1Q

import pandas as pd

import numpy as np

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

from sklearn.preprocessing import StandardScaler

# Load dataset

df = pd.read\_csv('smoking.csv')

# Convert categorical variables to numerical

df['gender'] = df['gender'].map({'M': 1, 'F': 0})

df['oral'] = df['oral'].map({'Y': 1, 'N': 0})

df['tartar'] = df['tartar'].map({'Y': 1, 'N': 0})

# Select only numerical columns

df = df.select\_dtypes(include=[np.number]).dropna()

# Define features and target variable

X = df.drop(columns=['smoking', 'ID']) # Drop ID as it's not a feature

y = df['smoking']

# Split dataset 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)

# Scale data

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Select two classes for analysis

class\_labels = y.unique()

if len(class\_labels) >= 2:

class1, class2 = class\_labels[:2]

# Extract feature vectors for two classes

class1\_vecs = X[y == class1]

class2\_vecs = X[y == class2]

# Compute class centroids

centroid1 = np.mean(class1\_vecs, axis=0)

centroid2 = np.mean(class2\_vecs, axis=0)

# Compute spread (standard deviation) for each class

spread1 = np.std(class1\_vecs, axis=0)

spread2 = np.std(class2\_vecs, axis=0)

# Compute Euclidean distance between class centroids

interclass\_distance = np.linalg.norm(centroid1 - centroid2)

print(f"Class {class1} Centroid: {centroid1}")

print(f"Class {class2} Centroid: {centroid2}")

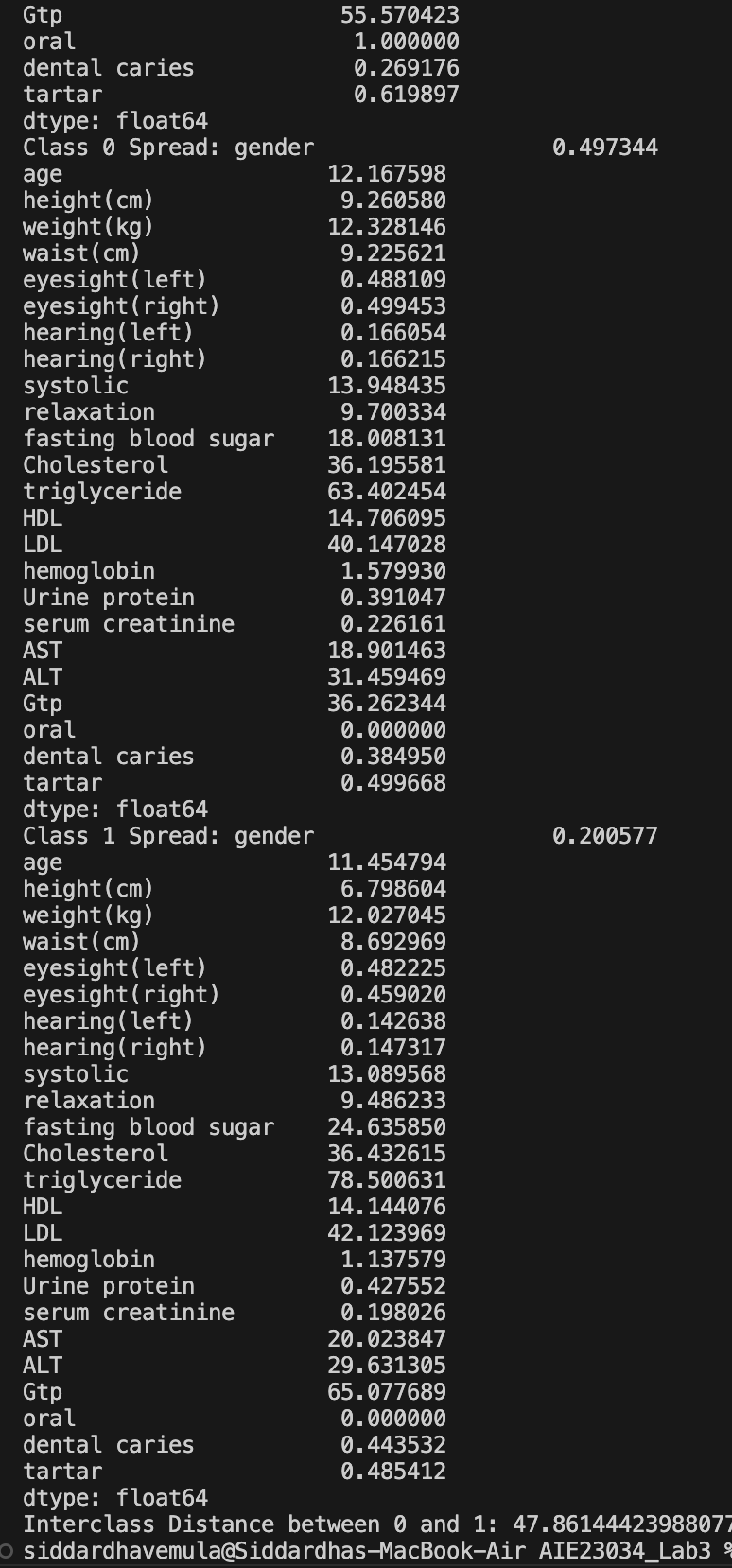
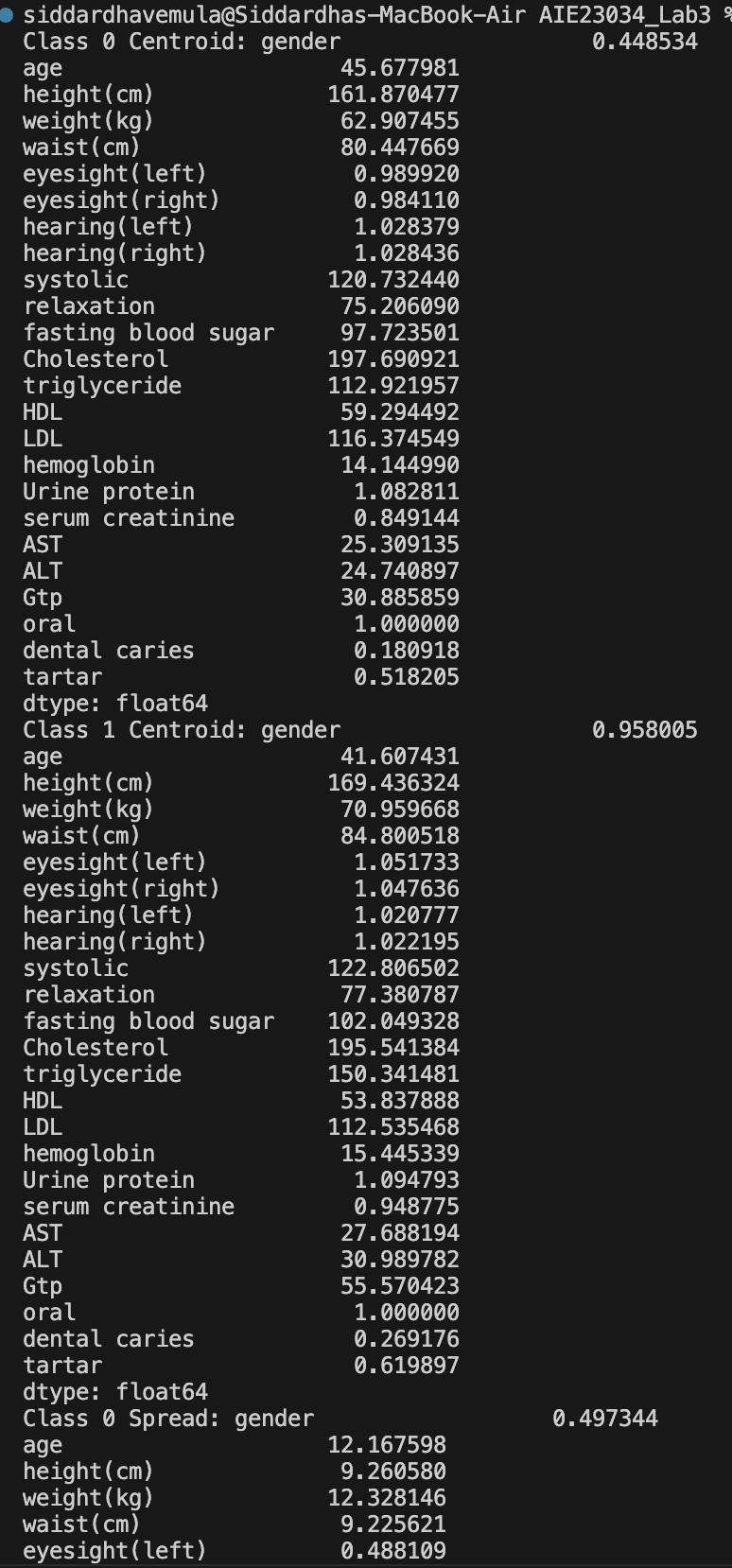
print(f"Class {class1} Spread: {spread1}")

print(f"Class {class2} Spread: {spread2}")

print(f"Interclass Distance between {class1} and {class2}: {interclass\_distance}")

else:

print("Not enough classes to compute interclass distances.")



