Establishing machine learning models to predict the early risk of gastric cancer based on lifestyle factors

Mohan Prasath S
CH.EN.U4CSE22137

Exploring Research Gaps in ML-Based Gastric Cancer Prediction:

1. **Limited Integration of Multi-type Data:**
   Currently existing models typically uses only a single data modality (eg., imaging or clinical data). However, multi-type data can be used to obtain more insights and lead to more accurate predictions. Future research should explore the integration of diverse data types to enhance predictive accuracy

2. **Lack of Real-world Applicability:**
   Many models are tested with only controlled and limited datasets, lacking generalization to real world (hospitals, clinics stc). In order to perform well in the real world, the model must be trained with more diverse dataset. Future studies should validate models on larger, more diverse patient populations to ensure clinical relevance.

3. **Limited Integration of Multimodal:**
   Current researches mainly use a single model to predict the gastric cancer. However, multiple different models can be used where each model can process and handle a specific data type and output from the models can be combined at the end to get more accurate predictions.

4. **Insufficient Focus on Early Detection:**
   Many currently existing models only focuses on detection at the advanced stage and it fails to make the predictions in the early stage itself. Research should prioritize early detection techniques to improve patient outcomes and survival rates.

5. **Need for Personalized Approaches in Patient Care**:
   Most of the currently available models fail to account for various individual patient-specific features, for example, genetics, lifestyle choice, or other health problems. All these might considerably influence gastric cancer risk and progression. Thus, future research should develop personalized predictive models considering all or any of the abovementioned differences. This is the only way by which prevention and treatment approaches will be delicately fashioned, and the prognosis of such patients may be improved.

Dataset:

in the current study two different data sets are used:

1. GasHisDB:
   It is a publicly available image data set and contains 245,196 tissue case images. Specifically, two types of data are included: abnormal and normal. Normal category contains all the images that are obtained from normal healthy people and abnormal category contains all the images that are obtained from the gastric cancer affected patients. These images are microscopic stomach gut tissue images of various peoples.

2. Synthetic dataset:
   A dataset that is created from the information obtained from a research paper about gastric cancer. Data was derived from the research paper's tabular representation of gastric cancer patients and their characteristics. In total, the dataset involved 2029 people: 429 patients with gastric cancer and 1780 healthy individuals as a comparison. The dataset included information about 28 different features (like age, lifestyle factors, etc.) for each person and one key outcome feature showing their risk of developing gastric cancer.

Combined this both datasets by assigning a random image for the people in synthetic dataset from GasHisDB. For healthy person a random normal image is selected and assigned to the corresponding person, for abnormal person a random abnormal is selected and assigned to the corresponding abnormal person. By this way a more comprehensive dataset is developed where each person has both image and life-style factors as their features.

Data preprocessing:

1. **Image data preprocessing:**
   - All the images are resized to 224 x 224 pixels to fed into the model.
   - Convert the images into RBG format.
   - Normalization is done by adjusting the brightness and contrast of the images to a standard level.
   - Converted images to **tensors** to prepare them for processing in PyTorch.

2. **Synthetic data preprocessing:**
   - Labelled encoding is done to convert all the categorical values into numbers so that machine learning models can understand them. For example, "yes" become 1 and "no" becomes 0.
   - Only important features that are very much crucial for the presence of gastric cancer is selected and the model is trained with these features.
   - Removed the entries where are some missing values in the row.

- The target variable (CancerStatus) is mapped from categorical labels ("With Gastric Cancer" and "Without Gastric Cancer") to binary numeric values: With Gastric Cancer → 1 and Without Gastric Cancer → 0
- The features (X) and target (y) are separated: Features include everything except CancerStatus and Patient_id. The target (y) is CancerStatus.

Proposed algorithm:

Combining XGBoost with Deep Learning for Enhanced Gastric Cancer Detection.
Most previous studies focus on using either machine learning techniques (like XGBoost or SVM) or deep learning approaches, but few have explored combining them. Limited studies combine image data (e.g., gut images) with non-image data (e.g., symptoms, lifestyle factors).

Used a convolutional neural network (CNN) based ResNet-50 model to process stomach images to detect cancerous patterns and used a traditional machine learning classifier XGBoost to process the structured data.
The outputs from the both models are combined, by this the proposed model provides a more accurate and holistic predictions. The merging of two outputs is done by stacking.
Stacking:
Train a meta-learner model (logistic regression) that takes the predictions from multiple base models (like ResNet and XGBoost) as input features to make the final prediction. This model learns from the input and provides the final outputs accordingly.

The main reason behind combining image and non-image dataset is to make early detections. Sometimes early signs are not visible in the early imaging. So, in this case non image data can be used for that particular patient as early indicators. Also, this dataset can be used as complementary to each other. If the image data is unclear for some people, then non-image data can be used to make predictions. If non-image data is missing, then image data can be used to make predictions.

Steps involved in the proposed model:

1. **Data Loading and Preprocessing:**
   - CSV data loading: Training and testing datasets were loaded from CSV files containing patient details such as Weight_loss, Smoking_status, and CancerStatus.
   - Label Encoding: The target variable CancerStatus was encoded as binary values, where 'With Gastric Cancer' was mapped to 1 and 'Without Gastric Cancer' to 0. Additionally, categorical columns representing lifestyle factors and health conditions (e.g., Smoking_status, Weight_loss) were label-encoded into numerical format for model compatibility.

```
# Load the CSV data
train_csv_path = r'D:\resources\ML\final_data\train\csv\train_data.csv'
test_csv_path = r'D:\resources\ML\final_data\test\csv\test_data.csv'

train_df = pd.read_csv(train_csv_path)
test_df = pd.read_csv(test_csv_path)

# Encode 'CancerStatus' with custom mapping
train_df['CancerStatus'] = train_df['CancerStatus'].map({'With Gastric Cancer': 1, 'Without Gastric Cancer': 0})
test_df['CancerStatus'] = test_df['CancerStatus'].map({'With Gastric Cancer': 1, 'Without Gastric Cancer': 0})

# Encode Yes/No columns
label_enc = LabelEncoder()
yes_no_columns = ['Weight_loss', 'Smoking_status', 'High_fat_foods_status',
                  'Stomach_ulcers', 'Chronic_atrophic_gastritis',
                  'High_salt_intake', 'Helicobacter_pylori_infection']

for col in yes_no_columns:
    train_df[col] = label_enc.fit_transform(train_df[col])
    test_df[col] = label_enc.transform(test_df[col])
```

2. **Feature and Target Separation:**
   - Features (x) was obtained by removing columns CancerStatus and Patient_id of the dataset. The target variable (y) consisted of the encoded values from the CancerStatus column.

```
# Separate features (X) and target (y)
X_train_csv = train_df.drop(columns=['CancerStatus', 'Patient_id'])
y_train_csv = train_df['CancerStatus']
X_test_csv = test_df.drop(columns=['CancerStatus', 'Patient_id'])
y_test_csv = test_df['CancerStatus']
```

3. **Image Dataset Construction:**
   - a custom dataset class was created to image data, where the patients gut images were paired with their respective labels (CancerStatus) based in patient_id. Images were preprocessed by resizing, random rotations and normalization is done to match the requirements of the pre-trained ResNet-50 model.

```python
class GutImagesDataset(Dataset):
    def __init__(self, df, img_folder, transform=None):
        self.df = df
        self.img_folder = img_folder
        self.transform = transform

    def __len__(self):
        return len(self.df)

    def __getitem__(self, idx):
        img_name = os.path.join(self.img_folder, f"patient_{self.df.iloc[idx]['Patient_id']}.jpg")
        image = Image.open(img_name).convert('RGB')
        if self.transform:
            image = self.transform(image)
        label = self.df.iloc[idx]['CancerStatus']
        return image, label
```

```python
# Transformations for image preprocessing with data augmentation
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.RandomHorizontalFlip(),  # Data augmentation: random horizontal flip
    transforms.RandomRotation(10),  # Data augmentation: random rotation
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])
```

4. ResNet50 for Image Classification:

- Image classification was achieved by using the pre-trained CNN ResNet50. The final fully connected layer was modified to output two classes to align with the binary classification task.

```python
class ResNet50(nn.Module):
    def __init__(self):
        super(ResNet50, self).__init__()
        self.resnet = models.resnet50(pretrained=True)
        self.resnet.fc = nn.Linear(self.resnet.fc.in_features, 2)  # Binary classification

    def forward(self, x):
        return self.resnet(x)
```

