a heatmap to visualize the correlation matrix
plt.figure(figsize=(10,10))
sb.heatmap(df.corr() > 0.8, annot=True, cbar=False)
plt.show()

removal = ['id', 'age_desc', 'used_app_before', 'austim']
features = df.drop(removal + ['Class/ASD'], axis=1)
```

```
target = df['Class/ASD']

X_train, X_val, Y_train, Y_val = train_test_split(features,
target, test_size = 0.2, random_state=10)

# As the data was highly imbalanced we will balance it by adding
repetitive rows
of minority class.
ros = RandomOverSampler(sampling_strategy='minority',random_state=0)
X, Y = ros.fit_resample(X_train,Y_train)
X.shape, Y.shape

# Normalizing the features for stable and fast training.
scaler = StandardScaler()
X = scaler.fit_transform(X)
X_val = scaler.transform(X_val)

from sklearn.impute import SimpleImputer

# Create an imputer with the desired strategy (e.g., mean)
imputer = SimpleImputer(strategy='mean')

# Apply imputation to fill missing values in X
X_imputed = imputer.fit_transform(X)
from sklearn.model_selection import train_test_split

# Split the data into training and validation sets
X_train, X_val, Y_train, Y_val = train_test_split(X_imputed,
Y, test_size=0.2, random_state=42)
models = [LogisticRegression(), XGBClassifier(), SVC(kernel='rbf')]

# Fit the models using the training data
```

```
for model in models:
    model.fit(X_train, Y_train)

    print(f'{model} : ')
    print('Training Accuracy : ', metrics.roc_auc_score(Y_train,
    model.predict(X_train)))
    print('Validation Accuracy : ', metrics.roc_auc_score(Y_val,
    model.predict(X_val)))

