CSC 590/599 ─ Graduate Project/Thesis (S25)
Project Progress Report

****

Project Title: Medical Text Classification using LLMs
Student Name: Medha Maisa

****

1. Description of Completed Work (Itemized)

Below is an itemized summary of the work completed to date for the project *"Medical Text Classification using LLMs."* The work is divided into two main phases corresponding to the two datasets explored: **TCGA Pathology Reports** and **Independent Medical Reviews (IMR)**.

1.1 Work Completed on TCGA Pathology Reports Dataset

- **Dataset Acquisition & Loading**
    - Successfully downloaded and loaded the TCGA Pathology Reports dataset from Kaggle.
    - Focused on key fields: text (pathology content) and patient_filename (identifier).
- **Initial Data Exploration & Cleaning**
    - Analyzed text lengths, checked for nulls, and removed duplicate or extremely short records.
    - Applied normalization (lowercasing, punctuation removal, etc.) to standardize the text.
- **Label Generation for Binary Classification**
    - Assigned binary labels: 1 if the word *"cancer"* appeared in the report text, 0 otherwise.
    - Evaluated class balance and ensured label reliability.
- **Baseline Model (TF-IDF + Logistic Regression)**
    - Built a machine learning pipeline using TF-IDF for feature extraction and Logistic Regression for classification.
    - Incorporated class weighting and stratified train-test split for fair evaluation.
    - Evaluated performance using accuracy, precision, recall, and F1-score.
- **Multi-Class Classification (Preliminary Exploration)**
    - Began examining cancer subtypes (e.g., BRCA, COAD) for potential multi-class tasks.
    - Noted significant class imbalance issues; currently identifying the most frequent subtypes for focused modeling.
- **Visualization and Reporting**
    - Created visual aids including class distribution charts, word clouds, and text length histograms.
    - Documented all work using Jupyter Notebooks for reproducibility and clarity.

Note: *Work on the TCGA dataset is currently paused as I explore an alternative dataset for potentially richer classification tasks.*

1.2 Preliminary Work on Independent Medical Reviews (IMR) Dataset

- **Dataset Loading & Inspection**
  - Loaded the *Independent Medical Reviews* dataset (~19,245 records, 11 columns) from a local source.
- **Initial Data Exploration**
  - Verified dataset shape and schema; identified missing values, especially in sub-category columns.
  - Computed word-level statistics and generated summary statistics on the Findings text length.
- **Preprocessing Pipeline**
  - Developed a preprocessing function to clean the review text (lowercasing, removing numbers, punctuation, extra spaces).
  - Removed duplicate entries and very short texts (less than 10 characters) to ensure data quality.
  - Resulted in a cleaner, leaner dataset for future classification tasks.

2. Remaining Work and Completion Plan

The following tasks are required to complete the project. These tasks are structured around key milestones to ensure timely and focused progress:

2.1 Dataset Refinement and Labeling

For TCGA Dataset:
• Conduct further exploratory data analysis (EDA) to detect potential biases, inconsistencies, and outliers in the dataset.
• Implement negation handling and context-based labeling to enhance the accuracy of cancer classification in pathology reports.
• Apply oversampling or undersampling techniques (e.g., SMOTE or random undersampling) to balance the classes.
• Validate and refine the binary classification labels based on improved contextual understanding.

For IMR Dataset:
• Conduct data labeling to ensure high-quality labels.
• Identify and handle any mislabeled or inconsistent samples in the dataset.
• Explore the possibility of shifting from binary to multi-class classification if the data supports meaningful label segmentation.

2.2 Rebuilding the Baseline Model
• Rebuild and retrain the baseline model(s) on the refined and accurately labeled data.
• Explore traditional ML algorithms such as Logistic Regression, Random Forest and SVM for both TCGA and IMR datasets.
• Evaluate all models using standard classification metrics: accuracy, F1-score, precision, and recall.
• Select the best-performing model per dataset as the final baseline and document reproducibility details. Explore the possibility of training a baseline model on both datasets.

2.3 Building the LLM-Based Classifier
• Investigate and evaluate approaches for applying Large Language Models (LLMs) to medical

text classification.
• Explore architectures such as BERT, BioBERT, and GPT variants.
• Ensure proper dataset formatting for compatibility with LLMs, especially for prompt-based and input-segmented classification.
• Finalize the best LLM-based approach for each dataset after comparative evaluation.

2.4 Few-Shot Learning Experimentation
• Investigate the potential of few-shot learning with LLMs for both datasets.
• Design and experiment with different prompt templates and input formats.
• Compare the few-shot model results with both the baseline and fine-tuned LLM models to assess effectiveness.

2.5 Analysis and Visualization
• Create comparative charts and tables for all models across both datasets.
• Use confusion matrices, ROC curves, bar graphs, and other visual tools to interpret model performance.
• Analyze misclassifications and edge cases to uncover deeper insights and suggest improvements.

