A picture containing sketch, drawing, text, font

Description automatically generated A picture containing text, font, logo, poster

Description automatically generated

Machine learning Project Report

Ahmed Ali 212065

Nour Ezz 212871

Abdalla Ahmed 212853

Bassel El sayed 214023

Supervised by: Dr.Ammar

# Introduction

# 1.1 Description of Dataset

# Diabetes classification involves considering various factors to determine the type and severity of the condition. These factors include Diabetes\_binary (presence or absence of diabetes), HighBP (high blood pressure), HighChol (high cholesterol levels), CholCheck (cholesterol screening), BMI (body mass index), Smoker (smoking status), Stroke, HeartDiseaseorAttack, PhysActivity (physical activity level), Fruits and Veggies consumption, HvyAlcoholConsump (heavy alcohol consumption), AnyHealthcare (access to healthcare services), NoDocbcCost (healthcare cost coverage), GenHlth (general health status), MentHlth (mental health status), PhysHlth (physical health status), DiffWalk (difficulty in walking), Sex, Age, Education, and Income. These indicators help in classifying and understanding the diverse factors contributing to diabetes, enabling personalized approaches to treatment and prevention strategies.

# 1.2 Description of Selected Classifiers

# Classifiers are machine learning algorithms used to classify or categorize data into different classes or categories. Selected classifiers include MLP (Multilayer Perceptron), Random Forest, SVM (Support Vector Machines), KNN (K-Nearest Neighbors), and Naive Bayes. MLP is a neural network with multiple layers, Random Forest combines decision trees, SVM separates data using hyperplanes, KNN classifies based on neighbors, and Naive Bayes uses Bayes' theorem. Each has unique strengths and is chosen based on the data and classification requirements.

# 1.3 Implementation of Selected Classifiers

# MLP: Implementing an MLP (Multilayer Perceptron) classifier involves defining the architecture of the neural network and training it using libraries like TensorFlow or Keras.

# Random Forest: Implementing a Random Forest classifier involves using libraries like scikit-learn in Python to create an ensemble of decision trees and train the model on the dataset.

# SVM: Implementing an SVM (Support Vector Machines) classifier involves using libraries like scikit-learn in Python to define the SVM model, choose appropriate kernels, and train it on labeled data for classification tasks.

# KNN: Implementing a KNN (K-Nearest Neighbors) classifier involves using libraries like scikit-learn in Python to define the number of neighbors, choose a distance metric, and train the model on labeled data for classification tasks.

# Naïve Bayes: Implementing a Naive Bayes classifier involves using libraries like scikit-learn in Python to define the Naive Bayes model, handle feature independence assumptions, and train it on labeled data for classification tasks.

# 1.4 Implementation of evaluation metrics

# Accuracy: Accuracy is an evaluation metric that measures the percentage of correctly classified instances by a model, calculated by dividing the number of correct predictions by the total number of instances.

# Recall: Recall is an evaluation metric that measures the ability of a model to correctly identify all relevant instances in a dataset. It is calculated by dividing the number of true positive predictions by the sum of true positives and false negatives, providing insights into the model's sensitivity to detecting positive instances.

# Precision: Precision is an evaluation metric that measures the proportion of correctly predicted positive instances out of all instances predicted as positive by a model. It is calculated by dividing the number of true positive predictions by the sum of true positives and false positives, indicating the model's accuracy in identifying true positives while minimizing false positives.

# F1-Score: The F1 score is an evaluation metric that combines precision and recall to provide a balanced measure of a model's performance. It is calculated by taking the harmonic mean of precision and recall, offering a single value that considers both the model's ability to identify positive instances and its ability to avoid false positives.

# Confusion Matrix: A confusion matrix is a tabular representation of a classification model's performance, providing a summary of the predicted and actual labels. It shows the counts of true positives, true negatives, false positives, and false negatives, allowing for a comprehensive assessment of the model's accuracy, precision, recall, and other performance metrics.

# Performance

# Naïve Bayes:

# Accuracy: 0.7735335856196783

# Recall: 0.7735335856196783

# Precision: 0.83427646328236

# F1-Score: 0.7967611741017034

# Confusion Matrix: [35412, 8306] [ 3184, 3834]

# KNN:

# Accuracy: 0.8512298959318827

# Recall: 0.8512298959318827

# Precision: 0.8145805253150754

# F1-Score: 0.8230378797757573

# SVM:

# Accuracy: 0.8624251024913276

# Recall: 0.8624251024913276

# Precision: 0.8216779723832509

# F1-Score: 0.8016643127907924

# Confusion Matrix: [65501, 104] [10366, 133]

# Random Forest:

# Accuracy: 0.945

# Recall: 0.945

# Precision: 0.9451118534264602

# F1-Score: 0.9450320488459881

# MLP:

# 

# Discussion of results

# Random Forest Scores the highest in all evaluation metrics as it is the closest to getting a 1.0 result scoring an average 0.945. However, Naïve bayes scored the lowest in all evaluation metrics as it is the closest result to 0 scoring an average 0.77. The rest of evaluation metrics (MLP, KNN, SVM) scored between

# 0.80 – 0.86

# Challenges faced

# The most challenging part is cleaning and sorting of the data. As in the dataset there is some null values, values containing 0, missing data, determine whether they are valid or erroneous data points, inconsistent or incorrect data formats, duplicate records, and conflicting values can further complicate the cleaning process.

# Reference

# Google Colab, all classifiers on MSA e-learning Machine learning labs, for Random Forest classifier: <https://builtin.com/data-science/random-forest-python-deep-dive>

# 