```
# Section A

import numpy as np
import pandas as pd

# 1. Create a NumPy array using the range function
array = np.array(range(10))
print("NumPy Array:", array)

# 2. Print the data type of each element within the array
print("Data types of elements:")
for element in array:
    print(type(element))

# 3. Create a pandas dataframe from the python dictionary for Car Record
car_data = {
    'Brand': ['Toyota', 'Honda', 'Ford'],
    'Model': ['Corolla', 'Civic', 'F-150'],
    'Year': [2010, 2012, 2015]
}
car_df = pd.DataFrame(car_data)
print("\nCar DataFrame:")
print(car_df)

# 4. Add one column in the above dataframe
car_df['Price'] = [8000, 9000, 15000]
print("\nUpdated Car DataFrame:")
print(car_df)



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


# Section B

import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, accuracy_score

# 1. Load the fracture.csv data into pandas dataframe and print the first 15 records
df = pd.read_csv("fracture.csv")
print("First 15 Records:")
print(df.head(15))

# 2. Add a new column named “bmi” using the formula
df['bmi'] = df['weight_kg'] / ((df['height_cm'] / 100) ** 2)

# 🔁 Convert 'sex' and 'fracture' columns to numeric
df['sex'] = df['sex'].map({'M': 1, 'F': 0})
df['fracture'] = df['fracture'].map({'fracture': 1, 'no fracture': 0})

# 3. Split the data set into test and train
X = df[['age', 'sex', 'bmi', 'bmd']]
y = df['fracture']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 4. Build a logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# 5. Apply the model on to the test and train data and plot prediction outcomes
y_pred_train = model.predict(X_train)
y_pred_test = model.predict(X_test)

# Plotting predictions
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.scatter(y_train, y_pred_train, alpha=0.5, color='blue')
plt.title('Train Data Predictions')
plt.xlabel('Actual')
plt.ylabel('Predicted')
```

```
plt.subplot(1, 2, 2)
plt.scatter(y_test, y_pred_test, alpha=0.5, color='green')
plt.title('Test Data Predictions')
plt.xlabel('Actual')
plt.ylabel('Predicted')

plt.tight_layout()
plt.show()

# 6. Calculate accuracy using confusion matrix
cm = confusion_matrix(y_test, y_pred_test)
acc = accuracy_score(y_test, y_pred_test)
print("\nConfusion Matrix:")
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()

print(f"Accuracy: {acc:.2f}")



#set2
# Section A

import numpy as np
import pandas as pd

# 1. Create a 3x3 NumPy array with values between 0 to 9
array = np.arange(9).reshape(3, 3)
print("3x3 NumPy Array:")
print(array)

# 2. Print the data type of element of the array
print("\nData type of array elements:")
print(array.dtype)

# 3. Create a pandas dataframe for Mobile Phone Details
mobile_data = {
    'Brand': ['Samsung', 'Apple', 'OnePlus'],
    'Model': ['Galaxy S23', 'iPhone 14', 'OnePlus 11'],
    'Price': [70000, 80000, 60000]
}
df_mobile = pd.DataFrame(mobile_data)
print("\nMobile Phone DataFrame:")
print(df_mobile)

# 4. Add one more column to the above dataframe
df_mobile['Storage'] = ['128GB', '256GB', '256GB']
print("\nDataFrame after adding 'Storage' column:")
print(df_mobile)



# Section B

import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, accuracy_score

# 1. Load the fracture.csv and display the last five records
df = pd.read_csv("fracture.csv")
print("Last 5 Records:")
print(df.tail())

