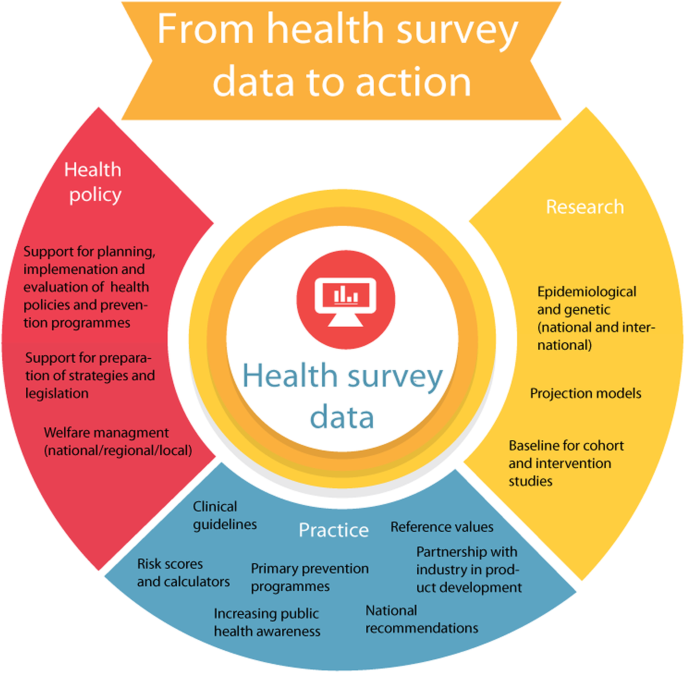
Tittle:Public Health Awareness Campaign Analysis

Phase 2

**ABSTRACT:**

Public health awareness campaigns play a vital role in promoting healthy behaviors and preventing diseases. To enhance the impact and efficiency of these campaigns, this research proposes the incorporation of machine learning algorithms to predict the success of future public health awareness campaigns based on historical data. The objective is to leverage data-driven insights to optimize campaign strategies, allocate resources effectively, and maximize the reach and influence of these critical initiatives

1. **Data Collection and Preprocessing:**

* Gather historical data on previous public health awareness campaigns. This data should include information about the campaign's objectives, target audience, messaging, channels used, campaign duration, and outcomes (e.g., increased awareness, changed behavior, or other relevant metrics).****
* Clean and preprocess the data. This may involve handling missing values, encoding categorical variables, and normalizing or scaling numerical features.

1. **Feature Engineering:**

* Identify relevant features or variables that may influence campaign success. This can include demographic data, campaign content, timing, and geographical information 
* Create new features or transform existing ones if necessary. For example, you might calculate the campaign's reach as a percentage of the target population or incorporate sentiment analysis of social media comments

1. **Data Spliting:**

* Split your data into training, validation, and test sets. This ensures you can train, tune, and evaluate your machine learning models properly.

***Syntax:***

from sklearn.model\_selection

import train\_test\_split

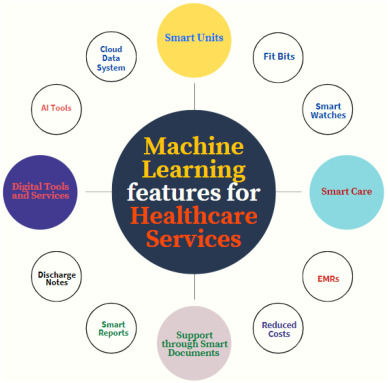
# Split the data into training (80%) and the rest (20%)

X\_train, X\_temp, y\_train, y\_temp = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Split the remaining data into validation (50%) and test (50%)

X\_val, X\_test, y\_val, y\_test = train\_test\_split(X\_temp, y\_temp, test\_size=0.5, random\_state=42)

**4)Selecting Machine Learning Algorithms:**

* Choose appropriate machine learning algorithms for the task. Regression, classification, or even time series forecasting models might be suitable, depending on how you frame the problem. ****
* Consider using ensemble methods or deep learning models for complex relationships within the data.

**5) Model Training and Tuning:**

* Train your chosen machine learning models on the training dataset. Tune hyper parameters to optimize their performance on the validation set. Common algorithms include decision trees, random forests, gradient boosting, neural networks, and more. ****
* Evaluate model performance using appropriate metrics such as accuracy, F1-score, or ROC-AUC, depending on the nature of the problem (classification or regression).

***Synatx:***

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split, GridSearchCV

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

**# Load and prepare the data**

# Load your dataset (replace 'data.csv' with your data file)

data = pd.read\_csv('data.csv')

# Split data into features and target

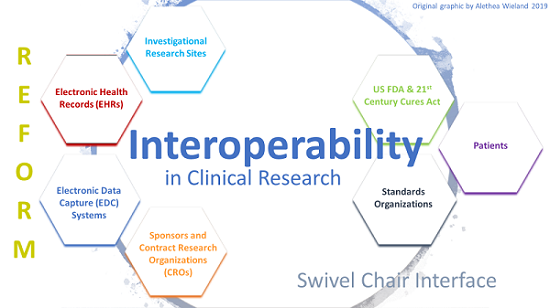
X = data.drop('target\_column', axis=1)

y = data['target\_column']

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**6) Interoperability and Explainable:**

* Ensure that your machine learning models are interpretable and explainable, especially in the context of public health campaigns. This will help stakeholders understand the factors contributing to campaign success. ****

***Syntax:***

# Define hyperparameter grid for tuning

param\_grid = {

'n\_estimators': [50, 100, 200],

'max\_depth': [None, 10, 20],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 2, 4]

}

# Create a grid search object

grid\_search = GridSearchCV(estimator=model, param\_grid=param\_grid, cv=5, n\_jobs=-1, verbose=2)

# Perform the grid search

grid\_search.fit(X\_train, y\_train)

# Get the best parameters

best\_params = grid\_search.best\_params\_

**7) Deployment and Monitoring:**

* Deploy the trained model to predict the success of future campaigns based on their input features. ****
* Continuously monitor and update the model as new campaign data becomes available to maintain its accuracy and relevance.

***Syntax:***

pip install Flask

from flask import Flask, request, jsonify

import joblib # Used to load the trained model

app = Flask(\_\_name\_\_)

# Load the trained model

model = joblib.load('your\_model.pkl')

@app.route('/predict', methods=['POST'])

def predict():

try:

# Get data from the request

data = request.get\_json()

# Make predictions using the model

predictions = model.predict(data['data'])

# Return the predictions as JSON

return jsonify({'predictions': predictions.tolist()})

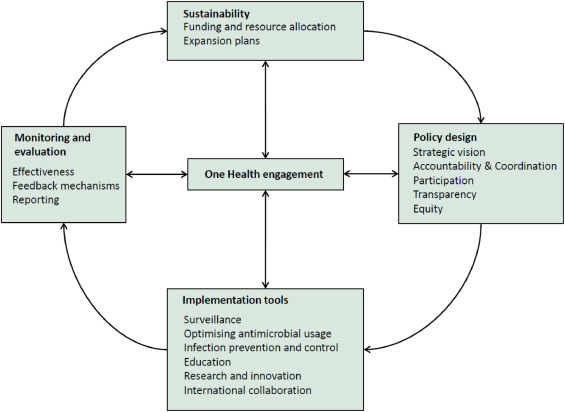
except Exception as e:

return jsonify({'error': str(e)})

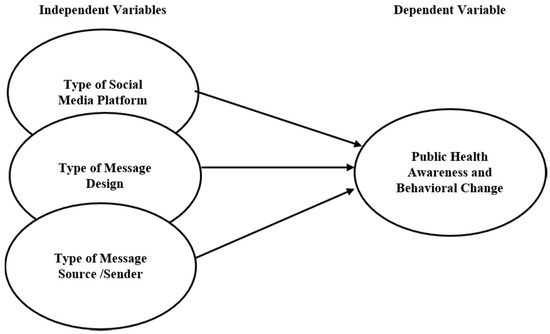
if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='0.0.0.0', port=5000)