2Q

import pandas as pd

import matplotlib.pyplot as plt

# Load the dataset

df = pd.read\_csv('smoking.csv')

# Convert categorical variables to numerical

df['gender'] = df['gender'].map({'M': 1, 'F': 0})

df['oral'] = df['oral'].map({'Y': 1, 'N': 0})

df['tartar'] = df['tartar'].map({'Y': 1, 'N': 0})

# Select numeric columns for analysis (excluding ID and target variable)

numeric\_columns = df.select\_dtypes(include=['number']).columns.tolist()

numeric\_columns.remove('ID') # Remove ID column

numeric\_columns.remove('smoking') # Remove target variable

# Drop rows with missing values

df = df[numeric\_columns].dropna()

# Select a feature for histogram plotting

feature = "age" # You can change this to any numeric column from the dataset

# Plot histogram

plt.hist(df[feature], bins=10, color='skyblue', edgecolor='black')

plt.title(f"Histogram of {feature}")

plt.xlabel(feature)

plt.ylabel("Frequency")

plt.show()

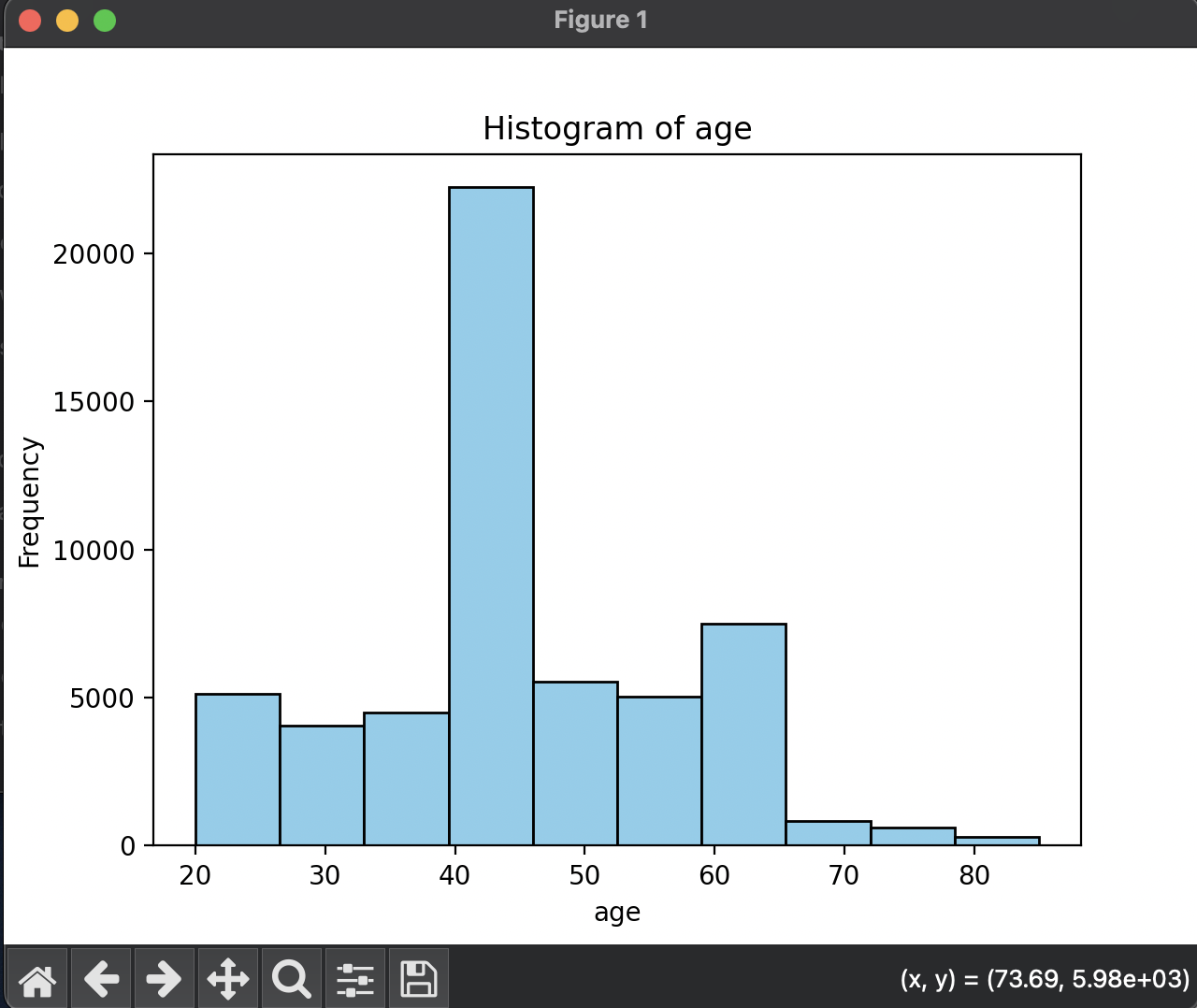
# Calculate mean and variance

mean\_value = df[feature].mean()

variance\_value = df[feature].var()

print(f"Mean of {feature}: {mean\_value}")

print(f"Variance of {feature}: {variance\_value}")



3Q

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

# Load the dataset

df = pd.read\_csv('smoking.csv')

# Convert categorical variables to numerical

df['gender'] = df['gender'].map({'M': 1, 'F': 0})

df['oral'] = df['oral'].map({'Y': 1, 'N': 0})

df['tartar'] = df['tartar'].map({'Y': 1, 'N': 0})

# Select only numeric columns for analysis

numeric\_columns = df.select\_dtypes(include=['number']).columns.tolist()

numeric\_columns.remove('ID') # Remove ID column

numeric\_columns.remove('smoking') # Remove target variable

# Drop rows with missing values

df = df[numeric\_columns].dropna()

# Select two feature vectors for Minkowski distance calculation

vector1 = df.iloc[0].values

vector2 = df.iloc[1].values