from sklearn.metrics import confusion_matrix

# Compute the confusion matrix
confusion = confusion_matrix(Y_val, models[0].predict(X_val))

# Plot the confusion matrix
plt.imshow(confusion, cmap='Blues', interpolation='nearest')
plt.title('Confusion Matrix')
plt.colorbar()
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.xticks(ticks=[0, 1], labels=['Negative', 'Positive'])
plt.yticks(ticks=[0, 1], labels=['Negative', 'Positive'])
plt.show()
```

Chapter 5

Result

5.1 Result

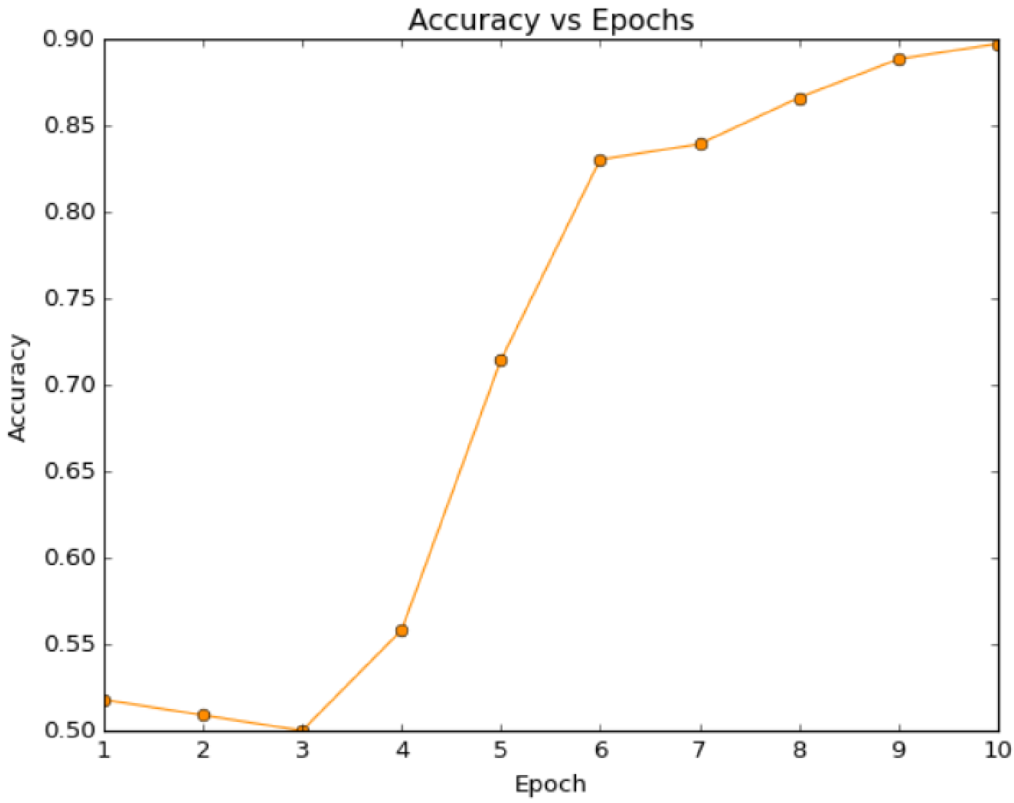
Both the models were trained on the cloud using the Google Colab environment with python language, which supports the most popular libraries for deep learning, e.g., TensorFlow and Keras. For first model the declaration epochs number was 10 with an 32 batch size.The model achieved 89.73% accuracy. For second model achieved 97.7% accuracy on the validation data and 98.1% on the training data.

5.2 Snapshots

Model Training:

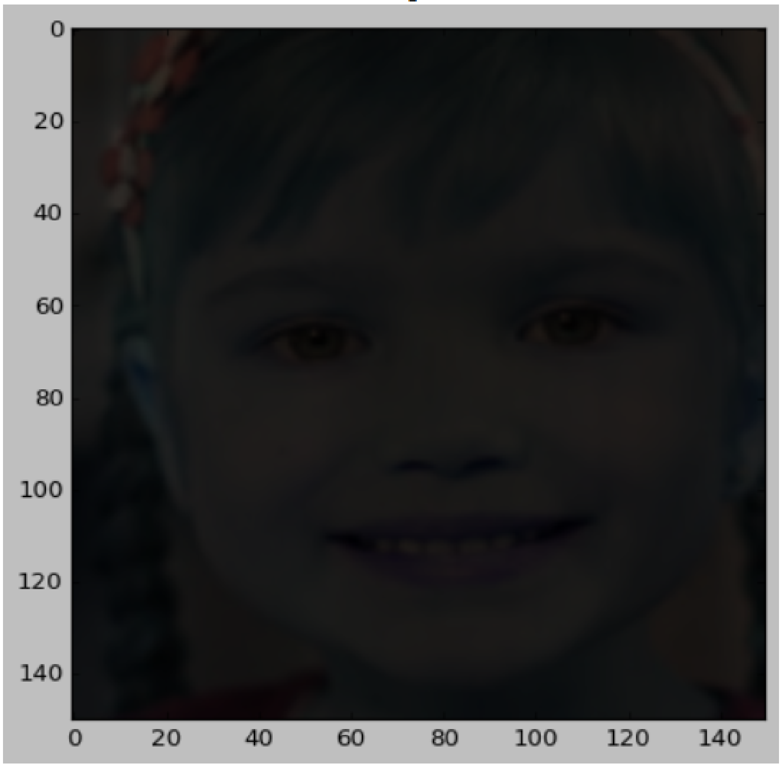
```
Epoch 1/10