# 2. Add BMI column: BMI = weight_kg / (height_cm/100)^2
df['bmi'] = df['weight_kg'] / ((df['height_cm'] / 100) ** 2)

# Convert 'sex' and 'fracture' to numeric
df['sex'] = df['sex'].map({'M': 1, 'F': 0})
df['fracture'] = df['fracture'].map({'fracture': 1, 'no fracture': 0})

# 3. Split dataset into train/test (70:30)
X = df[['age', 'sex', 'bmi', 'bmd']]
y = df['fracture']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
# 4. Build SVM model
svm_model = SVC(kernel='linear')
svm_model.fit(X_train, y_train)

# 5. Predict and plot outcomes
y_pred_train = svm_model.predict(X_train)
y_pred_test = svm_model.predict(X_test)

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.scatter(y_train, y_pred_train, alpha=0.5, color='orange')
plt.title('Train Data Predictions')
plt.xlabel('Actual')
plt.ylabel('Predicted')

plt.subplot(1, 2, 2)
plt.scatter(y_test, y_pred_test, alpha=0.5, color='purple')
plt.title('Test Data Predictions')
plt.xlabel('Actual')
plt.ylabel('Predicted')

plt.tight_layout()
plt.show()

# 6. Confusion Matrix and Accuracy
cm = confusion_matrix(y_test, y_pred_test)
acc = accuracy_score(y_test, y_pred_test)
print("\nConfusion Matrix:")
ConfusionMatrixDisplay(confusion_matrix=cm).plot()
plt.show()

print(f"\nAccuracy: {acc:.2f}")



#✅ SET-03 – Section A

import numpy as np
import pandas as pd

# 1. Create a NumPy array with elements that are multiples of 5
array = np.arange(5, 55, 5)
print("NumPy array with multiples of 5:")
print(array)

# 2. Print the standard deviation of the array
std_dev = np.std(array)
print("\nStandard Deviation:", std_dev)

# 3. Create a pandas dataframe for Book details
book_data = {
    'Title': ['Python 101', 'Data Science Handbook', 'AI Basics'],
    'Author': ['John Doe', 'Jane Smith', 'Alice Brown'],
    'Price': [250, 500, 300]
}
df_books = pd.DataFrame(book_data)
print("\nBook Details DataFrame:")
print(df_books)

# 4. Rename one column
df_books.rename(columns={'Price': 'Cost'}, inplace=True)
print("\nDataFrame after renaming 'Price' to 'Cost':")
print(df_books)



#✅ SET-03 – Section B

import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, accuracy_score

# 1. Load the fracture.csv into DataFrame
df = pd.read_csv("fracture.csv")
print("Fracture Data (First 5 rows):")
print(df.head())
```

```
# 2. Add a new column 'bmi'
df['bmi'] = df['weight_kg'] / ((df['height_cm'] / 100) ** 2)

# Encode categorical variables
df['sex'] = df['sex'].map({'M': 1, 'F': 0})
df['fracture'] = df['fracture'].map({'fracture': 1, 'no fracture': 0})

# 3. Split the data into 80% train, 20% test
X = df[['age', 'sex', 'bmi', 'bmd']]
y = df['fracture']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 4. Build SVM model
svm_model = SVC(kernel='linear')
svm_model.fit(X_train, y_train)

# 5. Predict and plot
y_pred_train = svm_model.predict(X_train)
y_pred_test = svm_model.predict(X_test)

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.scatter(y_train, y_pred_train, color='teal', alpha=0.6)
plt.title('Train Data Predictions')
plt.xlabel('Actual')
plt.ylabel('Predicted')

plt.subplot(1, 2, 2)
plt.scatter(y_test, y_pred_test, color='orange', alpha=0.6)
plt.title('Test Data Predictions')
plt.xlabel('Actual')
plt.ylabel('Predicted')

plt.tight_layout()
plt.show()

# 6. Confusion matrix & accuracy
cm = confusion_matrix(y_test, y_pred_test)
acc = accuracy_score(y_test, y_pred_test)

print("\nConfusion Matrix:")
ConfusionMatrixDisplay(confusion_matrix=cm).plot()
plt.show()

print(f"\nModel Accuracy: {acc:.2f}")



# SET-04 – Section A

import numpy as np
import pandas as pd

# 1. Create a NumPy array from a list
my_list = [12, 45, 67, 89, 23, 56]
np_array = np.array(my_list)
print("NumPy Array:")
print(np_array)

# 2. Print max and min values
print("\nMaximum value:", np.max(np_array))
print("Minimum value:", np.min(np_array))

# 3. Create pandas dataframe from Student Record dictionary
student_data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [20, 21, 19],
    'Grade': ['A', 'B', 'A']
}
df_students = pd.DataFrame(student_data)
print("\nStudent Record DataFrame:")
print(df_students)

# 4. Delete one column from the DataFrame
df_students.drop(columns='Grade', inplace=True)
```

```
print("\nDataFrame after deleting 'Grade' column:")
print(df_students)



# SET-04 – Section B

import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, accuracy_score

# 1. Load fracture.csv and print first 15 records
df = pd.read_csv("fracture.csv")
print("First 15 Records:")
print(df.head(15))

# 2. Add 'bmi' column
df['bmi'] = df['weight_kg'] / ((df['height_cm'] / 100) ** 2)

# 3. Encode 'sex' and 'fracture' for ML model
df['sex'] = df['sex'].map({'M': 1, 'F': 0})
df['fracture'] = df['fracture'].map({'fracture': 1, 'no fracture': 0})

# Split dataset
X = df[['age', 'sex', 'bmi', 'bmd']]
y = df['fracture']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 4. Build logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# 5. Predict and plot outcomes
y_pred_train = model.predict(X_train)
y_pred_test = model.predict(X_test)

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.scatter(y_train, y_pred_train, color='blue', alpha=0.5)
plt.title("Train Data Predictions")
plt.xlabel("Actual")
plt.ylabel("Predicted")

plt.subplot(1, 2, 2)
plt.scatter(y_test, y_pred_test, color='green', alpha=0.5)
plt.title("Test Data Predictions")
plt.xlabel("Actual")
plt.ylabel("Predicted")

plt.tight_layout()
plt.show()

# 6. Confusion matrix and accuracy
cm = confusion_matrix(y_test, y_pred_test)
acc = accuracy_score(y_test, y_pred_test)

print("\nConfusion Matrix:")
ConfusionMatrixDisplay(confusion_matrix=cm).plot()
plt.show()

print(f"\nAccuracy of Logistic Regression Model: {acc:.2f}")



#SET-05 – Section A

import numpy as np
import pandas as pd

# 1. Create a 5x5 NumPy array with values from 0 to 24
array_5x5 = np.arange(25).reshape(5, 5)
print("5x5 NumPy Array:")
print(array_5x5)

# 2. Print the size and shape of the array
print("\nArray Size:", array_5x5.size)
```

```
print("Array Shape:", array_5x5.shape)

# 3. Create a pandas dataframe for Mobile phone details
mobile_data = {
    'Brand': ['Samsung', 'Apple', 'Xiaomi'],
    'Model': ['S21', 'iPhone 13', 'Redmi Note 10'],
    'Price': [70000, 80000, 15000]
}
df_mobiles = pd.DataFrame(mobile_data)
print("\nMobile Phone DataFrame:")
print(df_mobiles)

# 4. Add one more column (e.g., 'RAM')
df_mobiles['RAM'] = ['8GB', '6GB', '4GB']
print("\nDataFrame after adding RAM column:")
print(df_mobiles)



#SET-05 – Section B

import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, accuracy_score

# 1. Load fracture.csv
df = pd.read_csv("fracture.csv")
print("Last 5 Records:")
print(df.tail())

# 2. Add BMI column
df['bmi'] = df['weight_kg'] / ((df['height_cm'] / 100) ** 2)

# 3. Encode categorical data
df['sex'] = df['sex'].map({'M': 1, 'F': 0})
