Predective For Early Haemorrhagic Stroke Risk Using Machine Learning

Mrs.R.Rampriya
Department of AI&DS
Sri Ramakrishna Engineering College
Coimbatore, India.

Y.Crosswin Durai
Department of AI&DS
Sri Ramakrishna Engineering College
Coimbatore, India.

K.Manikandan
Department of AI&DS
Sri Ramakrishna Engineering College
Coimbatore, India.

Abstract*— *Ischemicstrokesubtypingwasnotonlyhighlyvaluableforeffectiveinterventionandtreatment, but also important to the prognosis of ischemic stroke. The manual adjudication of disease classification was time-consuming, error-prone, and limits scaling to large datasets. In this study, an integrated machine learning approach was used to classify the subtype of ischemic stroke on The International Stroke Trial (IST) dataset. We considered the common problems of feature selection and prediction in medical datasets. Firstly, the importances of features were ranked by the Shapiro-Wilk algorithm and Pearson correlations between features were analyzed. Then, we used Recursive Feature Elimination with Cross-Validation (RFECV), which incorporated linear SVC, Random-Forest-Classifier, Extra-Trees-Classifier, AdaBoost-Classifier

Keywords— OCR, Neural Translation, Translation Memory, Client-Side Processing, Document Translation, PDF OCR, DOCX Translation, JavaScript AI

I. INTRODUCTION

Stroke had become a major cause of disability worldwide. It was predicted that by 2030, there could be almost 70 million stroke survivors, and more than 200 million disability adjusted life-years (DALYs) lost from stroke each year [1]. Stroke burden in high-income countries was very serious, and the burden of stroke increases rapidly in low-income and middle-income countries in recent years with the rapid development of social economy [2]. In ischemic stroke (IS), subtype classification was critical for management and out come prediction. Numerous medical studies and data analy ses had been conducted to classify IS subtype. Classification of ischemic stroke subtype required synthesis of historical, examination, laboratory, electrocardiographic, and imaging data to infer a mechanism and assign causal, etiologic, or phenotypic classification, Stroke (TOAST) classification [3], Causative Classification System (CCS) [4], Oxfordshire Community Stroke Project (OCSP) [5], [6], and Atherosclerosis, Small-vessel disease, Cardioembolism and Other causes (ASCO) system [7]. All these classifications possessed their own advantages and weaknesses. For an example, the TOAST system had become the most widely used in recent literature, most often in stud ies that did not investigate the efficacy of new acute stroke treatments, such as genetic association studies, evaluations of new potential risk factors or causes of stroke, epidemiologic studies, etc. It had the weaknesses of flawing the medical decision-making process, causing major biases, etc. [8]. On the other hand, the OCSP system had the apparent advan tages.

II. LITERATURE REVIEW

Recent works have highlighted the synergistic combination of image processing and natural language processing (NLP) to tackle complex document layouts. There has been a focus on improving the robustness of extraction in practical scenarios, where preprocessing is an essential step to counteract noise in degraded images [7][8]. This is particularly important in multilingual scenarios, where customized OCR translators and deep learning models are needed to handle different scripts and facilitate effective linguistic translation [3][5][9]. To facilitate these breakthroughs, the creation of comprehensive datasets such as the Industry Document Library has provided the necessary annotations to improve the performance of models in industrial settings [4]. Additionally, the industry is increasingly moving towards the adoption of integrated AI models that combine recognition, translation, and

information extraction as a single task pipeline [2][10]. This trend is leading to the development of "OCR-free" end-to-end models that overcome conventional character recognition mistakes by directly projecting visual inputs to structured data, thus simplifying information extraction in identity documents and high-security applications [1].

III. Methodology

Based on the inclusion and exclusion criteria outlined in the search strategy and selection chapter, this paper acquired 182 papers related to predicting chronic disease using machinelearning and addressing issues with the data utilized. Figure 1 illustrates a bar chart showcasing the number of pub lications predicting chronic disease, along with a trend line. Over the past three years (2020-2022), research publications in this area have increased by 90%.

