**AIDI 2001**

**Final Project Report**

**Medical Assistant – Disease Text Classification**

**GROUP #32**

**Yuming Yao 100904611**

**Binu Das 100894927  
Vishal Kochuparampu Roy 100900622**

1. **Problem Identification:**

There are two problems our application aims to solve. First, there is the time window between a patient first experience symptoms and the moment the patient received diagnosis and medical treatment from professionals. Timely disease diagnosis is crucial for medical interventions, which can significantly impact patient outcomes. If remote diagnosis is available, patient with life-threatening symptoms can take actions by themselves before they received emergency care.

Secondly there are many situations that in-person consultations are not possible or desirable. Patients will do self-diagnosis in those cases. However, patients with limited medical knowledge are very likely to make false diagnosis on their symptoms. If they take medicines or treatment from based on false self-diagnosis, their health condition might be harmed further.

Importance of the Problem:

1. Early Intervention: Timely and accurate disease diagnosis leads to early intervention and treatment, which can greatly improve patient prognosis and quality of life. By identifying potential diseases before they get access to professional treatment, patients can take immediate actions by themselves.
2. Resource Optimization: remote disease prediction can help patients decide whether to seek diagnosis and treatment from healthcare providers. It can be used to efficiently allocate resources.
3. Self-diagnosis Accuracy: Delayed or incorrect diagnoses can result in worsening health conditions and increased healthcare costs.
4. Reducing Workload: The healthcare industry often faces high patient loads and time constraints for doctors. AI can help streamline the diagnostic process by quickly analyzing symptom descriptions, potentially reducing the workload on healthcare professionals.

AI's Role in Addressing the Problem:

We implement artificial intelligence technology to address these two problems. Our application allows users to do self-diagnosis before they can get access to medical support, or when in-person diagnosis is not available or desired. AI, specifically natural language processing (NLP) and machine learning (ML) techniques, can significantly benefit disease prediction based on symptom descriptions:

1. Pattern Recognition: AI algorithms can identify subtle patterns and correlations in large datasets of symptom descriptions and disease labels. This can lead to more accurate and nuanced disease predictions.
2. Multi-Symptom Analysis: AI can efficiently analyze multiple symptoms in parallel and consider their complex interactions, which may be challenging for human diagnosticians.
3. Continuous Learning: AI models can continuously learn and adapt from new medical research and patient data, ensuring that predictions remain up-to-date and relevant.
4. Efficiency and Scale: AI can process a large number of symptom descriptions in real-time, making it practical for applications in telemedicine, healthcare chatbots, and clinics with high patient volumes.

By leveraging AI's ability to analyze and understand natural language, our AI product aims to revolutionize disease prediction by delivering more accurate and timely diagnoses. It will ultimately improve patient’s well-being and assist healthcare system.

1. **Language Model Selection**

For the medical assistant app aimed at disease prediction based on symptom descriptions, we chose to use the "distilbert-base-uncased" model from the Hugging Face Transformers library. This choice is based on the following justifications and the characteristics of the selected model:

1. Pre-trained Model: The "distilbert-base-uncased" model is a pre-trained language model that has been fine-tuned on a large corpus of text data. Pre-trained models capture rich semantic and syntactic information from a wide range of text sources, making them suitable for understanding and analyzing symptom descriptions.
2. Distilled Version: "Distillation" refers to a process where a larger and more complex model is distilled into a smaller and more efficient version while retaining most of its performance. The "distilbert-base-uncased" model is a distilled version of the original BERT model. It offers similar performance but with reduced computational requirements and memory footprint. This is particularly important for real-time applications like the medical assistant.
3. Uncased Tokens: The "uncased" variant of the model treats all text as lowercase, which helps in standardizing the input and reduces the complexity of the model. In medical contexts, where capitalization may not always be consistent, using uncased tokens helps ensure robust predictions.
4. Sequence Classification: The selected model is designed for sequence classification tasks, which aligns well with the problem of disease prediction based on symptom descriptions. It has a classification head that can be fine-tuned to predict the appropriate disease label.
5. Community and Documentation: The Hugging Face Transformers library is widely used and well-documented, making it easier to implement and fine-tune models for specific tasks. The availability of pre-trained models and tools for tokenization, training, and evaluation simplifies the development process.
6. Efficiency: The "distilbert-base-uncased" model provides a good balance between model size and performance. This is crucial for real-time applications like the medical assistant, where inference speed and memory usage are important considerations.
7. Good Generalization: Pre-trained language models like DistilBERT have demonstrated strong generalization capabilities across a wide range of natural language understanding tasks. This makes them suitable for capturing complex relationships between symptom descriptions and diseases.
8. **AI Product Design**

The designed AI product utilizes the "distilbert-base-uncased" language model to create a web-based medical assistant that predicts diseases based on user-entered symptom descriptions. The solution involves both back-end and front-end components.

