import pandas as pd

import matplotlib.pyplot as plt

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

def get\_country(image\_id):

return roadpredict.loc[roadpredict['Image ID'] == image\_id, 'Country'].values[0]

def get\_dataset\_source(image\_id):

return roadpredict.loc[roadpredict['Image ID'] == image\_id, 'Dataset Source'].values[0]

def get\_image\_resolution(image\_id):

return roadpredict.loc[roadpredict['Image ID'] == image\_id, 'Image Resolution'].values[0]

def get\_pothole\_count(image\_id):

return roadpredict.loc[roadpredict['Image ID'] == image\_id, 'Pothole Count'].values[0]

def get\_crack\_count(image\_id):

return roadpredict.loc[roadpredict['Image ID'] == image\_id, 'Crack Count'].values[0]

def get\_dataset\_split(image\_id):

return roadpredict.loc[roadpredict['Image ID'] == image\_id, 'Dataset Split'].values[0]

def get\_dataset\_size(image\_id):

return roadpredict.loc[roadpredict['Image ID'] == image\_id, 'Dataset Size'].values[0]

def get\_augmentation\_techniques(image\_id):

return roadpredict.loc[roadpredict['Image ID'] == image\_id, 'Dataset Augmentation Techniques'].values[0]

def get\_model\_architecture(image\_id):

return roadpredict.loc[roadpredict['Image ID'] == image\_id, 'Model Architecture'].values[0]

def get\_training\_time(image\_id):

return roadpredict.loc[roadpredict['Image ID'] == image\_id, 'Training Time'].values[0]

def get\_inference\_time(image\_id):

return roadpredict.loc[roadpredict['Image ID'] == image\_id, 'Inference Time'].values[0]

def get\_precision(image\_id):

return roadpredict.loc[roadpredict['Image ID'] == image\_id, 'Precision'].values[0]

def get\_recall(image\_id):

return roadpredict.loc[roadpredict['Image ID'] == image\_id, 'Recall'].values[0]

def get\_mean\_average\_precision(image\_id):

return roadpredict.loc[roadpredict['Image ID'] == image\_id, 'Mean Average Precision (mAP)'].values[0]

def calculate\_f1\_score(image\_id):

precision = get\_precision(image\_id)

recall = get\_recall(image\_id)

f1\_score = 2 \* (precision \* recall) / (precision + recall)

return f1\_score

def determine\_road\_condition(image\_id):

potholes = get\_pothole\_count(image\_id)

cracks = get\_crack\_count(image\_id)

safety\_score = potholes \* 0.3 + cracks \* 0.7

print("\n")

if safety\_score <= 5:

return "\*\*\*The road is safe to drive\*\*\*"

else:

return "\_\_\_\_\_\_The road needs repair Be Caution \_\_\_\_\_\_\_\_\_"

def main():

image\_id = int(input("Enter the Image ID: "))

print("\nFeatures for Image ID", image\_id, ":")

features = {

"Country": get\_country(image\_id),

"Dataset Source": get\_dataset\_source(image\_id),

"Image Resolution": get\_image\_resolution(image\_id),

"Pothole Count": get\_pothole\_count(image\_id),

"Crack Count": get\_crack\_count(image\_id),

"Dataset Split": get\_dataset\_split(image\_id),

"Dataset Size": get\_dataset\_size(image\_id),

"Dataset Augmentation Techniques": get\_augmentation\_techniques(image\_id),

"Model Architecture": get\_model\_architecture(image\_id),

"Training Time": get\_training\_time(image\_id),

"Inference Time": get\_inference\_time(image\_id),

"Precision": get\_precision(image\_id),

"Recall": get\_recall(image\_id),

"Mean Average Precision (mAP)": get\_mean\_average\_precision(image\_id),

"\nF1 Score": calculate\_f1\_score(image\_id)

}

for feature, value in features.items():

print(f"{feature}: {value}")

print(determine\_road\_condition(image\_id))

plot\_graph = input("\nDo you want to display the graphs? (y/n): ").lower()

if plot\_graph == 'y':

plot\_feature\_distribution('Pothole Count')

plot\_feature\_distribution('Crack Count')

def plot\_feature\_distribution(feature\_name):

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

roadpredict[feature\_name].value\_counts().plot(kind='bar')

plt.title(f'Distribution of {feature\_name}')

plt.xlabel(feature\_name)

plt.ylabel('Count')

plt.show()

# Call the main function

if \_\_name\_\_ == "\_\_main\_\_":

main()