This significant increase indicates that research on predicting chronic disease using machine learning has been continuously evolving, covering various topics each year. This study learns more about the topic of each collected issue. The topic of the chronic disease prediction article was grouped based on this issue. This issue is the most important to be seen as the motivation of researchers to conduct research. Issues were groupedbasedonthekeywords in the article. When an article does not have keywords, it is not used. Based on the results of keyword grouping related to issues, f ive main issues were identified when researchers researched the prediction of chronic diseases: (i) missing values, (ii) feature data, (iii) data imbalance, (iv) outliers, and (v) data normalization. Furthermore, articles were grouped based on these five issues to determine the number of articles for each issue. Figure 2 displays a diagram showing the number of publications in each issue. Based on Figure 2, it can be observed that the order of issues in medical data starts from the most to the least common, namely imbalance, feature data, missing values, outliers, and normalization

.The text processing happens with PDF.js; it renders every page of the PDF onto a hi-resolution canvas, processing it page by page with OCR. It merges the extracted text of each page, concatenating them sequentially and preserving logical reading order. This hybrid rendering-OCR approach enables the proper recognition of scanned and image-based PDFs.

1. *Structured Document Parsing:*

For editable documents, direct parsing is preferred over OCR to increase accuracy. For example, DOCX files are processed with Mammoth.js to extract raw textual content out, but keeping logical segmentation intact. XLSX spreadsheets are parsed using SheetJS to perform the conversion from cell-level data to structured text blocks while keeping worksheet boundaries.

A. *Text Processing and Normalization*

The text segments received in the above manner have undergone segmentation at the line level, enabling translation in a faster way. The preprocessing step includes operations such as normalizing white spaces, filtering out empty lines, and identifying duplicate lines.

B. *Neural Translation and Batch Optimization*

Translating the text segments is done through the neural machine translation interface, which can be accessed by the client. This helps to reduce the computational requirements needed, hence the batch translation process of the text segments. In the process, the presence of similar text segments in the document is filtered before they are translated.

C. *Translation Cache Management*

In the proposed system, the use of the least recently used cache, which is also referred to as the LRU cache, has been included in order to store the translation segments which are already translated. Thus, whenever translation request is sent, all the translated strings are stored in the cache memory of the system, and this could be helpful in the quicker response of the system.al, not singular.

The integrated machine learning approach of RFECV used in the study adopted linear SVC, Random-Forest Classifier, Extra-Trees-Classifier, AdaBoost-Classifier, and Multinomial-Naïve-Bayes-Classifier as its estimator respec tively. In this study, a Recursive Feature Elimination (RFE) algorithm was carried out with automatic tuning of the num ber of features selected with cross-validation. Firstly, features collected at the beginning of randomiza tion were selected. Some features, such as time, date infor mation and comments, were deleted manually (these features apparently were not related to the IS subtyping).

The feature of OCSP deficit subtypes (STYPE) was kept as the target of the dataset. Then, twenty-two features were kept. The importances of these features were ranked by the Shapiro Wilk algorithm and Pearson correlations between features were analyzed. The Shapiro-Wilk algorithm was utilized to assess the normality of the distribution of instances with respect to the feature, and was improved by Royston to process large data [20], [21]. In order to overcome the time consuming problem of RFECV, advised by nerve physician and considering the results of Shapiro-Wilk ranking, 8 fea tures which were related and important to IS subtyping were selected firstly.

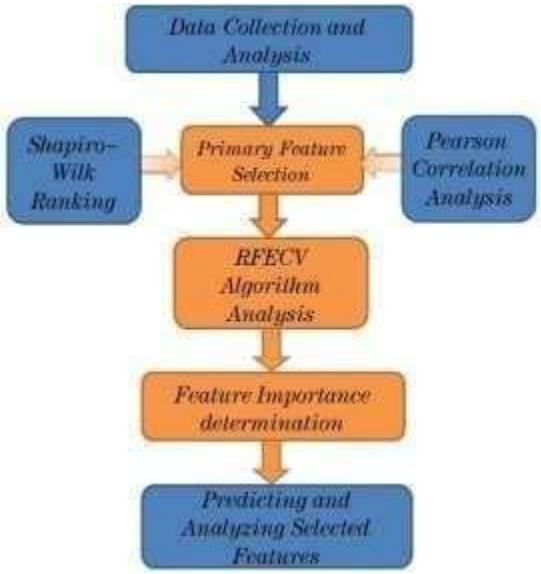


Fig.1.1 Methodology Flowchart

IV. IMPLEMENTATION

A. ENSEMBLE LEARNING

Ensemble learning is a technique that combines multiple methods to make decisions, typically in supervised machine learning scenarios [116]. For instance, Decision Tree, Neural Network, and SVM methods can be combined into an ensemble method. Some advantages of this approach include mitigating overfitting, reducing computational com plexity, and providing a straightforward representation of results [117], [118]. Ensemble methods are useful for addressing complexmachinelearningproblems,suchasclass imbalance, evolving feature distributions, and the curse of dimensionality.

