

```

from google.colab import files
import zipfile
import os

print("Please upload your dataset zip file:")
uploaded = files.upload()

# Extract uploaded ZIP file to ./dataset directory
for fname in uploaded.keys():
    if fname.endswith('.zip'):
        with zipfile.ZipFile(fname, 'r') as zip_ref:
            zip_ref.extractall('./dataset')

# Verify extraction by listing dataset folder contents
dataset_path = './dataset'
print("Dataset extracted at:", dataset_path)
print("Subfolders/classes found:", os.listdir(dataset_path))

Please upload your dataset zip file:

<IPython.core.display.HTML object>

Saving archive.zip to archive (1).zip
Dataset extracted at: ./dataset
Subfolders/classes found: ['dataset', 'diabetic_retinopathy',
'cataract', 'normal', 'glaucoma']

import os
import shutil
import random

def get_all_images(folder):
    img_files = []
    for root, _, files in os.walk(folder):
        for file in files:
            if file.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp',
            '.tiff')):
                img_files.append(os.path.join(root, file))
    return img_files

dataset_path = './dataset'
output_path = './split_dataset'

train_ratio = 0.7
val_ratio = 0.1
test_ratio = 0.2

# Create output directories and class subfolders
for split in ['train', 'val', 'test']:
    split_dir = os.path.join(output_path, split)
    os.makedirs(split_dir, exist_ok=True)

```



```

data_folders = {
    'train': './split_dataset/train',
    'val': './split_dataset/val',
    'test': './split_dataset/test'
}

datasets = {x: ImageFolder(root=path, transform=transform) for x, path
in data_folders.items()}
dataloaders = {x: DataLoader(datasets[x], batch_size=32,
shuffle=False, num_workers=2) for x in data_folders}

# Load MobileNetV2 pre-trained model
model =
models.mobilenet_v2(weights=models.MobileNet_V2_Weights.DEFAULT)
# Remove classifier for feature extraction
model = nn.Sequential(*list(model.children())[:-1])
model.to(device)
model.eval()

def extract_and_save_features():
    for phase in ['train', 'val', 'test']:
        features_path = f"{phase}_features.npy"
        labels_path = f"{phase}_labels.npy"

        if os.path.exists(features_path) and
os.path.exists(labels_path):
            print(f"Features and labels for {phase} already saved.
Loading them.")
            features = np.load(features_path)
            labels = np.load(labels_path)
        else:
            print(f"Extracting features for {phase} dataset...")
            features = []
            labels = []
            with torch.no_grad():
                for inputs, targets in dataloaders[phase]:
                    inputs = inputs.to(device)
                    output = model(inputs)
                    output = output.view(output.size(0), -1)
                    features.append(output.cpu().numpy())
                    labels.extend(targets.numpy())

            features = np.concatenate(features, axis=0)
            labels = np.array(labels)

            np.save(features_path, features)
            np.save(labels_path, labels)
            print(f"Saved {phase} features and labels.")

    print(f"{phase} features shape: {features.shape}")

```

```

        print(f"{phase} labels shape: {labels.shape}")

extract_and_save_features()

Downloading: "https://download.pytorch.org/models/mobilenet_v2-
7ebf99e0.pth" to /root/.cache/torch/hub/checkpoints/mobilenet_v2-
7ebf99e0.pth

100%|██████████| 13.6M/13.6M [00:00<00:00, 149MB/s]

Extracting features for train dataset...

Saved train features and labels.
train features shape: (3854, 62720)
train labels shape: (3854,)
Extracting features for val dataset...
Saved val features and labels.
val features shape: (792, 62720)
val labels shape: (792,)
Extracting features for test dataset...
Saved test features and labels.
test features shape: (1530, 62720)
test labels shape: (1530,)

import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Load features and labels
train_features = np.load('train_features.npy')
train_labels = np.load('train_labels.npy')
val_features = np.load('val_features.npy')
val_labels = np.load('val_labels.npy')
test_features = np.load('test_features.npy')
test_labels = np.load('test_labels.npy')

# Initialize models
models = {
    'RandomForest': RandomForestClassifier(n_estimators=100,
random_state=42),
    'DecisionTree': DecisionTreeClassifier(random_state=42),
    'XGBoost': XGBClassifier(use_label_encoder=False,
eval_metric='mlogloss', random_state=42)
}

