# Problem statement

The project involves analyzing data from public health awareness campaigns to measure their effectiveness in reaching the target audience and increasing awareness. The objective

is to provide insights that evaluate the impact of the campaigns and inform future strategies.

This project includes defining analysis objectives, collecting campaign data, designing relevant visualizations in IBM Cognos, and using code for data analysis

# Phase 1:Abstract

Public health awareness is essential for promoting the well-being of communities. It involves

educating the public about various health issues, preventive measures, and resources

available for healthcare.

# Design thinking

1. Analysis Objectives: Define specific objectives for analyzing public health awareness campaign data, such as measuring audience reach, awareness levels, and campaign impact.
2. Data Collection: Identify the sources and methods for collecting campaign data,

including engagement metrics, audience demographics, and awareness surveys.

1. Visualization Strategy: Plan how to visualize the insights using IBM Cognos to create

informative dashboards and reports.

1. Code Integration: Decide which aspects of the analysis can be enhanced using code, such as data cleaning, transformation, and statistical analysis.

# Phase 2:Innovation

Consider incorporating machine learning algorithms to predict the success of future

campaigns based on historical data

# Phase 3:Development Part 1

Start building the public health awareness campaign analysis using IBM Cognos for

visualization

# Phase 4:

Continue building the analysis by creating visualizations using IBM Cognos and integrating

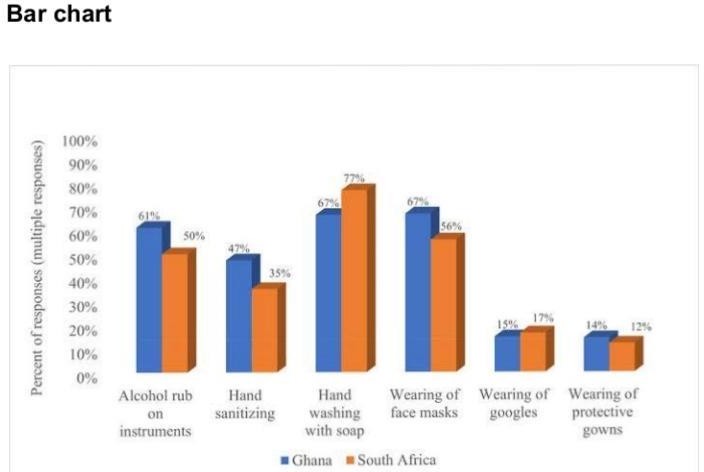
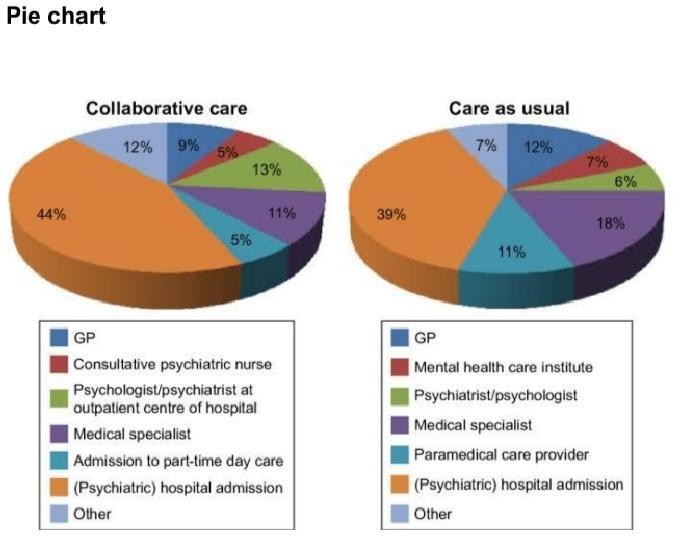
code for data analysis

# Phase 5:Project Documentation and submittion

Document the public health awareness campaign analysis project and prepare it for

submission

# Visualization section



Machine learning algorithm

Machine Learning (ML) is a subtype of Artificial Intelligence (AI) technology that aims to improve the speed and accuracy of physicians’ work.

This paper identifies the need for ML in healthcare. Paper identifies and discusses the significant applications of ML for Healthcare. Paper explores how ML-based tools are used to provide various treatment alternatives and individualised treatments and improve the overall efficiency of healthcare systems. Paper finds that ML will be crucial in developing clinical decision support, illness detection, and personalised treatment approaches to provide the best potential outcomes.

# Roles of machine learning

* Countries are currently dealing with an overburdened healthcare system with a shortage of skilled physicians, where AI provides a big hope.
* The healthcare data can be used gainfully to identify the optimal trial sample, collect more data points, assess ongoing data from trial participants, and eliminate data-based errors.
* ML-based techniques assist in detecting early indicators of an epidemic or pandemic.



