Data Collection and Preprocessing Phase

Date	14 June 2025
Team ID	SWTID1749876754
Project Title	SynapseScan- AI Driven Classification of Ovarian Cancer Variants
Maximum Marks	6 Marks

Data Exploration and Preprocessing Report

Dataset variables will be statistically analyzed to identify patterns and outliers in ovarian cancer medical imaging data, with Python employed for preprocessing tasks like image normalization, data augmentation, and feature engineering. Data cleaning will address missing values and image quality issues, ensuring quality for subsequent analysis and modeling, and forming a strong foundation for accurate cancer variant classification and predictions.

Section	Description
Data Overview	Medical imaging dataset containing histopathological images of ovarian cancer variants Both training set and test set were divided into 5 subfolders: CC, EC, HGSC, LGSC, MC, which are the 5 main types of ovarian cancer. Each subfolder in training set had between 3000-12000 images. Each subfolder in test set had about 800 images.

Data Preprocessing Code Screenshots

Preprocessing Step	Implementation

Loading Data	datadownload.py
	`import os`
	`os.environ['KAGGLE_CONFIG_DIR'] = os.path.join(os.getcwd(), ".kaggle")`
	`import subprocess`
	`subprocess.run([`
	`"kaggle", "datasets", "download",`
	`"-d", "sunilthite/ovarian-cancer-classification-dataset",`
	`"--unzip"`
	`], check=True)`
	`print('Data downloaded successfully')`

Handling Missing Data	`# Oversampling to fix discrepancy between classes in train set`
	`ros = RandomOverSampler(random_state=42)`
	`X_resampled, y_resampled = ros.fit_resample(df[['file_path']], df['category_encoded'])`
	`df_resampled = pd.DataFrame(X_resampled, columns=['file_path'])`
	`df_resampled['category_encoded'] = y_resampled`

Data Transformation	`tr_gen = ImageDataGenerator(rescale=1./255)`
	`ts_gen = ImageDataGenerator(rescale=1./255)`
	
	`train_gen_new = tr_gen.flow_from_dataframe(`
	`train_df_new,`
	`x_col='file_path',`
	`y_col='category_encoded',`
	`target_size=img_size,`
	`class_mode='sparse',`
	`color_mode='rgb',`
	`shuffle=True,`
	`batch_size=batch_size`
	`)`

```python
df = pd.DataFrame(data, columns=['file_path', 'label'])
label_encoder = LabelEncoder()
df['category_encoded'] = label_encoder.fit_transform(df['label'])
df = df[['file_path', 'category_encoded']]

# Oversampling to fix discrepancy between classes in train set
ros = RandomOverSampler(random_state=42)
X_resampled, y_resampled = ros.fit_resample(df[['file_path']], df['category_encoded'])
df_resampled = pd.DataFrame(X_resampled, columns=['file_path'])
df_resampled['category_encoded'] = y_resampled

# Split dataset
df_resampled['category_encoded'] = df_resampled['category_encoded'].astype(str)

train_df_new, temp_df_new = train_test_split(
    df_resampled,
    train_size=0.8,
    shuffle=True,
    random_state=42,
    stratify=df_resampled['category_encoded']
)

valid_df_new, test_df_new = train_test_split(
    temp_df_new,
    test_size=0.5,
    shuffle=True,
    random_state=42,
    stratify=temp_df_new['category_encoded']
)
```

Feature Engineering

```python
def create_inception_model(input_shape):
    inputs = Input(shape=input_shape)
    base_model = InceptionV3(weights='imagenet', input_tensor=inputs, include_top=False)
    for layer in base_model.layers:
        layer.trainable = False

    x = base_model.output
    height, width, channels = 5, 5, 2048
    x = Reshape((height * width, channels))(x)
    attention_output = DifferentialAttention(num_heads=8, key_dim=channels)(x)
    attention_output = Reshape((height, width, channels))(attention_output)
```

Save Processed Data

```python
dataset_path = "Train_Images"
data = []

for label in os.listdir(dataset_path):
    sub_dir = os.path.join(dataset_path, label)
    if os.path.isdir(sub_dir):
        for file_name in os.listdir(sub_dir):
            file_path = os.path.join(sub_dir, file_name)
            data.append([file_path, label])
```

```python
    cnn_model.save(model_path)
    print(f"Model saved to {model_path}")
```