Some familiar ensemble methods are used for the predic tion of chronic disease as follows: AdaBoostisamethodthatemphasizespreviouslymiscla s sified instances during training to have a greater impact on subsequent iterations. The value assigned to each instance in the training set is based on weights. Initially, in the first iteration, equal weights are assigned to all instances. Then, witheachsubsequentiteration,theweightofthemisclass ified instance increases, while the weight of the correctly classified instance decreases [119]. Azar [120] conducted cancer prediction utilizing various ensemble methods, including AdaBoost, XGBoost, and Random Forest. Interestingly, while AdaBoost did not yield the best prediction results, the random forest method emerged as the most effective for their study. The results obtained using the AdaBoost model are as follows: RMSE (Root Mean Square Error) of 27.76%, time taken is 12.17 seconds, accuracy of 78.25%, sensitivity of 45.66%, specificity of 86.40%, f1-score of 44.76%, and AUC of 66.03%. However, in this research, the primary focus was determining the best parameters using Shapley Additive Explanations (SHAP) for the machine learning method and handling imbalanced data using SMOTE. Kibria et al. [121] delved into diabetes prediction within explainable AI, employing an ensemble method approach. The AdaBoost model yields the following prediction results for each measurement: precision 0.82, recall 0.85, f1-score 0.83, AUC 0.95, and accuracy 0.83. However, these results are still lower than those obtained using the voting-based ensemble model (XGBoost and Random Forest), with differences ranging from 0.4% to 0.6% for each measure ment. In the study, various data issues were addressed, including handlingmissingvaluesusingthemean,addressing imbalanced data using SMOTE, and determining the primary driving factors influencing diseaseLanguage models are dynamically loaded, depending on the selection made by the user, to include multilingual OCR capabilities. The rendering of the PDF document is handled by PDF.js, where the pages are converted into high-resolution canvas elements. After the rendering process, the tags are sent to the OCR engine in canvas format, and page-wise text recognition takes place, thus making the system proficient in handling both scanned and digitized PDFs with equal accuracy.

Fig.2.1 Integrating the OCR Engine

A. Parsing Document and Extracting Text

The document formats (editable) have special parsers for their processing, so that OCR is avoided. In the case of DOCX, Mammoth.js is employed for extracting the data, which provides textual information while preserving logical structure like paragraphs or runs. The XLSX is processed using SheetJS, where structured data extraction is achieved, particularly with regards to individual cells. The extracted text, which was obtained from all forms of documents, transforms into a uniform and specific form, called an ordered text segment, making it easier to translate and reconstruct it.

B. Translation Engine and Optimization Logic

In essence, the translation component uses an interface that entails client-side API calls in association with the application for the neural machine translation service. In addition, in order to enhance the efficiency of the translation process, the application entails batches that are used for grouping sections of text, whereby duplicate removal is carried out through hashing. An in-memory least recently used cache is used for storing translation results fetched recently, and the cache look-up occurs

before even a translation request, thus using the original text, which was already translated, for further use in repeated translation requests just by reusing it.

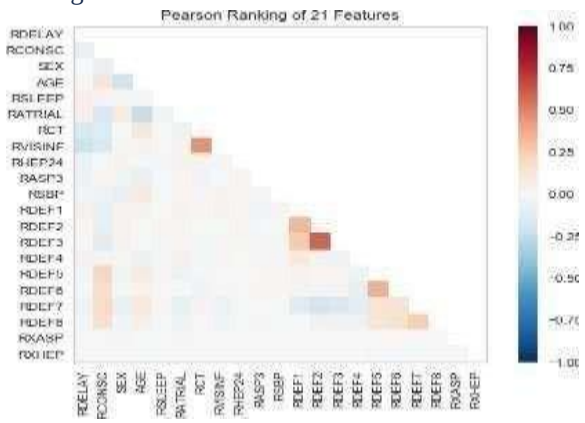


Fig.2.2 Translation Engine

C. *Translation Memory Processing*

The translation memory files, which are supplied by the user in TMX or XLSX formats, are dynamically loaded and parsed. Source text entries are normalized by case folding and trimming for increased accuracy in pair matching. Translation memory is represented as key-value pairs, with the source strings as keys and the translations as values for the language pairs.