# Ways for identification

* It identified and discussed the significant applications of ML for

healthcare.

* The applications of this technology in healthcare operations can be

tremendously advantageous to the organisation.

* ML-based tools are used to provide various treatment alternatives and individualised treatments and improve the overall efficiency of hospitals and healthcare systems while lowering the cost of care.
* Shortly, ML will impact both physicians and hospitals. It will be crucial in developing clinical decision support, illness detection, and personalised treatment approaches to provide the best potential outcomes.

# Importing library

Import pandas as pd import numpy as np import seaborn as sns

import matplotlib.pyplot as plt

print(‘Successfully imported’)

# Importing dataset

Data = pd.read\_csv(‘/kaggle/input/mental-health-in-tech- survey/survey.csv’)

data.head()

# Preprocessing and cleaning dataset

Check the data set for missing data

If data.isnull().sum().sum() == 0 :

print (‘There is no missing data in our dataset’)

else:

print(‘There is {} missing data in our dataset ‘.format(data.isnull().sum().sum()))

# Check our missing data from which volume and how many unique features they have

Frame = pd.concat([data.isnull().sum(), data.nunique(), data.dtypes], axis = 1, sort= False)

frame

## Look at what is in the ‘Work\_interfere’ column to

choose a suitable method to fill nan values.

Data[‘work\_interfere’].unique()

Plot \*\*work\_interfere\*\*

Add the value of each parametr on the Plot

ax = sns.countplot(data = data , x = ‘work\_interfere’);

ax.bar\_label(ax.containers[0]);

From sklearn.impute import SimpleImputer

import numpy as np

columns\_to\_drop = [‘state’, ‘comments’, ‘Timestamp’]

for column in columns\_to\_drop: if column in data.columns:

data = data.drop(columns=[column])

**Fill in missing values in work**\_**interfere column**

data[‘work\_interfere’] = np.ravel(SimpleImputer(strategy =

‘most\_frequent’).fit\_transform(data[‘work\_interfere’].values.reshape(-1,1)))

data[‘self\_employed’] = np.ravel(SimpleImputer(strategy =

‘most\_frequent’).fit\_transform(data[‘self\_employed’].values.reshape(-1,1)))

data.head()

# Bar chart representation

Ax = sns.countplot(data=data, x=‘work\_interfere’);

ax.bar\_label(ax.containers[0]);

### Check unique data in gender columns

print(data[‘Gender’].unique()) print(‘’)

print(‘-’\*75)

print(‘’)

### Check number of unique data too.

Print(‘number of unique Gender in our dataset is :’, data[‘Gender’].nunique())

## Gender data contains dictation problems, nonsense answers, and too unique Genders.

data[‘Gender’].replace([‘Male ‘, ‘male’, ‘M’, ‘m’, ‘Male’, ‘Cis Male’,

‘Man’, ‘cis male’, ‘Mail’, ‘Male-ish’, ‘Male (CIS)’,

‘Cis Man’, ‘msle’, ‘Malr’, ‘Mal’, ‘maile’, ‘Make’,], ‘Male’, inplace = True)

data[‘Gender’].replace([‘Female ‘, ‘female’, ‘F’, ‘f’, ‘Woman’, ‘Female’, ‘femail’, ‘Cis Female’, ‘cis-female/femme’, ‘Femake’, ‘Female (cis)’, ‘woman’,], ‘Female’, inplace = True)

data[“Gender”].replace([‘Female (trans)’, ‘queer/she/they’, ‘non-binary’,

‘fluid’, ‘queer’, ‘Androgyne’, ‘Trans-female’, ‘male leaning androgynous’, ‘Agender’, ‘A little about you’, ‘Nah’, ‘All’,

‘ostensibly male, unsure what that really means’,

‘Genderqueer’, ‘Enby’, ‘p’, ‘Neuter’, ‘something kinda male?’, ‘Guy (-ish) ^\_^’, ‘Trans woman’,], ‘Other’, inplace = True)

print(data[‘Gender’].unique())

# Plot Genders column after cleaning and new categorizing

ax = sns.countplot(data=data, x=‘Gender’);

ax.bar\_label(ax.containers[0]);

### Our data is clean now ? Let’s see.

If data.isnull().sum().sum() == 0:

print(‘There is no missing data’)

else:

print(‘There is {} missing

data’.format(data.isnull().sum().sum()))

**Lt’s check duplicated data.**

If data.duplicated().sum() == 0:

print(‘There is no duplicated data:’)

else:

print(‘Tehre is {} duplicated data:’.format(data.duplicated().sum())) #If there is duplicated data drop it.