5. Training the ResNet50 Model:

- The model was trained by using Adam optimizer and cross-entropy loss for 20 epochs. Finally, at each epoch, it passed over the batches of images, calculated predictions, and updated weights according to the loss function; it did this until convergence.

```python
# Training loop for ResNet50
num_epochs = 20  # Increase the number of epochs for better training
for epoch in range(num_epochs):
    resnet_model.train()
    running_loss = 0.0
    for images, labels in train_loader:
        images, labels = images.to(device), labels.to(device)
        optimizer.zero_grad()
        outputs = resnet_model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item()

    print(f"Epoch [{epoch+1}/{num_epochs}], Loss: {running_loss/len(train_loader):.4f}")
```

```
Epoch [1/20], Loss: 0.8190
Epoch [2/20], Loss: 0.2061
Epoch [3/20], Loss: 0.2227
Epoch [4/20], Loss: 0.1572
Epoch [5/20], Loss: 0.1704
Epoch [6/20], Loss: 0.1513
Epoch [7/20], Loss: 0.0874
Epoch [8/20], Loss: 0.0978
Epoch [9/20], Loss: 0.1231
Epoch [10/20], Loss: 0.0831
Epoch [11/20], Loss: 0.0941
Epoch [12/20], Loss: 0.1195
Epoch [13/20], Loss: 0.1188
Epoch [14/20], Loss: 0.0894
Epoch [15/20], Loss: 0.0763
Epoch [16/20], Loss: 0.0617
Epoch [17/20], Loss: 0.0487
Epoch [18/20], Loss: 0.0683
Epoch [19/20], Loss: 0.0526
Epoch [20/20], Loss: 0.0611
```

6. XGBoost for Tabular Data:

The XGBoost classifier was employed to analyse the tabular dataset, excluding image data, with the objective of predicting the target variable based on patient attributes, including weight loss, smoking behaviours, and infection status. The model was set up with suitable evaluation metrics, specifically mlogloss, to accommodate the multi-class characteristics of the algorithm, despite its primary design being intended for binary classification.

```python
# XGBoost model training
xgb_model = xgb.XGBClassifier(use_label_encoder=False, eval_metric='mlogloss')
xgb_model.fit(X_train_csv, y_train_csv)

# XGBoost predictions
y_pred_csv = xgb_model.predict(X_test_csv)
```

7. Stacking: Combining ResNet50 and XGBoost Predictions:

- Predictions from the ResNet50 model and the XGBoost model were generated for the test set. The predictions from both models were combined into a new feature matrix. This served as input for a meta-learner (logistic regression) and uses both image and non-image data for final predictions.

```python
# Stacking: Combine ResNet50 and XGBoost predictions
X_stack_train = np.column_stack((resnet_predictions, y_pred_csv))  # Stacking features
y_stack_train = y_test_csv  # Ground truth labels

# Split stacking dataset for training meta-
X_train_stack, X_val_stack, y_train_stack, y_val_stack = train_test_split(X_stack_train, y_stack_train, test_size=0.2, random_state=42)
```

(variable) y_val_stack: Any

8. Meta-Learner: Logistic Regression:

- used logistic regression model to combine the predictions from two other models, ResNet50 and XGBoost. By splitting the dataset into training and testing sets, logistic regression model was trained to make the final predictions based on the combined outputs of the other two models.

```
# Meta-learner predictions
y_pred_meta = meta_learner.predict(X_val_stack)

# Evaluate meta-learner
meta_accuracy = accuracy_score(y_val_stack, y_pred_meta)
print(f"Meta-Learner (Stacking) Accuracy: {meta_accuracy * 100:.2f}%")

# Final predictions using stacking
final_predictions = meta_learner.predict(X_stack_train)
```

9. Evaluation of the Stacking Model.

The performance of the meta-learner was tested on the validation set using accuracy, classification report precision, recall, F1-score, and AUC-ROC. This process ensured that the overall model was adequately able to utilize both image-based and tabular data in enhanced gastric cancer detection.

```
# Final evaluation
combined_accuracy = accuracy_score(y_stack_train, final_predictions)
print(f"Stacking Model Accuracy: {combined_accuracy * 100:.2f}%")

# Additional metrics
print(classification_report(y_stack_train, final_predictions))
print(f"Stacking AUC-ROC: {roc_auc_score(y_stack_train, final_predictions):.4f}")
```

10. Final output:

```
Stacking Model Accuracy: 92.53%
              precision    recall  f1-score   support

           0       0.94      0.91      0.93       234
           1       0.91      0.94      0.92       208

    accuracy                           0.93       442
   macro avg       0.92      0.93      0.93       442
weighted avg       0.93      0.93      0.93       442

Stacking AUC-ROC: 0.9260
```

Architecture diagram: