```
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()
```