Data.drop\_duplicates(inplace=True)

print(‘-’\*50)

print(data.duplicated().sum())

### Look unique data in Age column

data[‘Age’].unique()

### We had a lot of nonsense answers in the Age column too.

**This filtering will drop entries exceeding 100 years and those indicating negative values**.

Data.drop(data[data[‘Age’]<0].index, inplace = True) data.drop(data[data[‘Age’]>99].index, inplace = True)

print(data[‘Age’].unique())

### Let’s see the Age distribution in this dataset.

Plt.figure(figsize = (10,6))

age\_range\_plot = sns.countplot(data = data, x = ‘Age’); age\_range\_plot.bar\_label(age\_range\_plot.containers[0]); plt.xticks(rotation=90);

### In this plot moreover on Age distribution we can see treatment distribution by age

plt.figure(figsize=(10, 6)); sns.displot(data[‘Age’], kde = ‘treatment’); plt.title(‘Distribution treatment by age’);

* #Check Dtypes data.info()

# Use LabelEncoder to change the Dtypes to ‘int

’

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

#Make the dataset include all the columns we need to change their dtypes

columns\_to\_encode = [‘Gender’, ‘Country’, ‘self\_employed’,’family\_history’, ‘treatment’, ‘work\_interfere’,’no\_employees’,

‘remote\_work’, ‘tech\_company’,’benefits’,’care\_options’, ‘wellness\_program’,

‘seek\_help’, ‘anonymity’, ‘leave’, ‘mental\_health\_consequence’, ‘phys\_health\_consequence’, ‘coworkers’, ‘supervisor’, ‘mental\_health\_interview’,’phys\_health\_interview’, ‘mental\_vs\_physical’, ‘obs\_consequence’]

#Write a Loop for fitting LabelEncoder on columns\_to\_encode

for columns in columns\_to\_encode: data[columns] = le.fit\_transform(data[columns])

data.info()

* + #Let’s check Standard deviation
  + data.describe()

From sklearn.preprocessing import MaxAbsScaler, StandardScaler

data[‘Age’] = MaxAbsScaler().fit\_transform(data[[‘Age’]]) data[‘Country’] = StandardScaler().fit\_transform(data[[‘Country’]]) data[‘work\_interfere’] =

StandardScaler().fit\_transform(data[[‘work\_interfere’]])

data[‘no\_employees’] =

StandardScaler().fit\_transform(data[[‘no\_employees’]])

data[‘leave’] = StandardScaler().fit\_transform(data[[‘leave’]])

data.describe()

# Split the data to train and test

From sklearn.model\_selection import train\_test\_split #I wanna work on ‘treatment’ column.

X = data.drop(columns = [‘treatment’]) y = data[‘treatment’]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25)

print(X\_train.shape, y\_train.shape)

print(‘-’\*30)

print(X\_test.shape, y\_test.shape)

print(‘\_’\*30)

From sklearn.pipeline import Pipeline from sklearn.decomposition import PCA

from sklearn.ensemble import RandomForestClassifier as RFC from sklearn.neighbors import KNeighborsClassifier as KNN from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis as LDA

from sklearn.tree import DecisionTreeClassifier as DT

# Random forest classifier

Steps\_rfc = [(‘Scaler’, StandardScaler()),

(‘clf’, RFC(n\_estimators = 40))]

clf\_rfc = Pipeline(steps=steps\_rfc)

clf\_rfc.fit(X\_train, y\_train) y\_pred\_rfc = clf\_rfc.predict(X\_test)

print(‘RFC accuracy: ‘, accuracy\_score(y\_true=y\_test, y\_pred=y\_pred\_rfc)\*100)

# K nearest neighbor

Steps\_knn = [(‘Scaler’, StandardScaler()),

(‘clf’, KNN(n\_neighbors = 5))]

clf\_knn = Pipeline(steps=steps\_knn)

clf\_knn.fit(X\_train, y\_train) y\_pred\_knn = clf\_knn.predict(X\_test)

print(‘KNN accuracy :’, accuracy\_score(y\_true=y\_test, y\_pred=y\_pred\_knn)\*100)

# Support vector classifier

Steps\_svc = [(‘Scaler’, StandardScaler()),

(‘clf’, SVC())]

clf\_svc = Pipeline(steps=steps\_svc)

clf\_svc.fit(X\_train, y\_train) y\_pred\_svc = clf\_svc.predict(X\_test)

print(‘SVC accuracy :’, accuracy\_score(y\_true=y\_test, y\_pred=y\_pred\_svc)\*100)