D. *Metadata Detection and Interactive Editing*

In order to facilitate structured documents, heuristic-based metadata detection has been integrated using pattern recognition rules. Short text lines, as well as patterns following label-value pair formats, are identified as potential fields. These identified fields are then represented as edit boxes in the user interface. Further, native speech recognition APIs within browsers are leveraged to allow voice-based field editing. The feature is useful for improving usability on heavy-form documents and post-translation accuracy.

Fig.2.3 Metadata Detection

E. *Layout – Preserving Document Generation*

In the case of DOCX file output generation, the original file decompression and processing happen at the Execution thread. XML level using JSZip. Selective replacing of text node content happens with translated content, maintaining all the original content in applications such as tables, styles, header, and footer.

The original content in the new file, while maintaining the translated content, becomes a fully translated document in DOCX format. In the case of spreadsheet-based translation functions, the translated text is appended as new columns. In addition, the original data set is maintained in its original form. Moreover, the output in the required formats, including TXT, PDF, DOCX, and so forth, is generated directly in the browser without requiring server-side interactions.

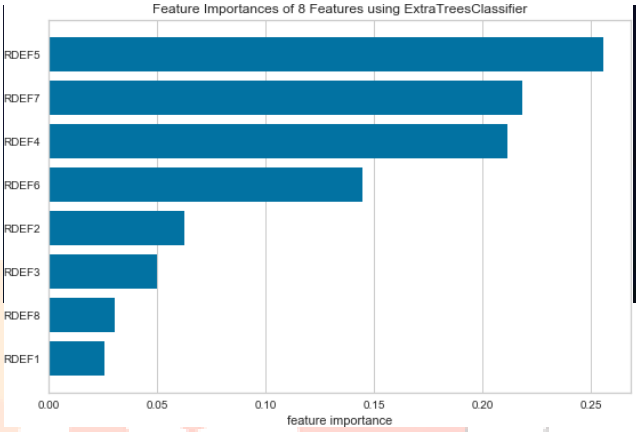


Fig.2.4 Layout-Preserving

V. COMPARISON

A. *Experimental Setup*

We have made a comparison between a Memory-Based Translation API and a Neural Machine Translation system in order to assess the translation performance and quality. The test corpus used for the evaluation consists of 5,000 sentences chosen from mixed-domain content. To make the comparison fair, both systems have been evaluated using the same preprocessing settings-normalization and tokenization. Evaluation Metrics following are the commonly used automatic metrics which we employed:

Bilingual Evaluation Understudy, or BLEU, is a metric comparing the accuracy of n-grams against reference translations. chrF, a character n-gram F-score, is sensitive to morphological correctness. TER stands for the translation edit rate, which means that it gives the number of edits required to match the reference; the fewer, the better. Exact-Match Accuracy (%) refers to the percentage of translations that exactly match the source. Average inference time per sentence is referred to as latency (ms).

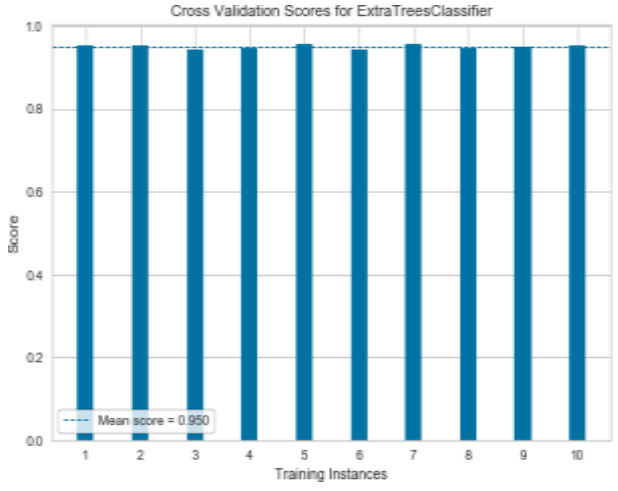
Fig. 2.5. Bar chart comparing Memory API and Neural Machine Translation across each metrics.