# Calculate Minkowski distances for r from 1 to 10

distances = []

r\_values = range(1, 11)

for r in r\_values:

distance = np.linalg.norm(vector1 - vector2, ord=r)

distances.append(distance)

# Plot Minkowski distances

plt.plot(r\_values, distances, marker='o', color='blue', linestyle='dashed')

plt.title("Minkowski Distance between Two Feature Vectors")

plt.xlabel("Order r")

plt.ylabel("Distance")

plt.grid(True)

plt.show()

print("Minkowski distances for r from 1 to 10:", distances)

A screen shot of a graph

AI-generated content may be incorrect.

4Q

import pandas as pd

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

# Load the dataset

df = pd.read\_csv('smoking.csv')

# Convert categorical variables to numerical

df['gender'] = df['gender'].map({'M': 1, 'F': 0})

df['oral'] = df['oral'].map({'Y': 1, 'N': 0})

df['tartar'] = df['tartar'].map({'Y': 1, 'N': 0})

# Select only numeric columns

numeric\_columns = df.select\_dtypes(include=['number']).columns.tolist()

numeric\_columns.remove('ID') # Remove ID column

# Drop rows with missing values

df = df[numeric\_columns].dropna()

# Select features and target for binary classification

X = df.drop("smoking", axis=1) # Features

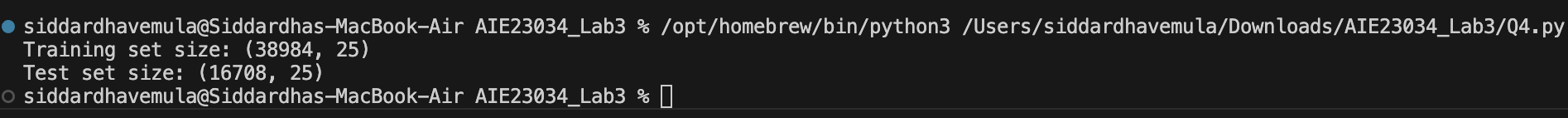
y = df["smoking"] # Target (binary classification)

# Split dataset into training and test sets (70% train, 30% test)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

print("Training set size:", X\_train.shape)

print("Test set size:", X\_test.shape)



5Q

import pandas as pd

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

from sklearn.neighbors import KNeighborsClassifier

# Load the dataset

df = pd.read\_csv('smoking.csv')

# Convert categorical variables to numerical

df['gender'] = df['gender'].map({'M': 1, 'F': 0})

df['oral'] = df['oral'].map({'Y': 1, 'N': 0})

df['tartar'] = df['tartar'].map({'Y': 1, 'N': 0})