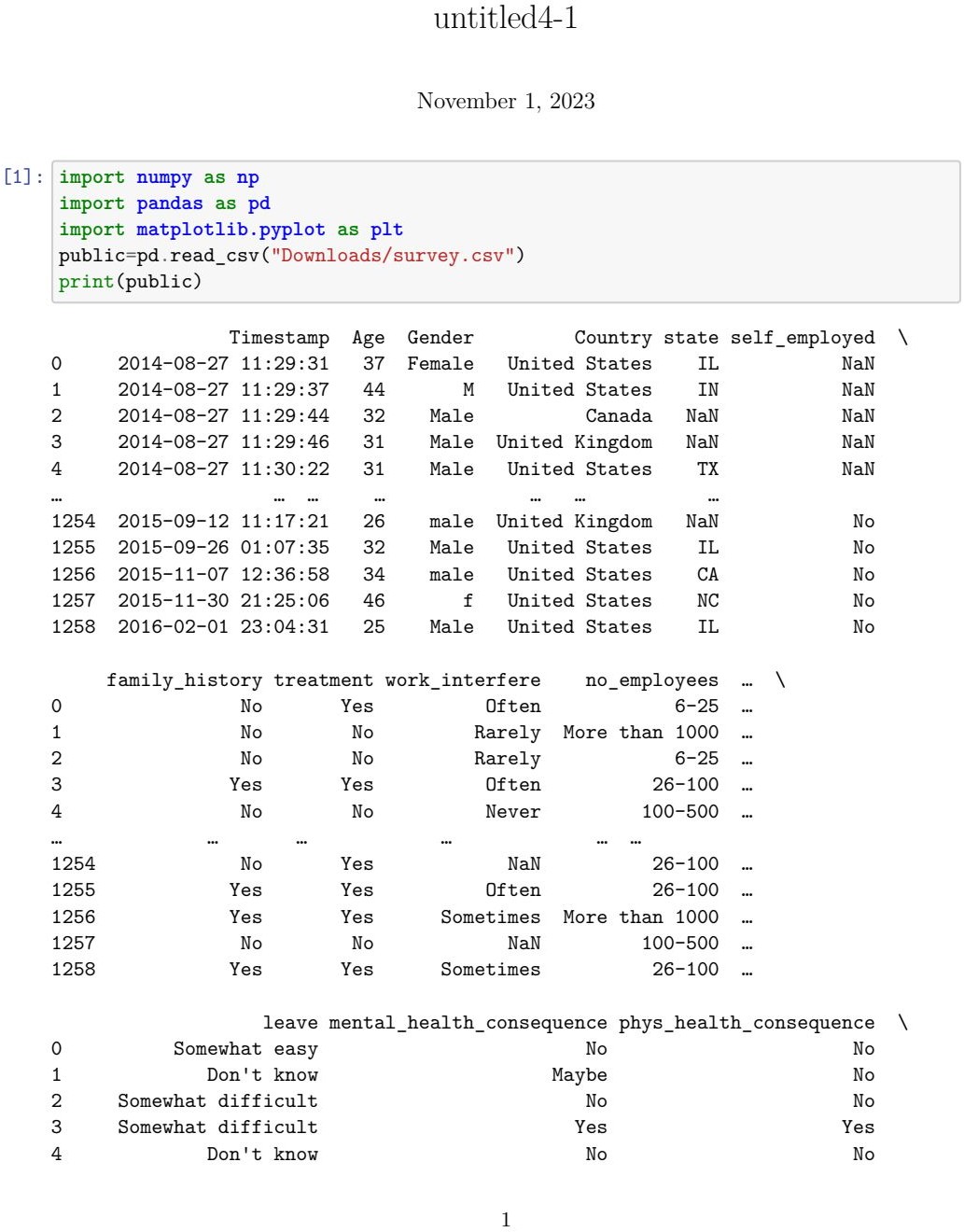
# Decision tree

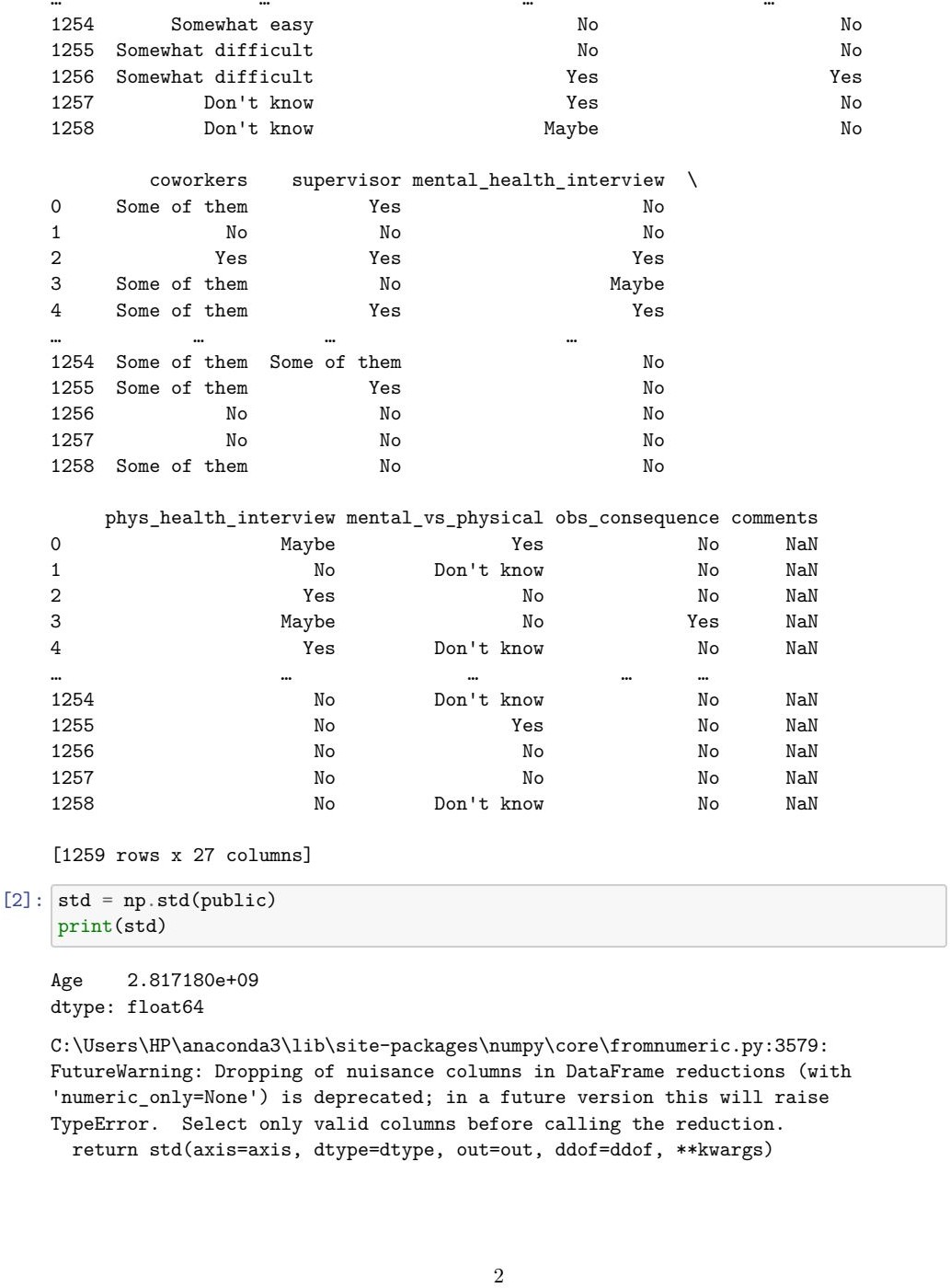
Steps\_dt = [(‘Scaler’, StandardScaler()), (‘clf’, DT())]

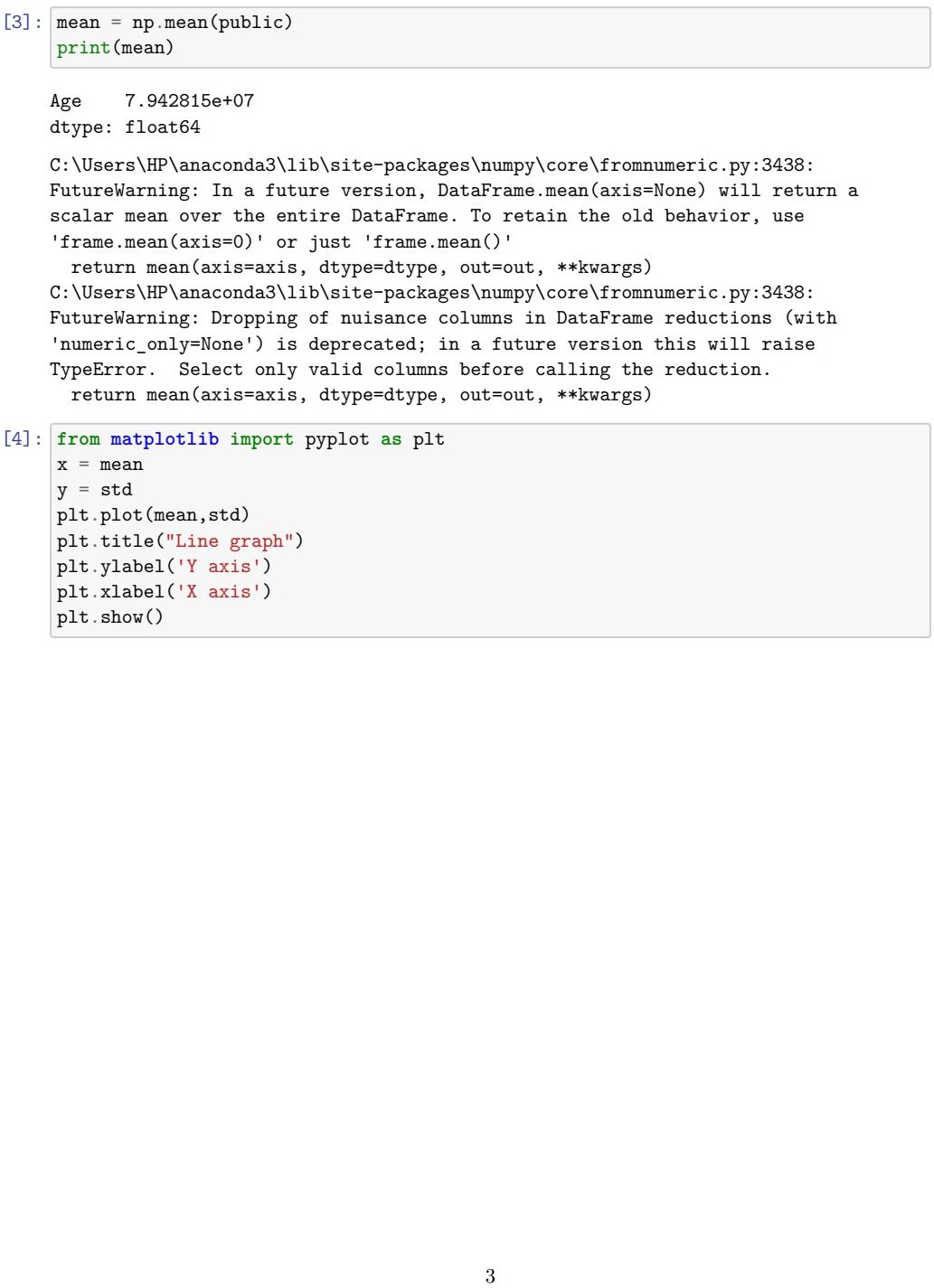
clf\_dt = Pipeline(steps=steps\_dt)

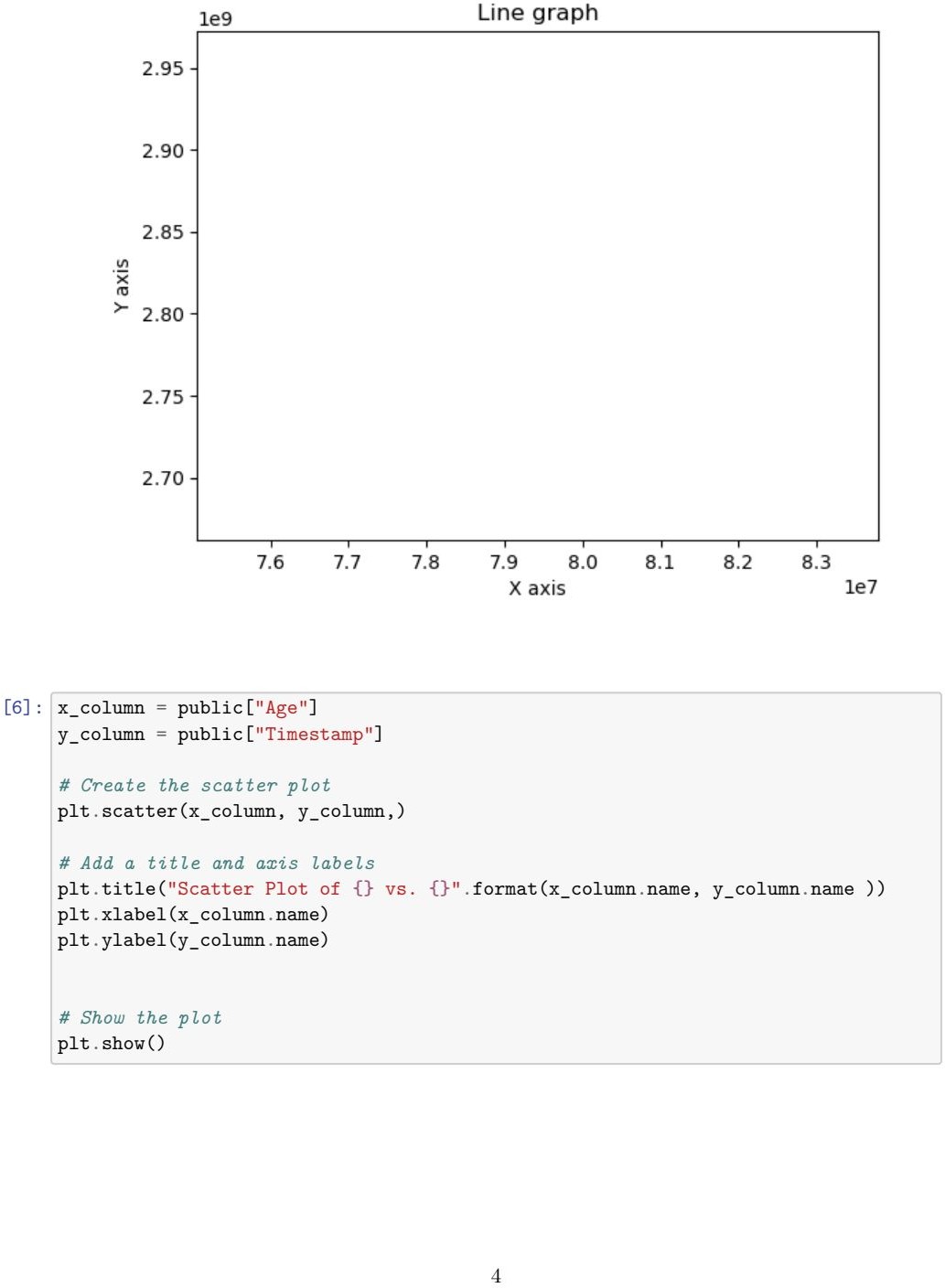
clf\_dt.fit(X\_train, y\_train) impeded = clf\_dt.predict(X\_test)

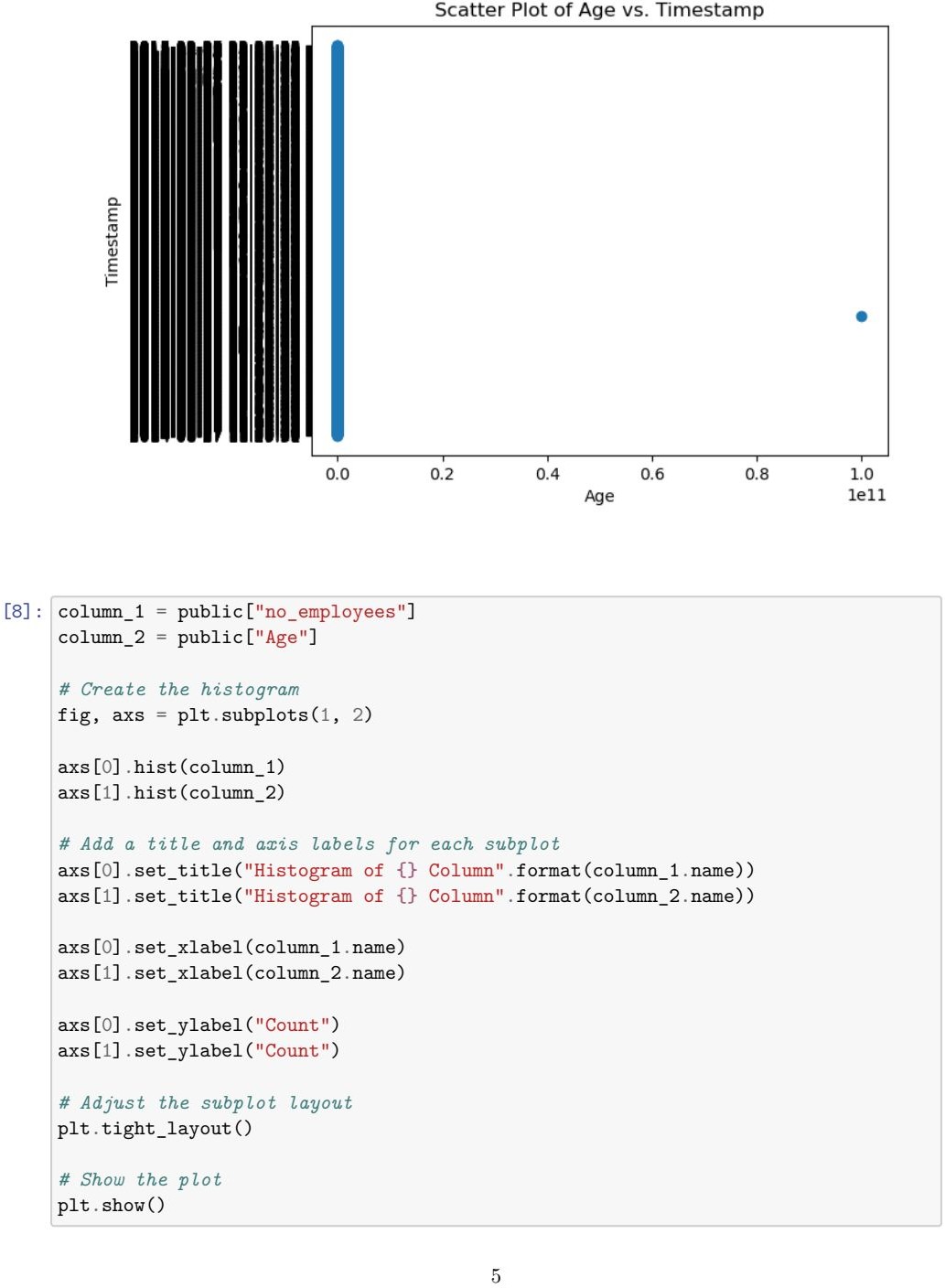
print(‘DT accuracy :’, accuracy\_score(y\_true=y\_test, y\_pred=y\_pred\_dt)\*100)

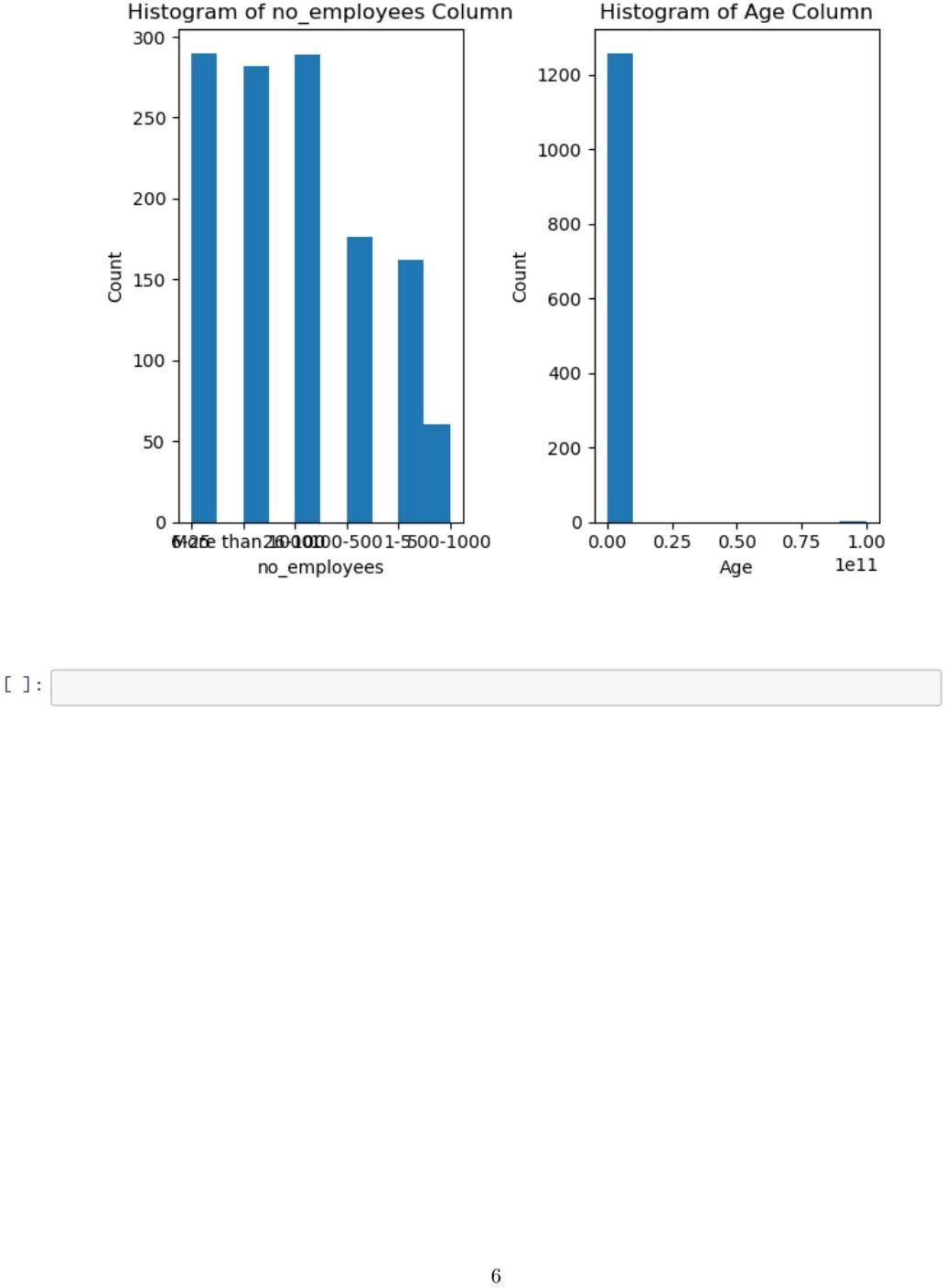












**CONCLUSION:**

Public health is a vital function that requires broad public concern and support in order to fulfill society's interest in assuring the conditions in which people can be healthy.