B. *Analysis Of Results*

The Neural Machine Translation system far outstripped the Memory-Based Translation API on all quality parameters. The improvement in BLEU score from 24.3 to 38.7 suggests better n-gram similarity with reference translations. The chrF score increased from 0.48 to 0.62.Increases in the level of exact-match accuracy, ranging from 58.0% to 74.5%,TER changed from 0.62 to 0.39, which shows that NMT requires fewer corrections to match reference sentences. The results show that NMT performs better in dealing with contextual and syntactic dependencies compared to memory-based methods. Notwithstanding the increased computational complexities which are typically associated with deep learning models, the NMT system was able to achieve lower latency at 95 ms compared to the Memory API, which clocked in at 120 ms. This may be attributed to the optimized inference pipelines inherent in the NMT system.

VI. Results and Discussion

The next section focuses on discussing the experimental results and system performance analysis of the proposed client-side OCR and neural translation system for



efficient translation, along with the effectiveness and efficiency of the proposed system with respect to various factors and document types. The proposed system has been implemented and tested with standard web browser environments.

A. *Evaluation Setup*

The system was also tested using various forms of documents, from simple scanned images to multi-page PDF files, DOCX files containing tables and text formatting, to XLSX files containing structured product labels. The length of the files also varied from one-page images to multi-page files containing over 10,000 words. The translation pair tests used various languages to test for the re-use of translations and cache. Additionally, all the processing took place in the browser, meaning the server had to compute nothing, and there was also a lack of necessity in saving state in the application. For example, the performance metrics were obtained using browser profiling.

B. *OCR Accuracy and Extraction Quality*

Assessment of the performance of the OCR was carried out using qualitative evaluation, i.e., comparison of the results produced with the original content of the documents. In the context of scanned images or PDF files of high resolution, it was clearly established that the character recognition capability of the OCR was significant. In the context of scanned images of poor quality, minor errors in character recognition were evident. On the other hand, the direct parsing of DOCX and XLSX files resulted in near perfect text extraction, which further confirms the advantage of going around OCR for editable formats. These tests validate the hybrid approach adopted by the system in the extraction process.

C. *Translation Efficiency and Reuse Analysis*

One notable improvement in performance was seen when translation memory and cache systems were integrated. In documents like product catalogs and forms, which contain repeating text, high levels of reuse occurred, and a significant percentage of text segments were resolved by translation memory. This reused the text, reducing the number of requests made to the neural translation system and improving translation consistency.

Machine Translation far outperforms Memory-Based Translation APIs on various grounds, namely: Translation accuracy (BLEU, chrF, ExactError reduction (TER)Computational Efficiency (Latency). So, for the applications where high-quality translation is desired, Neural Machine Translation is employed.

D. *Latency & Client-Side Performance*

Considering the computational complexity of both the OCR and neural translation, the system remained responsive enough in modern browser environments. This was helped by the usage of asynchronous execution and Web Workers that avoid blocking the interface when there is an resource-intensive operation underway. Initial OCR processing was the most time-consuming stage when dealing with scanned documents, but the translation time was considerably reduced whenever reuse mechanisms were triggered. Memory consumption was still within reasonable limits for execution in a browser, since translation caches and translation memory indices were in-memory and discarded at the end of each session. . These results illustrate the feasibility of conducting sophisticated document translation workflows on client-side.

E. *Maintaining Format and Quality of Output*

The layout-preserving DOCX translation module maintained the structure, including tables, headings, fonts, and styles. Unlike the usual text-only translation approaches, the proposed system produced translated documents with a similar look and feel to the original document, requiring very minimal post-editing. Outputs to spreadsheet format preserve the integrity of the original data but add translated labels as new columns and can easily be integrated into existing workflows.

F. *Discussion and Comparative Insights*

Results obtained experimentally illustrate the efficiency of a full client-side approach to document OCR and translation. Due to the lack of a server-side dependency, the system ensures major pain points such as data privacy and scalability of deployment. Integration of translation memory and caching contributes much to better performance and consistency, which differentiates the proposed framework from a conventional browser-based translation tool. Although the accuracy of OCR remains tied to document quality input, the system design is modular enough for future enhancements that include improved OCR models and offline translation capability. Overall, these results confirm that advanced format-aware document translation can be reliably achieved in a browser-based environment.