**8) Feedback loop:**

* Establish a feedback loop where the model's predictions and insights are used to refine future campaign strategies. This iterative process can help improve campaign effectiveness over time. ****
* Public health organizations collect data on various aspects of health, such as disease prevalence, risk factors, and public knowledge and attitudes.
* The collected data is analyzed to identify trends, patterns, and areas where public health awareness may need improvement**.**

**9) Ethical Considerations:**

* Be mindful of ethical considerations when using machine learning in public health. Ensure fairness, avoid bias, and protect sensitive information. ****

***Syntax:***

from aif360.datasets import StandardDataset

from aif360.metrics import BinaryLabelDatasetMetric

from aif360.metrics import ClassificationMetric

# Convert the dataset to AIF360 format

privileged\_groups = [{'race': 1}]

unprivileged\_groups = [{'race': 0}]

dataset = StandardDataset(df, label\_name='target\_column', favorable\_classes=[1], protected\_attribute\_names=['race'], privileged\_classes=[[1]])

# Calculate metrics for fairness assessment

metric = BinaryLabelDatasetMetric(dataset, unprivileged\_groups=unprivileged\_groups, privileged\_groups=privileged\_groups)

print("Disparate Impact:", metric.disparate\_impact())

print("Statistical Parity Difference:", metric.statistical\_parity\_difference())

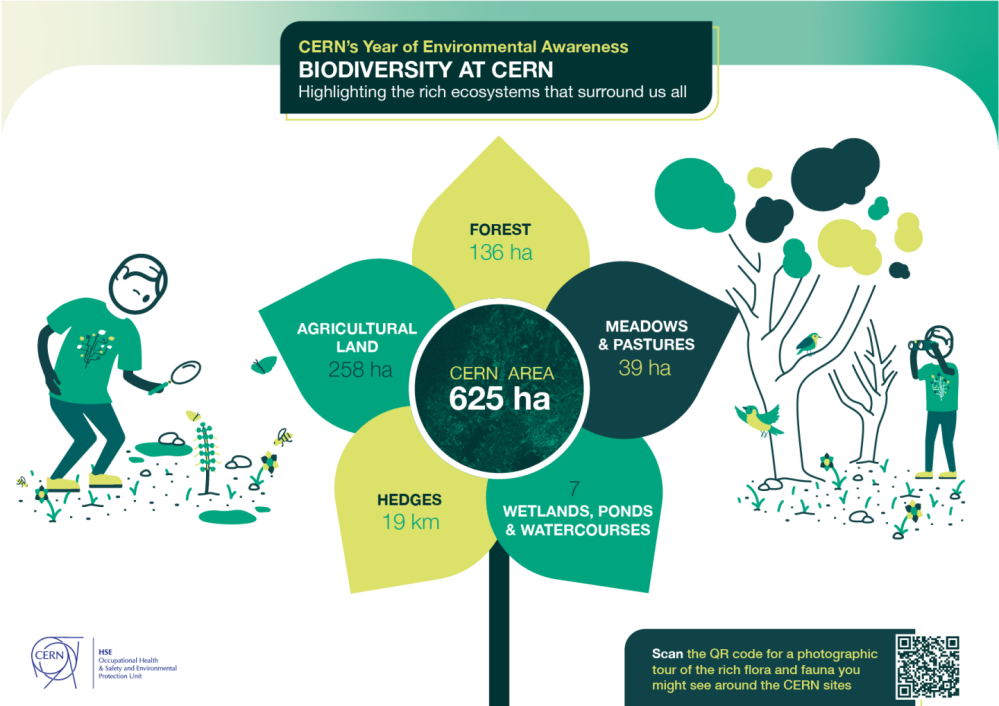
# Calculate metrics for classification fairness

classified\_metric = ClassificationMetric(dataset, y\_pred, unprivileged\_groups=unprivileged\_groups, privileged\_groups=privileged\_groups)

print("Balanced Accuracy:", classified\_metric.balance\_accuracy())

print("Statistical Parity Difference (classification):", classified\_metric.statistical\_parity\_difference())

**10) Document and Reporting:**

* Document your entire machine learning pipeline, including data sources, pre-processing steps, model selection, and evaluation metrics. This documentation is crucial for transparency and productivity. ****