Block Diagram: Back-End (API Call)

+-----------------------------+ +-------------------------------+

| | | |

| Disease Prediction | API | Web Server (Gradio) |

| AI Model (Backend) +<------->+ |

| | +---------+-------------------+

+-----------------------------+ |

|

+--------------------------------------v-----------------------+

| |

| API Endpoint (Predict Diseases Function) |

| |

+--------------------------------------------------------------+

Back-End (API) Description:

1. AI Model (Disease Prediction): The heart of the back-end, the "distilbert-base-uncased" model, performs disease prediction based on symptom descriptions.
2. API Endpoint: The AI model is incorporated into an API endpoint. This endpoint receives user-provided symptom descriptions and responds with predicted diseases and their probabilities.
3. Web Server: A web server hosts the API endpoint, enabling communication between the front-end and the AI model.

Block Diagram: Front-End

+---------------------------------------------------------------+

| User Interface |

| |

| +-------------------------+ +------------------+ |

| | Text Input | | Predicted | |

| | (Textbox) | | Diseases | |

| +-------------------------+ +------------------+ |

| | |

| v |

| |

| +-------------------------+ |

| | Submit Button | |

| +-------------------------+ |

+----------------------------------------------------------------+

Front-End Description:

1. User Interface: The front-end provides a simple and intuitive interface for users to input their symptom descriptions.
2. Text Input (Textbox): Users can enter their symptom descriptions into a textbox.
3. Submit Button: After entering symptoms, users can click the "Submit" button to initiate the prediction process.
4. Predicted Diseases: Once the prediction is made, the front-end displays the top two predicted diseases along with their corresponding probabilities (scores).

Workflow:

1. User Interaction: Users input symptoms in the textbox and click "Predict."
2. API Communication: The front-end sends an API request to the back-end API endpoint.
3. Disease Prediction: The back-end AI model processes the input and predicts diseases with corresponding probabilities.
4. Result Display: Predicted diseases and scores are sent back to the front-end and displayed to users.
5. **Front-end Application:**

Our front-end application, powered by Gradio, streamlines the process of disease prediction based on symptom descriptions. With an intuitive textbox, users effortlessly input their symptoms and trigger the prediction process by clicking "Predict." Gradio's seamless integration facilitates real-time communication with the back-end AI model through an API call. Leveraging the advanced "distilbert-base-uncased" model, the back-end accurately predicts the top two likely diseases and their scores. This information is then swiftly displayed on the user interface, providing users with valuable insights for informed decision-making. Gradio's efficiency ensures a smooth user experience, making advanced AI-driven disease prediction accessible and effective.

1. **Technical Explanation**

Our AI product leverages the power of the "distilbert-base-uncased" language model to provide accurate disease prediction based on symptom descriptions. The technical workflow involves data preprocessing, utilizing the architecture of the language model, and fine-tuning for the specific problem.

1. Data Pre-processing:

The dataset containing symptom descriptions and corresponding disease labels is loaded and split into training and testing sets. The symptom text is tokenized and encoded using the DistilBERT tokenizer, which segments the text into subword tokens and converts them into numerical representations. Attention masks are also created to indicate the positions of valid tokens. The disease labels are mapped to numerical values for classification.

1. Architecture of the "distilbert-base-uncased" Model:

The "distilbert-base-uncased" model is a variant of the BERT architecture, distilled to retain essential semantic information while reducing computational requirements. It comprises a transformer encoder with self-attention mechanisms that capture contextual relationships between tokens. The model embeds input tokens into continuous vectors, which are passed through multiple layers to learn contextualized representations.

1. Fine-Tuning for Disease Prediction:

To adapt the language model for disease prediction, the classification head is added to the model architecture. The classification head consists of a linear layer that takes the contextualized embeddings and predicts the disease labels. The model is fine-tuned on the training data using the AdamW optimizer, minimizing the cross-entropy loss between predicted and actual labels.

1. Prediction and Inference:

During inference, user-provided symptom descriptions are tokenized and encoded using the same tokenizer. These input encodings are then passed through the fine-tuned "distilbert-base-uncased" model. The logits from the classification head are converted to probabilities using the softmax function, providing the likelihood of each disease label. The top two diseases with the highest probabilities are returned as predictions.

The technical implementation integrates data preprocessing, architecture utilization, and fine-tuning to create an effective AI model for disease prediction. The "distilbert-base-uncased" model's transformer architecture, combined with task-specific fine-tuning, enables the model to capture intricate relationships between symptom descriptions and diseases, resulting in accurate and reliable predictions.