df['fracture'] = df['fracture'].map({'fracture': 1, 'no fracture': 0})

# 4. Prepare data
X = df[['age', 'sex', 'bmi', 'bmd']]
y = df['fracture']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# 5. Build KNN model
knn_model = KNeighborsClassifier(n_neighbors=3)
knn_model.fit(X_train, y_train)

# Predict
y_pred_train = knn_model.predict(X_train)
y_pred_test = knn_model.predict(X_test)

# Plot predictions
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.scatter(y_train, y_pred_train, alpha=0.6, color='blue')
plt.title("Train Data Predictions")
plt.xlabel("Actual")
plt.ylabel("Predicted")

plt.subplot(1, 2, 2)
plt.scatter(y_test, y_pred_test, alpha=0.6, color='orange')
plt.title("Test Data Predictions")
plt.xlabel("Actual")
plt.ylabel("Predicted")

plt.tight_layout()
plt.show()

# 6. Confusion matrix & accuracy
cm = confusion_matrix(y_test, y_pred_test)
acc = accuracy_score(y_test, y_pred_test)

print("\nConfusion Matrix:")
ConfusionMatrixDisplay(confusion_matrix=cm).plot()
plt.show()
```

```
print(f"\nKNN Model Accuracy: {acc:.2f}")



#SET-06 – Section A

import numpy as np
import pandas as pd

# 1. Create a NumPy array using random function
random_array = np.random.rand(5, 5)
print("Random NumPy Array (5x5):")
print(random_array)

# 2. Print the average value in the above array
average_value = np.mean(random_array)
print("\nAverage value of the array:", average_value)

# 3. Create a pandas dataframe from a Python dictionary for Book details
book_data = {
    'Title': ['The Alchemist', '1984', 'To Kill a Mockingbird'],
    'Author': ['Paulo Coelho', 'George Orwell', 'Harper Lee'],
    'Price': [350, 400, 300]
}
df_books = pd.DataFrame(book_data)
print("\nBook DataFrame:")
print(df_books)

# 4. Rename one column from the above dataframe (e.g., rename 'Price' to 'Cost')
df_books.rename(columns={'Price': 'Cost'}, inplace=True)
print("\nRenamed DataFrame:")
print(df_books)



#SET-06 – Section B

import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, accuracy_score

# 1. Load fracture.csv
df = pd.read_csv("fracture.csv")
print("First 5 Records:")
print(df.head())

# 2. Add new column 'bmi'
df['bmi'] = df['weight_kg'] / ((df['height_cm'] / 100) ** 2)

# 3. Encode 'sex' and 'fracture'
df['sex'] = df['sex'].map({'M': 1, 'F': 0})
df['fracture'] = df['fracture'].map({'fracture': 1, 'no fracture': 0})

# 4. Split the data in 20:80 ratio
X = df[['age', 'sex', 'bmi', 'bmd']]
y = df['fracture']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 5. Build SVM model
svm_model = SVC(kernel='linear')
svm_model.fit(X_train, y_train)

# Predictions
y_pred_train = svm_model.predict(X_train)
y_pred_test = svm_model.predict(X_test)

# Plotting predictions
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.scatter(y_train, y_pred_train, color='blue', alpha=0.5)
plt.title("Train Data Predictions")
plt.xlabel("Actual")
plt.ylabel("Predicted")

plt.subplot(1, 2, 2)
plt.scatter(y_test, y_pred_test, color='green', alpha=0.5)
```

```
plt.title("Test Data Predictions")
plt.xlabel("Actual")
plt.ylabel("Predicted")

plt.tight_layout()
plt.show()

# 6. Confusion Matrix & Accuracy
cm = confusion_matrix(y_test, y_pred_test)
acc = accuracy_score(y_test, y_pred_test)

print("\nConfusion Matrix:")
ConfusionMatrixDisplay(confusion_matrix=cm).plot()
plt.show()

print(f"\nSVM Model Accuracy: {acc:.2f}")
```