# Select only numeric columns

numeric\_columns = df.select\_dtypes(include=['number']).columns.tolist()

numeric\_columns.remove('ID') # Remove ID column

# Drop rows with missing values

df = df[numeric\_columns].dropna()

# Select features and target for classification

X = df.drop("smoking", axis=1) # Features

y = df["smoking"] # Target (binary classification)

# Split dataset into training and test sets (70% train, 30% test)

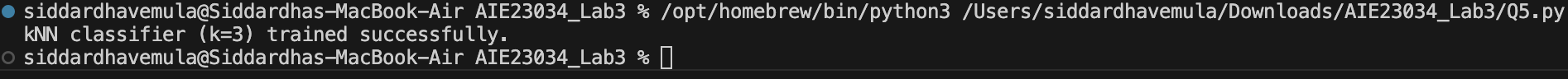
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Train kNN classifier with k=3

knn = KNeighborsClassifier(n\_neighbors=3)

knn.fit(X\_train, y\_train)

print("kNN classifier (k=3) trained successfully.")



6Q

import pandas as pd

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

from sklearn.neighbors import KNeighborsClassifier

# Load the dataset

df = pd.read\_csv('smoking.csv')

# Convert categorical variables to numerical

df['gender'] = df['gender'].map({'M': 1, 'F': 0})

df['oral'] = df['oral'].map({'Y': 1, 'N': 0})

df['tartar'] = df['tartar'].map({'Y': 1, 'N': 0})

# Select only numeric columns

numeric\_columns = df.select\_dtypes(include=['number']).columns.tolist()

numeric\_columns.remove('ID') # Remove ID column

# Drop rows with missing values

df = df[numeric\_columns].dropna()

# Select features and target for classification

X = df.drop("smoking", axis=1) # Features

y = df["smoking"] # Target (binary classification)

# Split dataset into training and test sets (70% train, 30% test)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Train kNN classifier with k=3

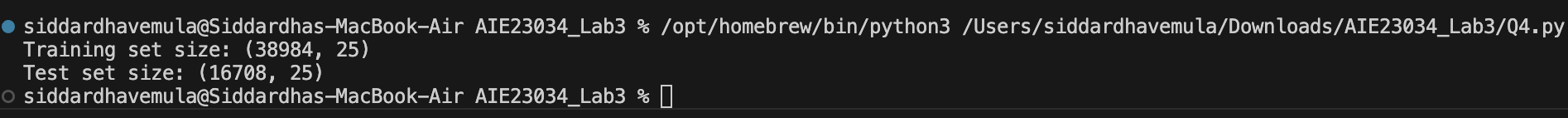
knn = KNeighborsClassifier(n\_neighbors=3)

knn.fit(X\_train, y\_train)

# Test accuracy

accuracy = knn.score(X\_test, y\_test)

print("kNN classifier accuracy on test set:", accuracy)



7Q

import pandas as pd

import numpy as np

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

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

# Load dataset

df = pd.read\_csv('smoking.csv')

# Convert categorical variables to numerical

df['gender'] = df['gender'].map({'M': 1, 'F': 0})

df['oral'] = df['oral'].map({'Y': 1, 'N': 0})

df['tartar'] = df['tartar'].map({'Y': 1, 'N': 0})

# Select only numeric columns

numeric\_columns = df.select\_dtypes(include=['number']).columns.tolist()

numeric\_columns.remove('ID') # Remove ID column

# Drop rows with missing values

df = df[numeric\_columns].dropna()

# Split features and target

X = df.drop(columns=['smoking']) # Features

y = df['smoking'] # Target variable

# Split dataset into training and test sets (80% train, 20% test)

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

# Scale data

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Train classifier

k = 3 # Default k value for kNN

model = KNeighborsClassifier(n\_neighbors=k)

model.fit(X\_train\_scaled, y\_train)

# Predict and analyze behavior

y\_test\_pred = model.predict(X\_test\_scaled)

print("Predictions for test data:", y\_test\_pred)

# Perform classification on a single test vector

test\_vect = X\_test\_scaled[0].reshape(1, -1) # Selecting a single test vector

predicted\_class = model.predict(test\_vect)

print(f"Predicted class for test vector {test\_vect}: {predicted\_class[0]}")

A screenshot of a computer

AI-generated content may be incorrect.

8Q

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

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

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

# Load dataset

df = pd.read\_csv('smoking.csv')

# Convert categorical variables to numerical

df['gender'] = df['gender'].map({'M': 1, 'F': 0})

df['oral'] = df['oral'].map({'Y': 1, 'N': 0})

df['tartar'] = df['tartar'].map({'Y': 1, 'N': 0})

# Select only numeric columns

numeric\_columns = df.select\_dtypes(include=['number']).columns.tolist()

numeric\_columns.remove('ID') # Remove ID column

# Drop rows with missing values

df = df[numeric\_columns].dropna()

# Split features and target

X = df.drop(columns=['smoking']) # Features

y = df['smoking'] # Target variable

# Split dataset into training and test sets (80% train, 20% test)

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

# Scale data

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Evaluate accuracy for different k values

k\_values = range(1, 12)

accuracies = []

for k in k\_values:

model = KNeighborsClassifier(n\_neighbors=k)

model.fit(X\_train\_scaled, y\_train)

y\_pred = model.predict(X\_test\_scaled)

accuracies.append(accuracy\_score(y\_test, y\_pred))

# Plot accuracy vs k values

plt.plot(k\_values, accuracies, marker='o', linestyle='-')

plt.xlabel('Number of Neighbors (k)')

plt.ylabel('Accuracy')

plt.title('kNN Accuracy for Different k Values')

plt.grid(True)

plt.show()

# Compare NN (k=1) with kNN (k=3)

k1\_model = KNeighborsClassifier(n\_neighbors=1)

k1\_model.fit(X\_train\_scaled, y\_train)

k1\_accuracy = accuracy\_score(y\_test, k1\_model.predict(X\_test\_scaled))

k3\_model = KNeighborsClassifier(n\_neighbors=3)

k3\_model.fit(X\_train\_scaled, y\_train)

k3\_accuracy = accuracy\_score(y\_test, k3\_model.predict(X\_test\_scaled))

print(f"Accuracy with k=1 (NN): {k1\_accuracy:.4f}")

print(f"Accuracy with k=3 (kNN): {k3\_accuracy:.4f}")

A graph with a line and a point

AI-generated content may be incorrect.

9Q

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

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

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

# Load dataset

df = pd.read\_csv('smoking.csv')

# Convert categorical variables to numerical

df['gender'] = df['gender'].map({'M': 1, 'F': 0})

df['oral'] = df['oral'].map({'Y': 1, 'N': 0})

df['tartar'] = df['tartar'].map({'Y': 1, 'N': 0})

# Select only numeric columns

numeric\_columns = df.select\_dtypes(include=['number']).columns.tolist()

numeric\_columns.remove('ID') # Remove ID column

# Drop rows with missing values

df = df[numeric\_columns].dropna()

# Split features and target

X = df.drop(columns=['smoking']) # Features

y = df['smoking'] # Target variable

# Split dataset into training and test sets (80% train, 20% test)

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

# Scale data

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Train classifier with optimal k (assuming k=3 based on previous evaluation)

k\_optimal = 3

model = KNeighborsClassifier(n\_neighbors=k\_optimal)

model.fit(X\_train\_scaled, y\_train)

# Predictions

y\_train\_pred = model.predict(X\_train\_scaled)

y\_test\_pred = model.predict(X\_test\_scaled)

# Evaluate confusion matrix and performance metrics

conf\_matrix\_train = confusion\_matrix(y\_train, y\_train\_pred)

conf\_matrix\_test = confusion\_matrix(y\_test, y\_test\_pred)

print("Confusion Matrix (Training Data):\n", conf\_matrix\_train)

print("Confusion Matrix (Test Data):\n", conf\_matrix\_test)

print("Classification Report (Training Data):\n", classification\_report(y\_train, y\_train\_pred))

print("Classification Report (Test Data):\n", classification\_report(y\_test, y\_test\_pred))

# Model evaluation for overfitting or underfitting

train\_accuracy = accuracy\_score(y\_train, y\_train\_pred)

test\_accuracy = accuracy\_score(y\_test, y\_test\_pred)

print(f"Training Accuracy: {train\_accuracy:.4f}")

print(f"Test Accuracy: {test\_accuracy:.4f}")

if train\_accuracy > test\_accuracy + 0.1:

print("Model is overfitting.")

elif test\_accuracy > train\_accuracy:

print("Model might be underfitting.")

else:

print("Model is well-generalized (regular fit).")

A screenshot of a computer

AI-generated content may be incorrect.