1. **Evaluation Metrics**

The performance of our AI product, the Disease Prediction Medical Assistant, is assessed using several appropriate evaluation metrics, based on the code provided. These metrics focus on measuring the accuracy and effectiveness of disease predictions made by the model.

1. Accuracy:

The primary evaluation metric is accuracy, which measures the proportion of correctly predicted disease labels out of the total predictions. It is computed as the sum of true positive and true negative predictions divided by the total number of predictions.

1. Precision:

Precision quantifies the fraction of true positive predictions among all positive predictions. In the context of disease prediction, precision indicates how many of the predicted diseases were indeed accurate. It is calculated as true positives divided by the sum of true positives and false positives.

1. Recall (Sensitivity):

Recall, also known as sensitivity or true positive rate, measures the proportion of true positive predictions among all actual positive instances. In the medical context, recall indicates how well the model captures all relevant diseases. It is computed as true positives divided by the sum of true positives and false negatives.

1. F1-Score:

The F1-score is the harmonic mean of precision and recall. It provides a balance between precision and recall and is particularly useful when the dataset is imbalanced. The F1-score is computed as 2 times the product of precision and recall, divided by the sum of precision and recall.

1. **Limitations and Ethical Considerations**

While our AI product, the Disease Prediction Medical Assistant, offers a valuable tool for disease prediction based on symptom descriptions, it is important to recognize its limitations and address ethical considerations associated with its use.

Limitations:

1. Limited Disease Labels:

The AI product is trained on a dataset containing only 24 disease labels. This limitation may affect the accuracy of predictions for less common or newly emerging diseases that are not present in the training data.

1. Bias in Data:

The dataset used for training may contain inherent biases due to factors such as the demographics of the population represented, healthcare disparities, or data collection methods. This bias could influence the model's predictions and potentially lead to disparities in healthcare outcomes.

1. Symptom Variation:

The model relies solely on symptom descriptions, which may not always provide a complete picture of a patient's health. Certain diseases may share similar symptoms, leading to potential misclassifications.

1. Overconfidence:

The model may exhibit overconfidence in its predictions, assigning high probabilities to incorrect disease labels. This can lead to inaccurate diagnoses and treatment recommendations.

Ethical Considerations:

1. Bias and Fairness:

The AI product may inadvertently reflect and perpetuate biases present in the training data, potentially resulting in unequal and unfair treatment across different groups of patients. Efforts should be made to mitigate bias and ensure equitable healthcare recommendations.

1. Transparency:

Users should be informed that the AI predictions are based solely on symptom descriptions and that the model's decisions may not be fully interpretable. Transparency in how predictions are generated is essential for building trust.

1. Informed Consent:

Users should be aware of the AI's capabilities, limitations, and potential risks before using the application. Informed consent is crucial to ensure that users make informed decisions about their health.

1. Ongoing Monitoring and Updates:

The AI product should be continuously monitored and updated to account for new medical research, evolving disease patterns, and changing patient demographics. Outdated or inaccurate information can lead to misdiagnoses and compromised patient care.

1. Patient-Doctor Relationship:

The AI product should complement, rather than replace, the role of medical professionals. It should be used as a tool to aid clinical decision-making, and the importance of consulting with healthcare providers should be emphasized.

1. **Documentation (README.md)**

Project Overview:

The Disease Prediction Medical Assistant is an AI-powered application that predicts diseases based on symptom descriptions. The project is implemented using the "distilbert-base-uncased" language model and Gradio for the front-end interface. This app can be used to identify potential diseases early on, allowing patients to seek medical attention and treatment promptly. Also, In situations where in-person consultations are not possible or desirable, the app can be used to provide remote diagnosis based on the user's symptoms.

Running the Gradio Application:

1. Install the required dependencies: pip install -r requirements.txt
2. Run the Gradio application: app.py
3. Access the application interface: Open a web browser and go to http://localhost:7860

Customization:

1. To use your own dataset, replace the Symptoms2dataset.csv file in the dataset/ directory.
2. To fine-tune and train the model, modify the training code and update the model\_weights.pth file in the model/ directory.

Note: This documentation provides a basic guide to running the Disease Prediction Medical Assistant. For detailed instructions and additional features, refer to the GitHub repository and code files.

Important Ethical Consideration: When using the AI product, ensure that you are aware of its limitations and ethical considerations. The predictions provided by the AI model are based solely on symptom descriptions and should not replace consultation with qualified medical professionals.

Disclaimer: This project is intended for educational and demonstration purposes only. It should not be used as a substitute for professional medical advice, diagnosis, or treatment. Always consult with a qualified healthcare provider for medical concerns.