capstone project Data Preparation, Feature Engineering, and Model Exploration assignment

Project Title: Medical Diagnostic System for Breast Cancer Detection

Team Members:
1. Mohammed Adil
2. Ali Soltan Dileyta
3. Kena Teshome
4. Dagim Faji
5. Mahdi Abdi Rayaleh
6. Impundu Joyeux Thevenin

1. Data Preparation/Feature Engineering

1.1 Overview

Data preparation and feature engineering are critical phases in any machine learning project. These steps involve collecting, cleaning, transforming, and preparing the data to ensure that the model has the best possible information to learn from. Proper data preparation can significantly impact the accuracy and efficiency of the model, making it crucial for the overall success of the project.

1.2 Data Collection

- **Source of the Dataset**: The dataset used in this project comprises breast cancer images categorized into benign and malignant classes. The data was sourced from a Kaggle repository that provides labeled images for the training of machine learning models.
- **Preprocessing Steps**:
  - The images were collected and stored in separate directories for each class.
  - Images were resized to a consistent dimension of 150x150 pixels to standardize input size.
  - The images were converted to NumPy arrays and normalized by dividing pixel values by 255.0.

1.3 Data Cleaning

- **Handling Missing Values**: Since the dataset consisted of image files, there were no missing values in the traditional sense. However, care was taken to ensure that all images were readable and of the correct format.
- **Outliers**: Outliers in image data might include images that are mislabeled or contain irrelevant content. A visual inspection of the dataset was conducted to remove any such images.

- **Data Quality Issues**: Any corrupted images or those that did not meet the required dimensions were excluded from the dataset.

1.4 Exploratory Data Analysis (EDA)

- **Visualization**:
  - Distribution of classes (benign vs. malignant) was visualized to assess class imbalance.
  - Sample images from each class were displayed to understand the visual differences between benign and malignant cases.
  - Techniques like histogram equalization were considered to enhance image contrast, but ultimately, the decision was made to use data augmentation techniques to enhance the dataset.
- **Key Insights**:
  - The dataset showed a slight imbalance between the benign and malignant classes.
  - Malignant images typically showed more irregular patterns and structures compared to benign images.

1.5 Feature Engineering

- **New Features**:
  - Since the dataset consists of images, feature engineering was focused on extracting and enhancing visual features.
  - Data augmentation techniques were applied to artificially increase the diversity of the training data. This included:
    - Random rotations
    - Zoom transformations
    - Horizontal and vertical flips
    - Width and height shifts
- **Rationale**: The augmentation was aimed at making the model more robust to variations in image orientation, scale, and translation, thus improving its generalization capability.

1.6 Data Transformation

- **Scaling**: Image data was normalized by scaling pixel values to the range [0, 1].
- **Encoding**: The target labels (benign and malignant) were one-hot encoded to facilitate multi-class classification.
- **Code Snippet**:

```
from keras.preprocessing.image import ImageDataGenerator

# Data Augmentation
datagen = ImageDataGenerator(
    rescale=1.0/255,
    shear_range=0.2,
    zoom_range=0.2,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

datagen.fit(x_train)
```

2. Model Exploration

2.1 Model Selection

- **Rationale**: The VGG16 model was chosen for this project due to its proven effectiveness in image classification tasks. VGG16 is a deep convolutional neural network that has been widely used in computer vision tasks and is known for its ability to capture complex visual patterns.
- **Strengths**:
  - **Pre-trained Weights**: The use of pre-trained weights on the ImageNet dataset allows the model to leverage pre-learned features, reducing the need for extensive training on the new dataset.
  - **Depth**: VGG16's deep architecture (16 layers) enables it to capture intricate details in images, which is critical for distinguishing between benign and malignant cases.
- **Weaknesses**:
  - **Computational Cost**: VGG16 is computationally expensive, requiring significant processing power and memory.
  - **Risk of Overfitting**: Given the model's complexity, there is a risk of overfitting, especially if the training data is limited.

2.2 Model Training

- **Hyperparameters**:
  - **Optimizer**: Adam optimizer was used with a learning rate of 0.0001.
  - **Batch Size**: 32
  - **Epochs**: 20 epochs were used to ensure the model had enough iterations to learn effectively.
  - **Class Weights**: Class weights were computed and applied to handle the slight imbalance in the dataset.
- **Cross-Validation**: K-fold cross-validation was considered, but due to the computational cost, a simple train/test split was used for model evaluation.

2.3 Model Evaluation

- **Metrics**:

    - **Accuracy**: The primary metric used to assess model performance.
    - **Confusion Matrix**: To visualize the performance of the classifier across the two classes.
    - **ROC Curve**: To evaluate the model's ability to discriminate between the positive and negative classes.
    - **Precision and Recall**: To assess the model's performance in predicting each class.

- **Code Snippet for Evaluation**:

```
# Evaluate the model
results = model.evaluate(x_test, y_test, batch_size=150)
print("Validation loss, accuracy:", results)

# Confusion Matrix
from sklearn.metrics import confusion_matrix
import seaborn as sns

y_pred = model.predict(x_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = np.argmax(y_test, axis=1)

conf_matrix = confusion_matrix(y_true, y_pred_classes)
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

2.4 Code Implementation

- **Data Preparation**:
    - Loading and preprocessing images, applying data augmentation, and splitting data into training and testing sets.
- **Model Training**:
    - Transfer learning with VGG16, followed by training with additional convolutional and dense layers.
- **Model Evaluation**:
    - Evaluating the model using accuracy, confusion matrix, and ROC curve.