🛰️ AI-Powered Land Use & Value Predictor - Task Breakdown

This document explains each section of the code notebook, clearly labeling the tasks involved in building the AI-powered land use and land value predictor.

🔧 Task 1: Install Dependencies

Installs the required Python libraries for deep learning, data processing, and visualization.

```
!pip install tensorflow tensorflow-datasets matplotlib numpy pandas
```

📥 Task 2: Load the EuroSAT Dataset

Loads the EuroSAT RGB satellite dataset with labeled land use types. Splits into training and test sets.

```
import tensorflow_datasets as tfds
import numpy as np
import tensorflow as tf

(ds_train, ds_test), ds_info = tfds.load(
    'eurosat/rgb',
    split=['train[:80%]', 'train[80%:]'],
    with_info=True,
    as_supervised=True
)

print(ds_info)
```

🖼️ Task 3: Visualize Sample Images

Displays a grid of sample images from the dataset, along with their land use labels (e.g., Industrial, Forest, Residential).

```
import matplotlib.pyplot as plt
```

```
class_names = ds_info.features['label'].names

plt.figure(figsize=(12,8))

# Take a batch of 9 images from the dataset
for image_batch, label_batch in ds_train.batch(9).take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        image = image_batch[i].numpy()
        label = label_batch[i].numpy()
        plt.imshow(image)
        plt.title(class_names[label])
        plt.axis("off")
```

🧹 Task 4: Preprocess the Data

Resizes and normalizes images for model input. Batches and optimizes data pipeline for training.

```
def preprocess(image, label):
    image = tf.image.resize(image, (64,64)) / 255.0
    return image, label

train_ds =
ds_train.map(preprocess).batch(32).prefetch(tf.data.AUTOTUNE)
test_ds =
ds_test.map(preprocess).batch(32).prefetch(tf.data.AUTOTUNE)
```

🧠 Task 5: Build CNN Model

Defines a Convolutional Neural Network to classify satellite images into 10 land use types.

```
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, 3, activation='relu',
input_shape=(64,64,3)),
    tf.keras.layers.MaxPooling2D(),
    tf.keras.layers.Conv2D(64, 3, activation='relu'),
    tf.keras.layers.MaxPooling2D(),
    tf.keras.layers.Conv2D(128, 3, activation='relu'),
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.summary()
```

🏋️ Task 6: Train the Model

Trains the model for 10 epochs and evaluates performance on the test set.

```
history = model.fit(
    train_ds,
    epochs=10,
    validation_data=test_ds
)
```

💰 Task 7: Simulate Land Value from Predicted Classes

Maps predicted land use types to simulated land values for interpretability.

```
land_value_map = {
    'AnnualCrop': 500,
    'Forest': 300,
    'HerbaceousVegetation': 350,
    'Highway': 800,
    'Industrial': 1000,
    'Pasture': 400,
    'PermanentCrop': 450,
    'Residential': 900,
    'River': 200,
    'SeaLake': 150
}

def predict_land_value(image_batch):
    preds = model.predict(image_batch)
    predicted_labels = np.argmax(preds, axis=1)
    predicted_classes = [class_names[i] for i in predicted_labels]
    predicted_values = [land_value_map[cls] for cls in
predicted_classes]
    return predicted_classes, predicted_values

# Test prediction on a batch
for images, labels in test_ds.take(1):
    classes, values = predict_land_value(images)
    for cls, val in zip(classes[:5], values[:5]):
        print(f"Land use: {cls}, Predicted land value: ${val}")
```

🎨 Task 8: Visualize Predictions with Labels and Prices

Plots predicted class and price for 9 test images to visually verify results.

```
plt.figure(figsize=(12,8))
for images, labels in test_ds.take(1):
    predicted_classes, predicted_values = predict_land_value(images)
    for i in range(9):
        ax = plt.subplot(3,3,i+1)
        plt.imshow(images[i].numpy())
        plt.title(f"{predicted_classes[i]}
(${predicted_values[i]})")
        plt.axis("off")
```

📦 Task 9: Install OpenCV and Gradio for Interface

Installs libraries needed for image annotation and building the user interface.

```
!pip install opencv-python
!pip install gradio
```

🖼️ Task 10: Define Prediction Function for Gradio

Defines how uploaded images are processed, classified, and annotated for display.

```
import cv2
from PIL import Image
import io

def predict_land_value_from_image(img):
    # Resize and normalize for model
    resized_img = tf.image.resize(img, (64, 64)) / 255.0
    input_tensor = tf.expand_dims(resized_img, axis=0)
    # Predict
    preds = model.predict(input_tensor)
    class_index = np.argmax(preds[0])
    confidence = preds[0][class_index] * 100
    land_use_type = class_names[class_index]
    land_value = land_value_map[land_use_type]

    # Annotate image with prediction
    display_img = np.array(img).astype(np.uint8).copy()
    display_img = cv2.resize(display_img, (256, 256))  # upscale for
better visibility

    label_text = f"{land_use_type} (${land_value}) -
{confidence:.1f}%"
    cv2.putText(display_img, label_text, (10, 30),
                cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 255, 255), 2,
cv2.LINE_AA)
    cv2.rectangle(display_img, (5, 5), (250, 40), (0, 0, 0),
thickness=-1)  # label background
    cv2.putText(display_img, label_text, (10, 30),
                cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 0), 2,
cv2.LINE_AA)

    annotated_img = Image.fromarray(display_img)

    return annotated_img, f"Land Use: {land_use_type}", f"Estimated
Value: ${land_value}", f"Confidence: {confidence:.1f}%"
```

🌐 Task 11: Launch Gradio Interface

Creates and launches a web-based interactive interface for real-time predictions.

```
import gradio as gr

gr.Interface(
    fn=predict_land_value_from_image,
    inputs=gr.Image(type="numpy", label="Upload Satellite Image"),
    outputs=[
        gr.Image(label="Annotated Image"),
        gr.Text(label="Land Type"),
        gr.Text(label="Estimated Land Value"),
        gr.Text(label="Confidence")
    ],
    title="🌍 Land Use & Value Predictor",
    description="Upload a satellite-style image to predict land use
type, estimated value, and confidence. Based on EuroSAT and
simulated pricing."
).launch(share=True)
```