```

```

# Assuming class names are known or extracted from dataset folder
names
import os
class_names = sorted(os.listdir('./split_dataset/train'))

results = {}

for name, model in models.items():
    print(f'Training {name}...')
    model.fit(train_features, train_labels)

    test_preds = model.predict(test_features)

    acc = accuracy_score(test_labels, test_preds)
    prec = precision_score(test_labels, test_preds,
average='weighted')
    rec = recall_score(test_labels, test_preds, average='weighted')
    f1 = f1_score(test_labels, test_preds, average='weighted')
    print(f'{name} Test Accuracy: {acc:.4f}')
    print(classification_report(test_labels, test_preds,
target_names=class_names))

    # Plot confusion matrix
    cm = confusion_matrix(test_labels, test_preds)
    plt.figure(figsize=(8,6))
    sns.heatmap(cm, annot=True, fmt='d', xticklabels=class_names,
yticklabels=class_names, cmap='Blues')
    plt.title(f'{name} Confusion Matrix')
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.show()

    results[name] = {'accuracy': acc, 'precision': prec, 'recall':
rec, 'f1_score': f1}

# Summary
print("Models performance summary:")
for name, metrics in results.items():
    print(f"{name}: Accuracy={metrics['accuracy']:.4f},
Precision={metrics['precision']:.4f}, Recall={metrics['recall']:.4f},
F1-score={metrics['f1_score']:.4f}")

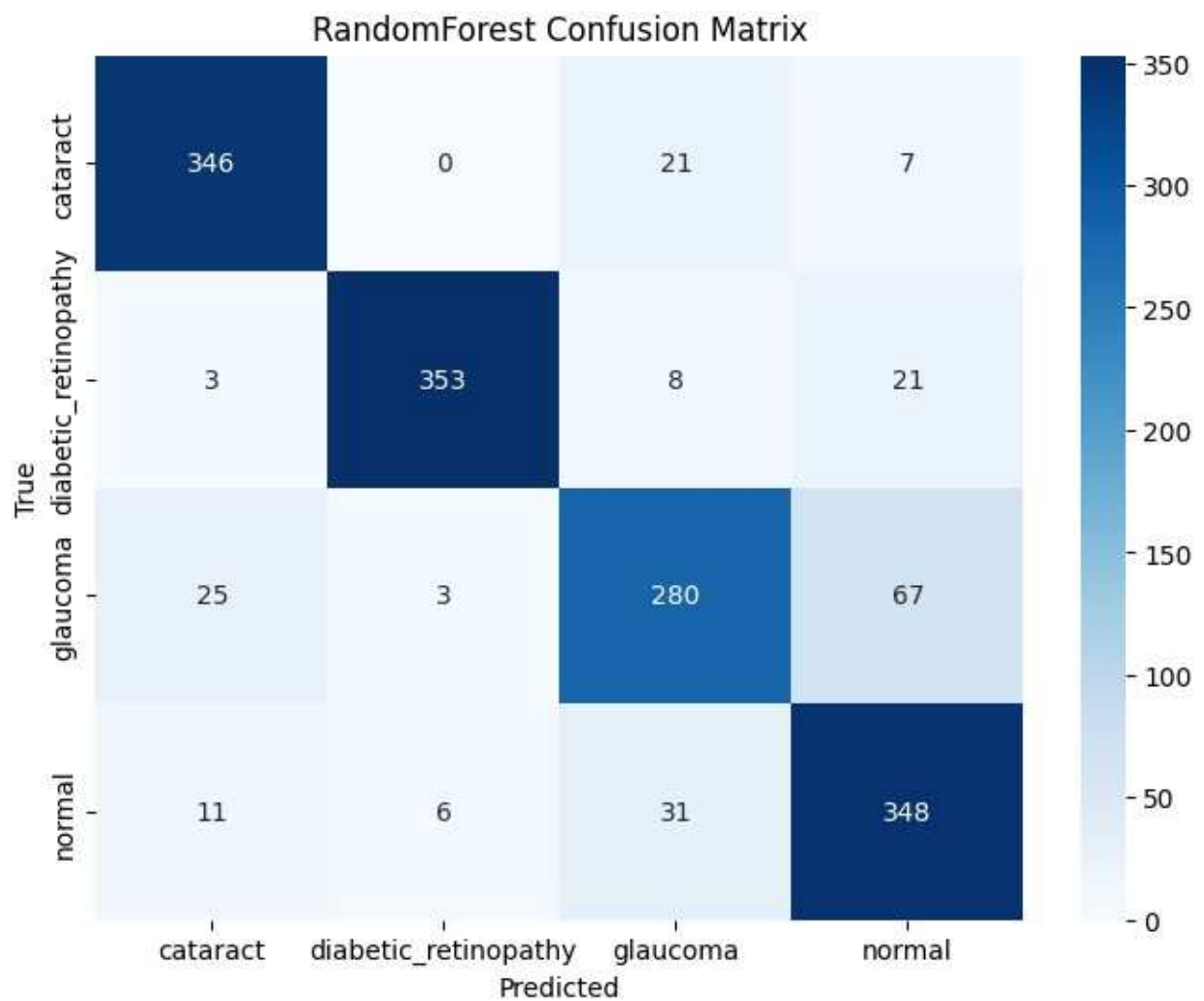
```

Training RandomForest...

RandomForest Test Accuracy: 0.8673

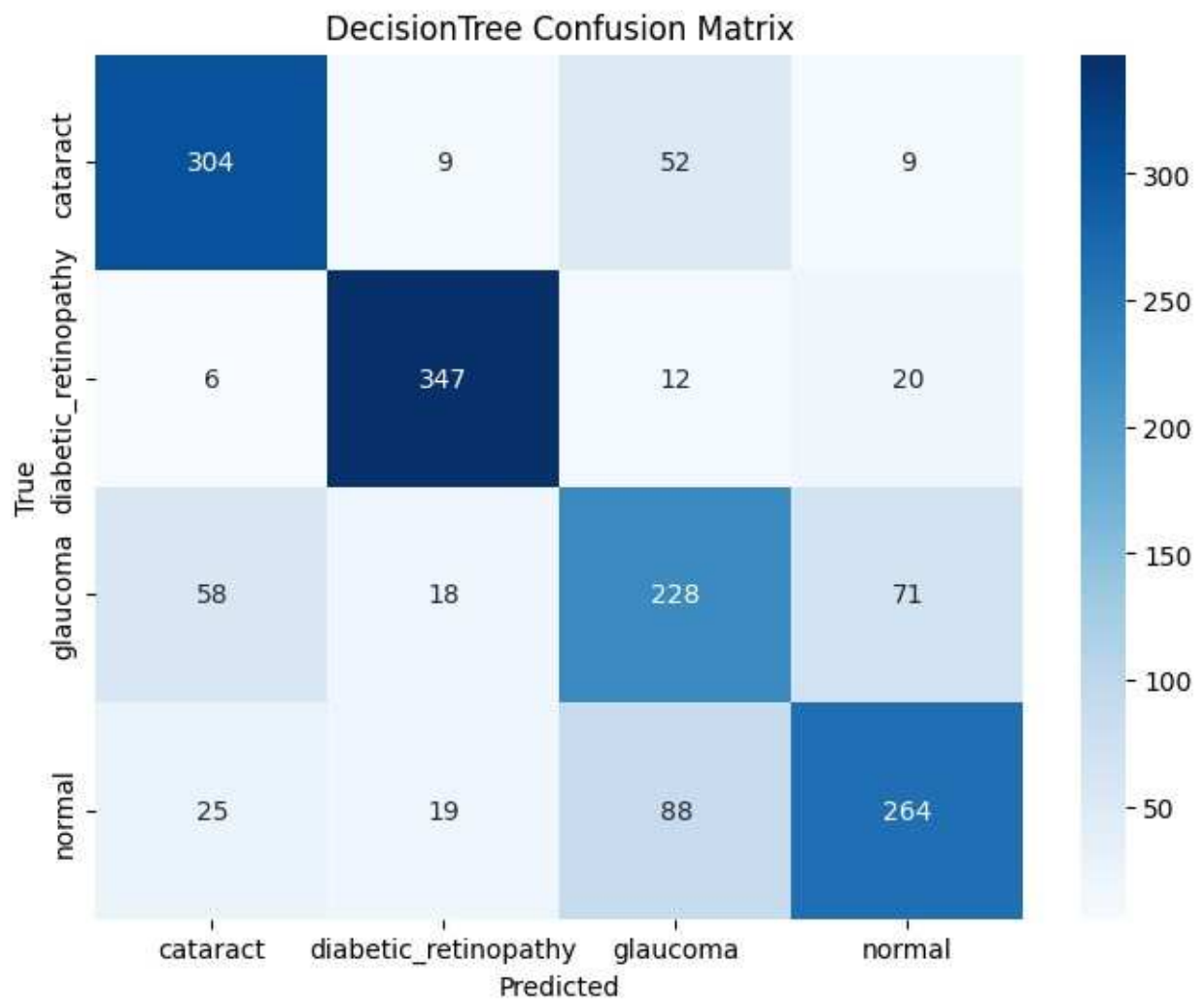
	precision	recall	f1-score	support
cataract	0.90	0.93	0.91	374
diabetic_retinopathy	0.98	0.92	0.95	385
glaucoma	0.82	0.75	0.78	375

normal	0.79	0.88	0.83	396
accuracy			0.87	1530
macro avg	0.87	0.87	0.87	1530
weighted avg	0.87	0.87	0.87	1530



Training DecisionTree...				
DecisionTree Test Accuracy: 0.7471				
	precision	recall	f1-score	support
cataract	0.77	0.81	0.79	374
diabetic_retinopathy	0.88	0.90	0.89	385
glaucoma	0.60	0.61	0.60	375
normal	0.73	0.67	0.69	396
accuracy			0.75	1530
macro avg	0.75	0.75	0.75	1530

weighted avg	0.75	0.75	0.75	1530
--------------	------	------	------	------



Training XGBoost...

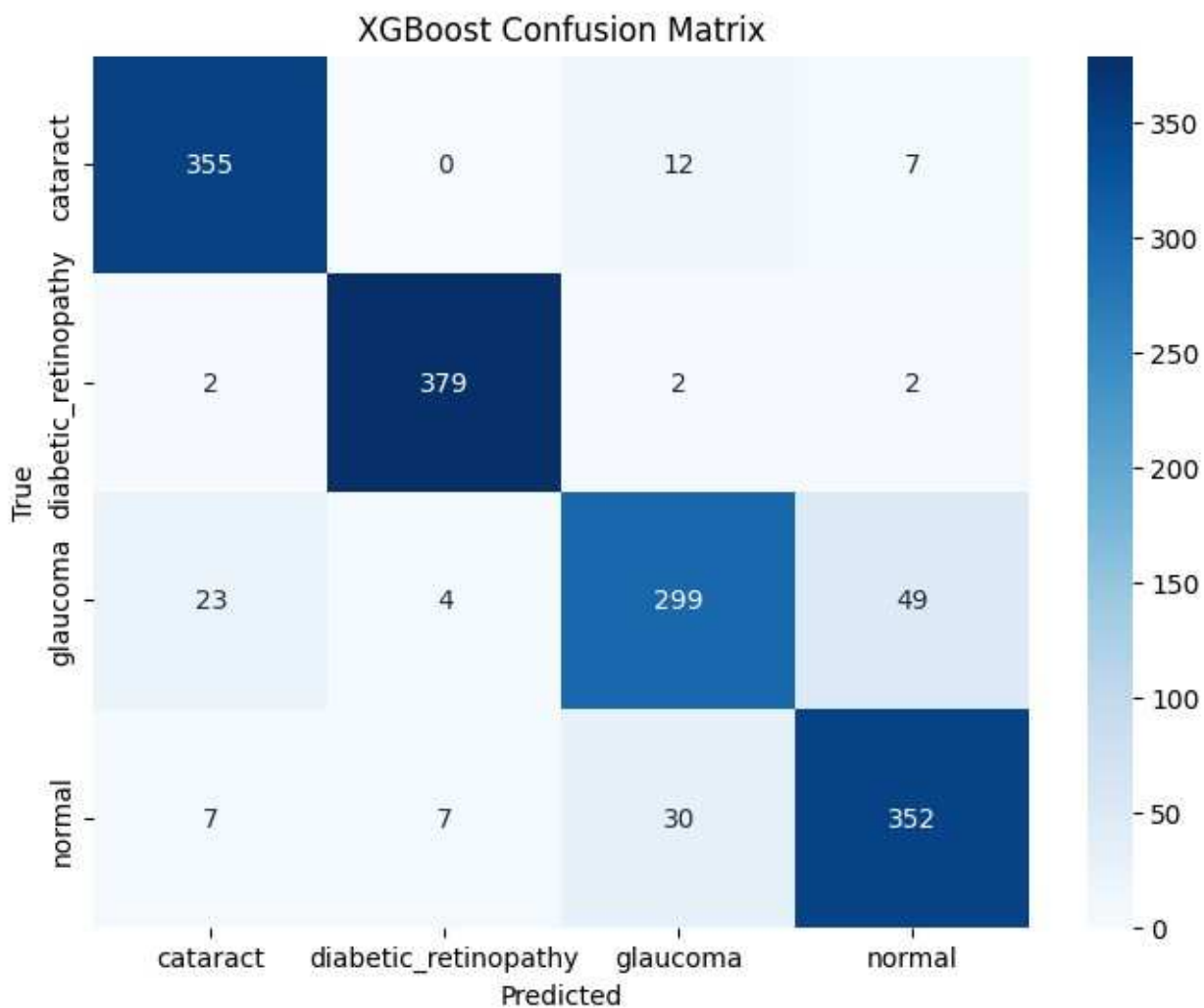
```
/usr/local/lib/python3.12/dist-packages/xgboost/training.py:199:  
UserWarning: [15:45:29] WARNING: /workspace/src/learner.cc:790:  
Parameters: { "use_label_encoder" } are not used.
```

```
bst.update(dtrain, iteration=i, fobj=obj)
```

XGBoost Test Accuracy: 0.9052

	precision	recall	f1-score	support
cataract	0.92	0.95	0.93	374
diabetic_retinopathy	0.97	0.98	0.98	385
glaucoma	0.87	0.80	0.83	375
normal	0.86	0.89	0.87	396

accuracy				0.91	1530
macro avg	0.90	0.90	0.90	0.90	1530
weighted avg	0.90	0.91	0.90	0.90	1530



Models performance summary:

RandomForest: Accuracy=0.8673, Precision=0.8702, Recall=0.8673, F1-score=0.8674

DecisionTree: Accuracy=0.7471, Precision=0.7460, Recall=0.7471, F1-score=0.7461

XGBoost: Accuracy=0.9052, Precision=0.9046, Recall=0.9052, F1-score=0.9044