3. Appendix – Source Code

The full source code for data preprocessing, model training, evaluation, and visualizations for **TCGA dataset** and The **Preliminary Work on Independent Medical Reviews (IMR) Dataset** is included in this section. The code is well-organized and properly documented to ensure reproducibility. Specific code snippets and references are provided below:

Appendix A.1: Model building using TCGA Dataset

```
# Load dataset
file_path = "path_to_the_file"
df = pd.read_csv(file_path)

# Check for missing values
print("\nChecking for missing values:")
print(df.isnull().sum())
```

```
# Remove missing and duplicate values
df.dropna(subset=["text"], inplace=True)
df.drop_duplicates(subset=["text"], inplace=True)

# ==============================
# 2. Define Labeling Functions
# ==============================

def binary_label(text):
    """Label data for binary classification: Cancer vs Non-Cancer."""
    cancer_keywords = ["cancer", "tumor", "carcinoma", "malignant", "neoplasm"]
    return "Cancer" if any(word in text.lower() for word in cancer_keywords) else "Non-Cancer"

def multiclass_label(text):
    """Label data for multi-class classification."""
    text = text.lower()
    if any(word in text for word in ["malignant", "carcinoma", "neoplasm", "cancer"]):
        return "Cancer"
    elif any(word in text for word in ["benign", "non-cancerous", "harmless"]):
        return "Benign"
    elif any(word in text for word in ["precancerous", "dysplasia"]):
        return "Pre-cancerous"
    elif any(word in text for word in ["normal", "no abnormality", "clear"]):
        return "Normal"
    else:
        return "Other"

# Apply binary classification label
df["binary_label"] = df["text"].apply(binary_label)

# Apply multi-class classification label
df["multiclass_label"] = df["text"].apply(multiclass_label)
```

```python
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import GridSearchCV

# Defining the Model
model = LogisticRegression(max_iter=1000, class_weight="balanced")

# Hyperparameter Tuning using GridSearchCV
param_grid = {
    'C': [0.1, 1, 10],
    'solver': ['liblinear', 'saga']
}
grid_search = GridSearchCV(model, param_grid, cv=5, scoring="accuracy")
grid_search.fit(X_train_resampled, y_train_resampled)

# Get Best Model from GridSearch
best_model = grid_search.best_estimator_

# Make Predictions
y_pred = best_model.predict(X_test_tfidf)

# Model Evaluation
print("Accuracy Score:", accuracy_score(y_test, y_pred))
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

Appendix A.3: Preliminary Work on Independent Medical Reviews (IMR) Dataset.

```python
import pandas as pd
import numpy as np
import re
import string
import matplotlib.pyplot as plt
import seaborn as sns

# Step 1: Load and Analyze Dataset
file_path =
"C:/Users/Medha/Documents/CSUDH/Spring2025/Independent_Medical_Reviews.csv"  #
Update this with the correct file path
df = pd.read_csv(file_path)

# Display dataset info
print("Dataset Shape:", df.shape)
#print("First 5 rows:\n", df.head())
print("Columns:", df.columns)

print("\n Is NULL:",df.isnull().sum())
#print('\nDiagnosis Category:\n',df['Diagnosis Category'].value_counts())
print(f"\nTotal records: {df.shape[0]}")
df['text_length'] = df['Findings'].apply(lambda x: len(str(x).split()))  # For Kaggle
print('\ntext_length\n',df['text_length'].describe())

'''print("\nCOUNTING UPHELD AND OVERTURNED DECISIONS\n")
decision_counts = df['Determination'].value_counts()
```

```
# Display the counts
print(decision_counts)'''

# Check for missing values
print("\nMissing values per column:\n", df.isnull().sum())

# Step 2: Preprocessing
def clean_text(text):
    if pd.isnull(text):
        return ""  # Handle NaN values
    text = text.lower()  # Convert to lowercase
    text = re.sub(r'\d+', '', text)  # Remove numbers
    text = text.translate(str.maketrans("", "", string.punctuation))  # Remove punctuation
    text = re.sub(r'\s+', ' ', text).strip()  # Remove extra spaces
    return text

df["Cleaned_Text"] = df["Findings"].apply(clean_text)  # Assuming "Findings" contains the
medical reviews

# Remove duplicates and short texts
df.drop_duplicates(subset=["Cleaned_Text"], inplace=True)
df = df[df["Cleaned_Text"].str.len() > 10]

print(f"Dataset after preprocessing: {df.shape}")
```

Appendix A.4: Sample Findings from the Independent Medical Reviews (IMR) Dataset.

```
Dataset Shape: (19245, 11)

Columns: Index(['Reference ID', 'Report Year', 'Diagnosis Category',

       'Diagnosis Sub Category', 'Treatment Category',

       'Treatment Sub Category', 'Determination', 'Type', 'Age Range',

       'Patient Gender', 'Findings'],

      dtype='object')


 Is NULL: Reference ID                 0

Report Year                  0

Diagnosis Category          59

Diagnosis Sub Category    1904

Treatment Category         450

Treatment Sub Category    1268

Determination                0

Type                         0
```

```
Age Range           1210
Patient Gender      1210
Findings            20
dtype: int64

Total records: 19245
Dataset after preprocessing: (19130, 13)
```