7/7 [==============================] - 11s 1s/step - loss: 0.8172 - accuracy: 0.5179
Epoch 2/10
7/7 [==============================] - 8s 1s/step - loss: 0.6919 - accuracy: 0.5089
Epoch 3/10
7/7 [==============================] - 11s 1s/step - loss: 0.6861 - accuracy: 0.5000
Epoch 4/10
7/7 [==============================] - 10s 2s/step - loss: 0.6475 - accuracy: 0.5580
Epoch 5/10
7/7 [==============================] - 7s 1s/step - loss: 0.5765 - accuracy: 0.7143
Epoch 6/10
7/7 [==============================] - 9s 1s/step - loss: 0.4524 - accuracy: 0.8304
Epoch 7/10
7/7 [==============================] - 9s 1s/step - loss: 0.3560 - accuracy: 0.8393
Epoch 8/10
7/7 [==============================] - 11s 2s/step - loss: 0.3273 - accuracy: 0.8661
Epoch 9/10
7/7 [==============================] - 9s 1s/step - loss: 0.3064 - accuracy: 0.8884
Epoch 10/10
7/7 [==============================] - 8s 1s/step - loss: 0.2708 - accuracy: 0.8973
```

Plot of Accuracy vs Epochs:



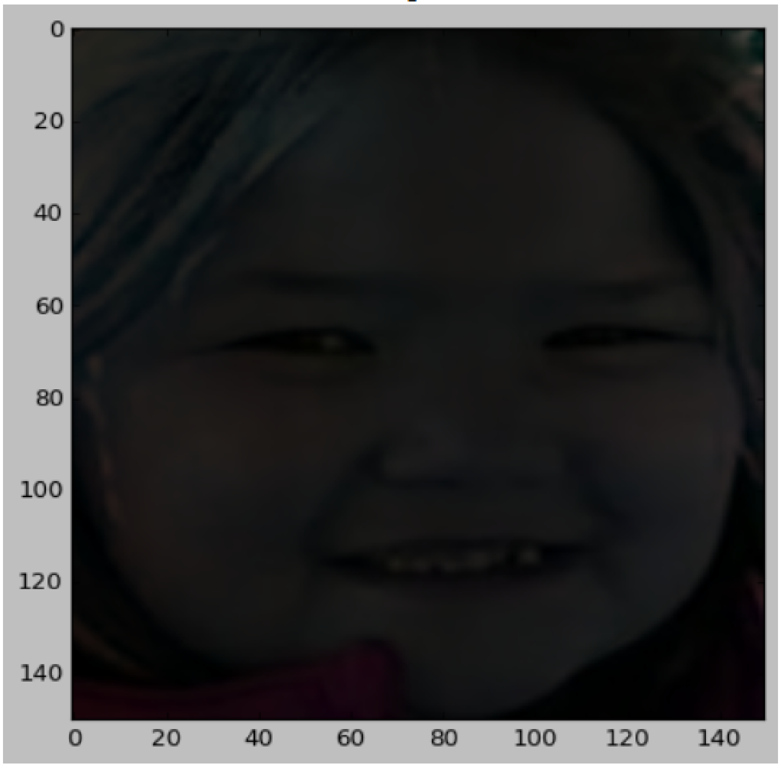
Prediction, n=24:

```
1/1 [==============================] - 0s 30ms/step
The prediction for this image is:  [[0.7510751]]
The actual label for this image is:  1
```

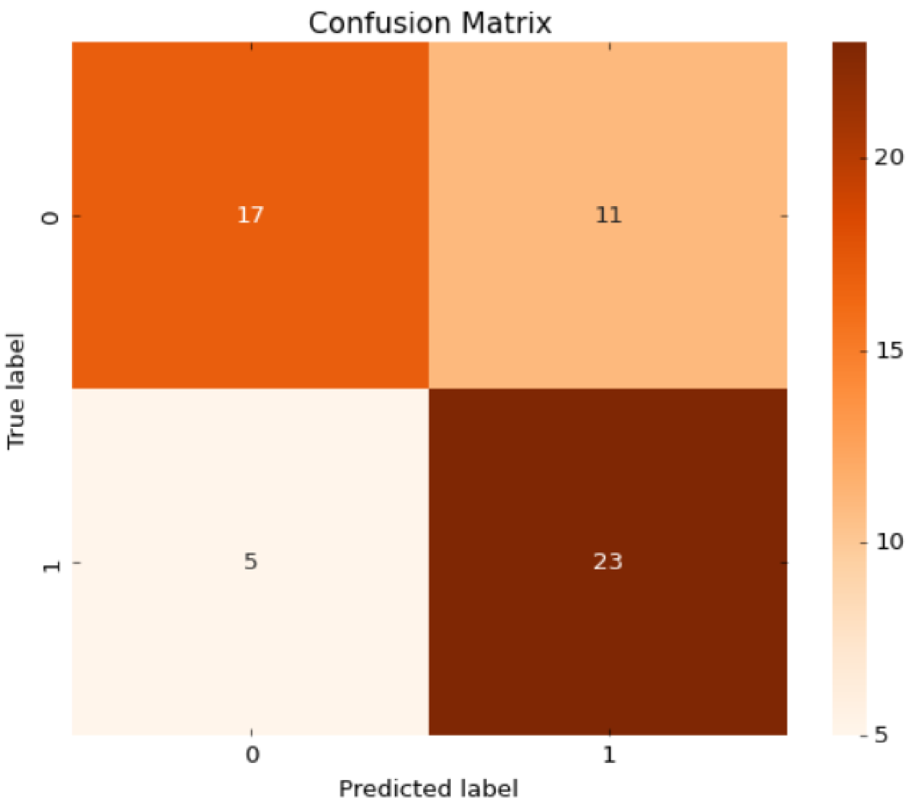


Prediction: n=3

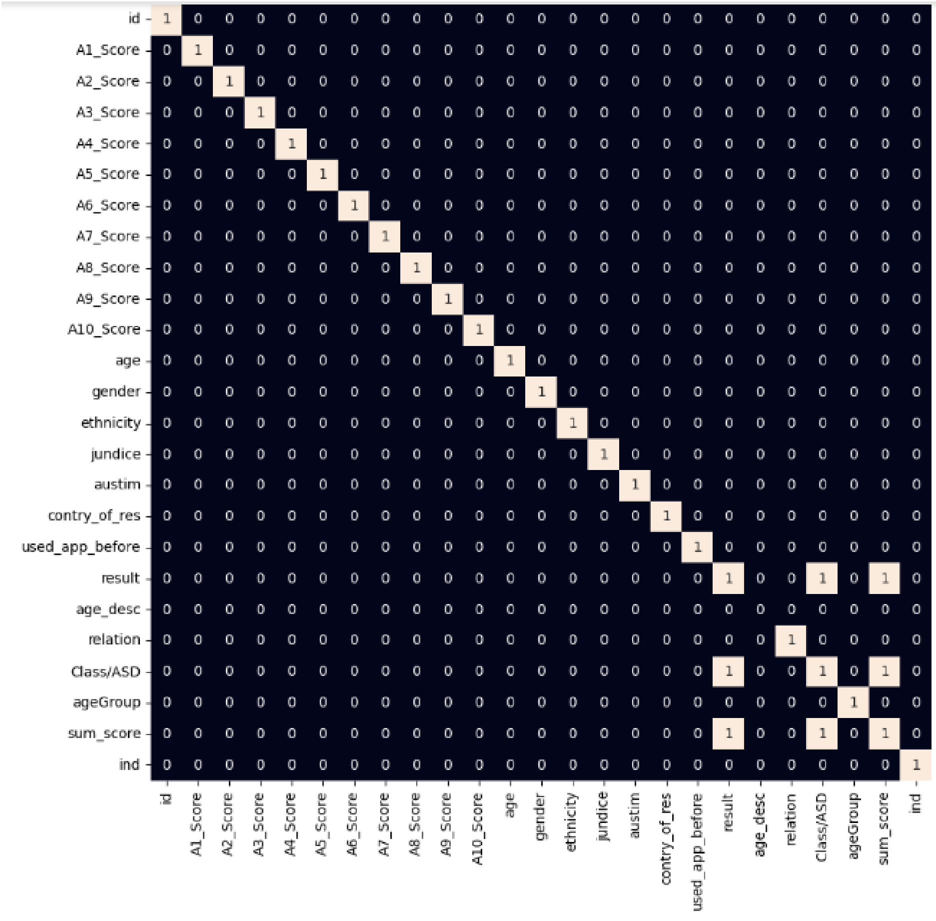
```
1/1 [==============================] - 0s 60ms/step
The prediction for this image is:  [[0.8637257]]
The actual label for this image is:  0
```



Confusion matrix:



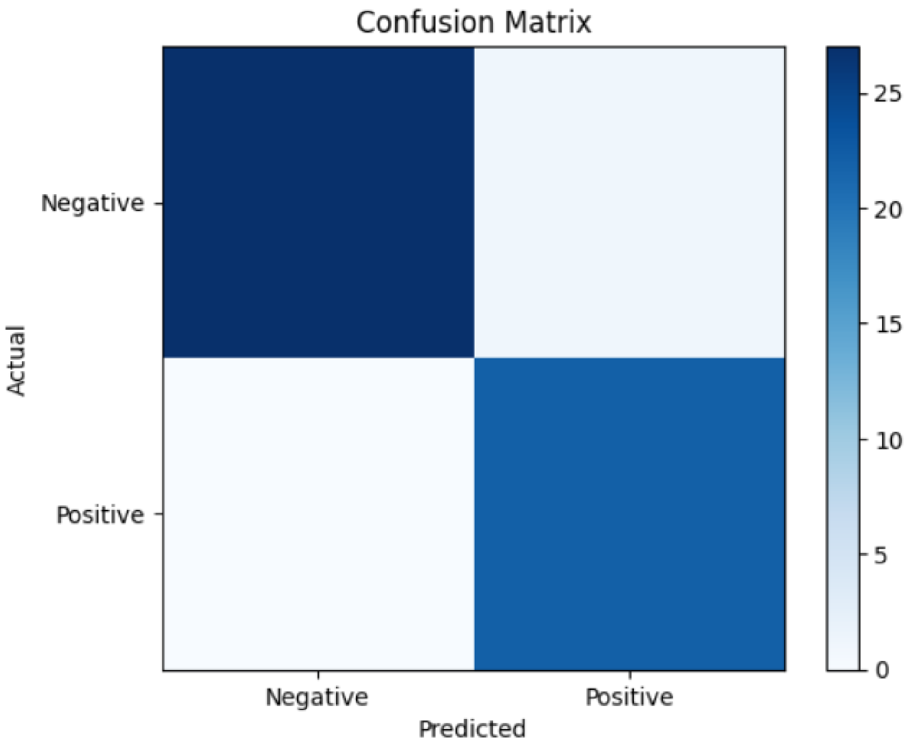
Confusion matrix:



Confusion matrix:

```
LogisticRegression() :
Training Accuracy :  0.995049504950495
Validation Accuracy :  0.9821428571428572
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              n_estimators=100, n_jobs=None, num_parallel_tree=None,
              predictor=None, random_state=None, ...) :
Training Accuracy :  0.995049504950495
Validation Accuracy :  1.0
SVC() :
Training Accuracy :  0.9897863470557582
Validation Accuracy :  0.9772727272727273
```

Confusion matrix:



Chapter 6

Conclusion

Interest in child autism has risen due to the advances in global health know-how and-capacities. Moreover, the number of autistic children has increased in recent years, due to which researchers and academics have intensified their efforts to uncover the causes of autism and to detect it early in order to give autistic people behavioral development treatment programs that should help them integrate into society and leave the isolation of the autistic world.

In this work, detection of Autism Spectrum Disorder was attempted using various machine learning and deep learning techniques. Various performance evaluation metrics were used to analyze the performance of the models implemented for ASD detection on non-clinical dataset.Although accuracy was used to measure model performance during training, sensitivity and specificity are more important to consider for Bayesian predictions.

It is important to remember that models are fallible. A positive result from a model does not mean a positive result in the real world. Instead, this new information, like a predicted positive, should be used to update our prior knowledge about that event.

Future Work In this study, the output of this model was autism/nonautism. We obtained an accuracy of 89% and 97% from both the models respectively. However while studying about ASD Detection we foound Detection of ASD using Brain imaging is more accurate compare to using Facial features of images and CSV data. So we are determined to train a model using dataset related to brain-imaging for detection of ASD and to develop a deep learning-based web application for autism detection using HTML, CSS, and JS as the front-end and Flask with the trained model's architecture as the back-end for an easy interactive user interface that enables specialists to test the child's behavioral state through features.

Bibliography

[1] Suman Raj, Sarfaraz Masood, "Analysis and Detection of Autism Spectrum Disorder Using Machine Learning Techniques", International Conference on Computational Intelligence and Data Science, 2019.

[2] Mikian Musser, "Detecting Austism Spectrum Disorder in Children With Computer Vision", Towards DataScience, Apr. 2020.

[3] Fawaz Waselallah Alsaade, Mohammed Saeed Alzahrani, "Classification and Detection of Autism Spectrum Disorder Based on Deep Learning Algorithms", Computational Intelligence and Neuroscience, vol. 2022, Article ID 8709145, 10 pages, 202- https://doi.org/10.1155/2022/8709145

[4] Ahmed ZAT, Aldhyani THH, Jadhav ME, Alzahrani MY, Alzahrani ME, Althobaiti MM, Alassery F, Alshaflut A, Alzahrani NM, Al-Madani AM. Facial Features Detection System To Identify Children With Autism Spectrum Disorder: Deep Learning Models. Comput Math Methods Med. 2022 Apr 4;2022:3941049. doi: 10.1155/2022/3941049. PMID: 35419082; PMCID: PMC9001065.