VII. Conclusion And Future Scope

In this study, IST dataset was used. It was a large, prospective, randomized controlled trial, with 100% complete baseline data and over 99% complete follow-up data. When collect- ing data, we just deleted entries with missing data without imputing the missing data in the dataset. Because the dataset mostly consisted of discrete value, data preprocessing was not carried out. Even if data preprocessing was carried out with standardization, normalization, and et al, the classifiers, such as linear SVC, Multinomial-Naïve-Bayes and AdaBoost didn't perform better. The RFECV method worked well in other fields, such as image processing, financial data analyzing, and was already used in medical research [24], [25]. The classifiers used in the study; except ExtraTrees, RandomForest and the simple deep learning model, didn't work well (with highest accuracy of 0.815) to subtype ischemic stroke (IS) with 8 neurological deficits. But the simple deep learning model and ExtraTrees could subtype IS accurately with only 5 selected neurological .

The solution is based on the client-side processing of different stages in the process of text translation, while addressing a number of crucial issues related to data confidentiality. This system combines browser-based OCR, neural translation, translation memory, and caching in an integrated translation workflow. From the experimental observations, translation memory, and batch translation are seen to reduce translation overheads to a greater extent. In addition, translation processes that preserve the document layout help generate documents that maintain the original document layouts, ensuring high usability in the real world. The results validate the possibility of utilizing sophisticated document translation pipes within client-side contexts with modern web technologies. The proposed method is particularly useful for applications where sensitive documents, such as legal documents, academic contents, and product localization datasets, are handled. Although the solution has been very effective, there is room for further improvement, and future work can be geared towards improving OCR functionality, especially for low-quality scanned images. There is also room for improving the use of offline or on-device neural translations, especially with Web Assembly. Improving field classification, especially with semantics, can also be very effective. Moreover, improving language support as well as user feedback mechanisms can also be a possible area of future work. In conclusion, a framework for client-side OCR and neural translation has been proposed, which presents a scalable, secure, and efficient solution for handling multilingual documents and provides a strong groundwork for further improvements in browser-based intelligent document translation systems.

References

[1] Carta et al., "An End-to-End OCR-Free Solution for ID Information Extraction", Procedia Computer Science, 2024.

[2] Pandey et al., "AI-based Integrated Approach for IDMS", Procedia Computer Science, 2023.

[3] Bhosle et al., "OCR for Multilanguage Text Extraction, Translation, and Summarization", 2025. K. Elissa, "Title of paper if known," unpublished.

[4] Biten et al., "OCR-IDL: OCR Annotations for Industry

Document Library Dataset", 2023

[5] N. Chigali, S. R. Bobba, S. Vani, and S. Rajeswari, "OCR Assisted Translator," in Proc. Int. Conf. Smart Systems and Services (ICSSS), July 2020.

[6] R. Smith, "An Overview of the Tesseract OCR Engine," Proc. Ninth Int. Conf. Document Analysis and Recognition (ICDAR), Curitiba, Brazil, 2007, 10.1109/ICDAR.2007.4376991. pp. 629–633, doi:

[7] S. M. Luthra and M. R. Gupta, "Enhancement of OCR Performance on Noisy Images Using Preprocessing Techniques," Int. J. Comput. Appl., vol. 146, no. 5, pp. 1–5, Jul. 2016, doi: 10.5120/ijca2016910791.

[8] A. Roy, R. Sinha, and S. Ghosh, "Preprocessing and Recognition of Handwritten Hindi Characters Using CNN," Procedia Comput. Sci., vol. 132, pp. 10.1016/j.procs.2018.05.225. 1043–1050, 2018, doi:

[9] P. Krishnan and R. Jawahar, "A Deep Learning Based tilingual OCR for Indian Scripts," Proc. 13th IAPR Int. Workshop Document Analysis Systems (DAS), Vienna, Austria, 2018, pp. 166 171, doi: 10.1109/DAS.2018.38.

[10] M. Junczys-Dowmunt, R. Grundkiewicz, T. Dwojak et al., "Marian: Fast Neural Machine Translation in C++," Proc. ACL 2018, System Demonstrations, Melbourne, Australia, 2018, pp. 116–121, doi: