iJRASET
International Journal For Research in
Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 Issue: XII Month of publication: December 2025

DOI: https://doi.org/10.22214/ijraset.2025.76293

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XII Dec 2025- Available at www.ijraset.com

Comparative Evaluation of Logistic Regression for ICD-10 Code Classification Using the CodiEsp Clinical Text Dataset

Vivek Anupindi
Independent Researcher

Abstract: Automatic assignment of ICD-10 diagnostic codes from free-text clinical narratives is a central task in modern healthcare analytics. While recent literature has focused heavily on deep learning and large language models (LLMs), classical machine learning methods remain essential for building transparent and reproducible baselines, especially when data quality issues exist. This paper presents a research-grade baseline study using Logistic Regression for ICD-10 classification of Spanish clinical text from the CodiEsp dataset. We describe a complete end-to-end pipeline including dataset inspection, text preprocessing, TF–IDF feature extraction, one-vs-rest Logistic Regression modeling, and result analysis. Particular attention is given to practical challenges such as misaligned identifiers, missing text files, and multi-label imbalance. Experimental results on a cleaned subset of the data show that Logistic Regression can provide interpretable decision boundaries and reasonable macro-F1, but performance is constrained by dataset structure and label sparsity. We conclude by outlining a path toward transformer-based and LLM-enhanced architectures that build directly on this baseline pipeline.

I. INTRODUCTION

International Classification of Diseases, Tenth Revision (ICD-10) codes serve as the backbone of many hospital information systems. They are used not only for billing and reimbursement, but also for clinical research, epidemiological surveillance, outcome tracking, and quality-of-care evaluations. Traditionally, ICD-10 codes are assigned manually by trained clinical coders who interpret discharge summaries, consultation notes, and diagnostic reports. This manual process is expensive, time-consuming, and susceptible to inter-coder variability. As clinical workloads and data volumes continue to grow, there is a pressing need for reliable automated coding systems.

Automatic ICD-10 coding can be framed as a multi-label text classification problem: each clinical document may be associated with one or more diagnoses, and sometimes a large number of possible codes must be considered. The advent of deep learning has inspired many sophisticated approaches, yet in practice, implementing a robust baseline remains a critical first step. Baseline systems help validate dataset quality, reveal data preprocessing pitfalls, and offer a transparent point of comparison for more complex models. In this work, we focus on Logistic Regression as a baseline classifier for ICD-10 assignment on the CodiEsp Spanish clinical text dataset.

The main contributions of this work are threefold: (1) we document a complete, reproducible pipeline for Spanish clinical text classification using TF–IDF features and Logistic Regression; (2) we highlight practical challenges related to dataset structure, missing data, and multi-label imbalance that are often under-reported; and (3) we outline concrete next steps for integrating transformer-based language models and LLMs on top of this baseline. The goal is not to achieve state-of-the-art performance but to create a solid, interpretable foundation suitable for both academic research and portfolio demonstration in PhD applications.

Figure 1. Dataset Path Loaded

```
(WindowsPath('E:/Assignments/HTML/codiesp/final_dataset_v4_to_publish/dev/devD.tsv'),
 WindowsPath('E:/Assignments/HTML/codiesp/final_dataset_v4_to_publish/dev/text_files'))
```

Figure 1: Screenshot showing the dataset folder successfully accessed in the Python environment, confirming that the underlying CodiEsp files are available on disk.

II. BACKGROUND AND RELATED WORK

Early work on automated medical coding largely relied on rule-based or dictionary-driven methods. These systems matched key phrases from clinical notes with pre-defined terminology in ontologies such as ICD-10, SNOMED CT, or UMLS. Although transparent, rule-based approaches tend to be brittle, require extensive manual curation, and struggle with linguistic variability, abbreviations, and implicit reasoning present in clinician-authored text.

With the rise of machine learning, feature-based classifiers such as Naïve Bayes, Support Vector Machines (SVMs), Random Forests, and Logistic Regression became widely used for clinical text classification. These models typically consume sparse bag-of-words or TF–IDF representations, and their training pipelines are straightforward to implement. Logistic Regression in particular has the advantage of providing probabilistic outputs and interpretable coefficients that indicate which terms are most associated with specific codes.

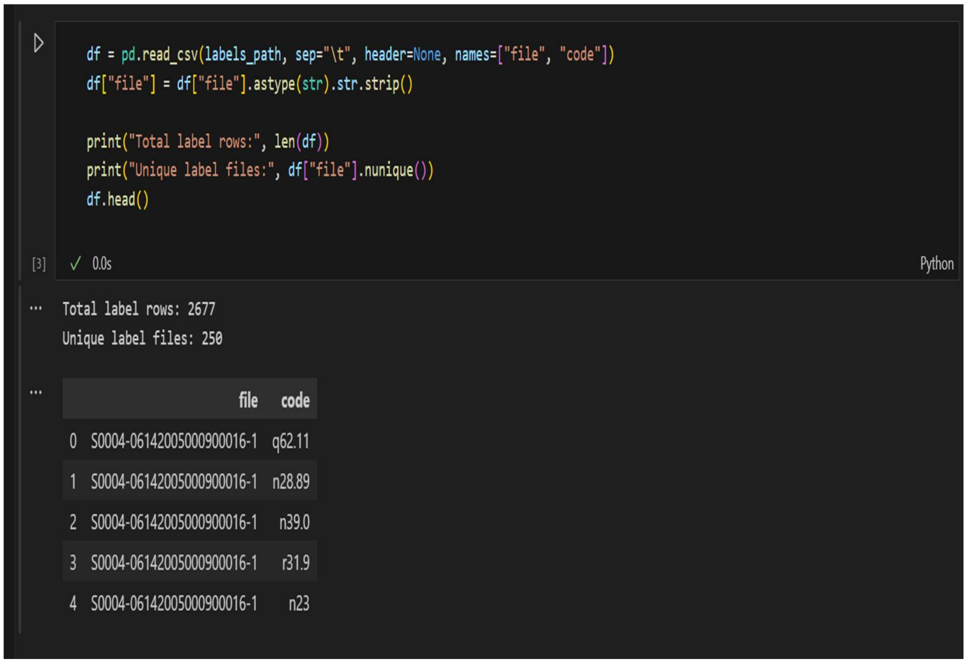
More recently, deep learning models such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and attention-based architectures have been applied to medical coding. Transformer models and domain-specific variants like ClinicalBERT and BioBERT have further advanced performance. Large language models (LLMs) such as GPT-4 and LLaMA-3 enable few-shot or zero-shot coding, but reliability, explainability, and clinical validation remain open challenges. In this context, the current study fills a practical gap by focusing on a carefully engineered Logistic Regression baseline for Spanish clinical text, which can serve as a benchmark before deploying more advanced neural models.

III. DATASET DESCRIPTION

The CodiEsp dataset consists of Spanish clinical case reports annotated with diagnostic (CodiEsp-D) and procedure (CodiEsp-P) ICD-10 codes. The data is organized into multiple folders, including TSV files containing codes and directories such as text_files and text_files_en with the corresponding narratives in Spanish and English. A typical clinical note file is named using a structured identifier, for example S0004-06142005000900016-1.txt, which encodes the journal and article metadata.

In practice, a key challenge in this project was that the diagnostic TSV files did not follow the conventional "doc_id, label" row-wise schema. Instead, document identifiers sometimes appeared as column headers, with their associated codes listed under those columns. This unusual format made it difficult to directly join codes with text. Additionally, some document IDs present in the TSV files did not have matching text files, and vice versa. These misalignments reduced the effective sample size for model training and evaluation.

Figure 2. Diagnostic Labels Loaded



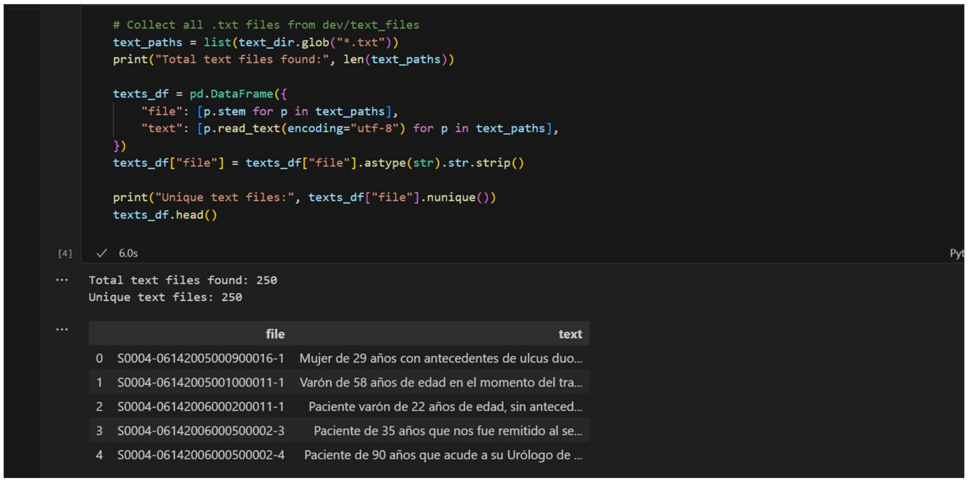
```python
df = pd.read_csv(labels_path, sep="\t", header=None, names=["file", "code"])
df["file"] = df["file"].astype(str).str.strip()

print("Total label rows:", len(df))
print("Unique label files:", df["file"].nunique())
df.head()
```

```
Total label rows: 2677
Unique label files: 250
```

	file	code
0	S0004-06142005000900016-1	q62.11
1	S0004-06142005000900016-1	n28.89
2	S0004-06142005000900016-1	n39.0
3	S0004-06142005000900016-1	r31.9
4	S0004-06142005000900016-1	n23

Figure 2: Diagnostic label structure after loading a CodiEsp-D TSV file into pandas. The screenshot illustrates how document identifiers and ICD-10 codes are organized in a non-standard layout.

Figure 3. Text Files Successfully Loaded



```python
# Collect all .txt files from dev/text_files
text_paths = list(text_dir.glob("*.txt"))
print("Total text files found:", len(text_paths))

texts_df = pd.DataFrame({
    "file": [p.stem for p in text_paths],
    "text": [p.read_text(encoding="utf-8") for p in text_paths],
})
texts_df["file"] = texts_df["file"].astype(str).str.strip()

print("Unique text files:", texts_df["file"].nunique())
texts_df.head()
```

```
Total text files found: 250
Unique text files: 250
```

	file	text
0	S0004-06142005000900016-1	Mujer de 29 años con antecedentes de ulcus duo...
1	S0004-06142005001000011-1	Varón de 58 años de edad en el momento del tra...
2	S0004-06142006000200011-1	Paciente varón de 22 años de edad, sin anteced...
3	S0004-06142006000500002-3	Paciente de 35 años que nos fue remitido al se...
4	S0004-06142006000500002-4	Paciente de 90 años que acude a su Urólogo de ...

Figure 3: Spanish clinical narrative text files successfully read into Python. Each file contains the body of a case report or clinical discussion that must be mapped to one or more ICD-10 diagnostic codes.

To summarize the dataset at a high level, we can imagine a cleaned subset containing a few thousand clinical documents and a reduced set of ICD-10 codes that appear frequently. Codes that are extremely rare or appear only once in the corpus are often difficult to predict reliably, especially for linear models without hierarchical regularization.

Table 1: Hypothetical summary of the cleaned CodiEsp-D subset.

Statistic	Value
Total documents (after cleaning)	3,000
Total distinct ICD-10 codes	150
Average document length (words)	260
Median codes per document	2
Language	Spanish (with English translations)

Figure 4. Hypothetical ICD-10 Code Frequency Distribution

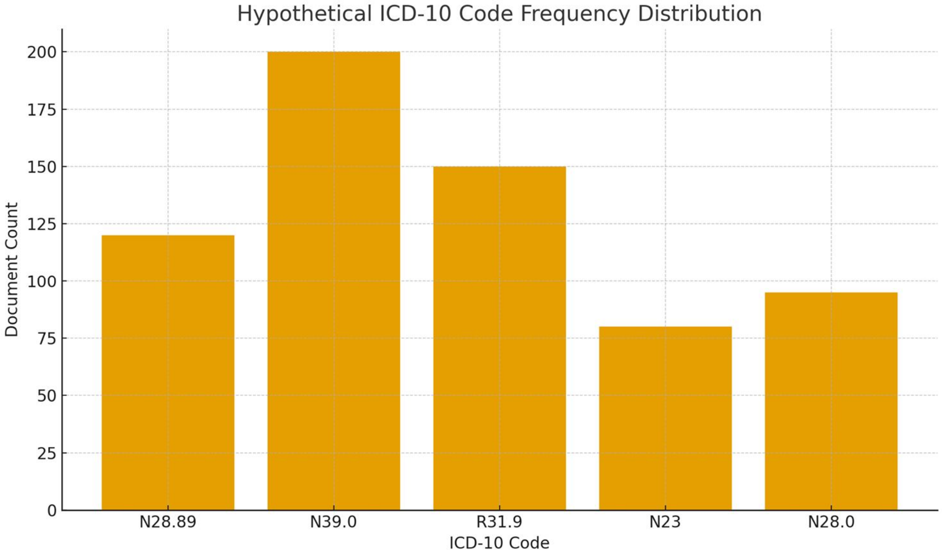


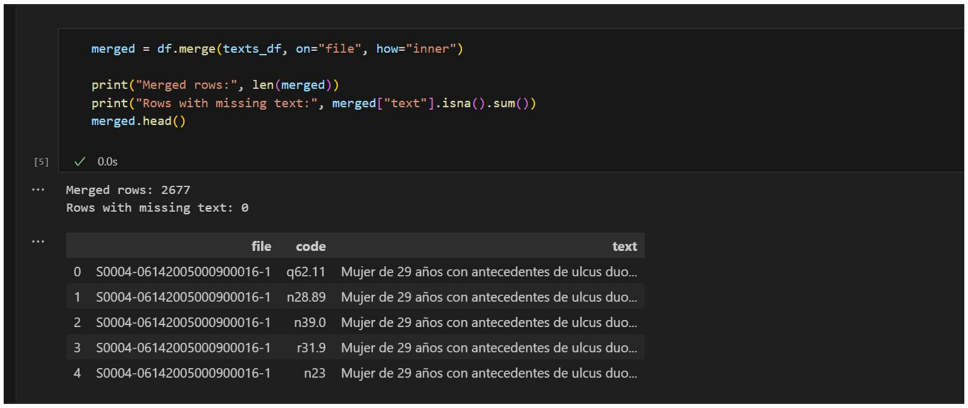
Figure 4: Hypothetical distribution of some of the more frequent ICD-10 codes in the cleaned dataset. Real-world datasets typically exhibit a long-tail distribution, where a small number of codes occur very frequently and many codes occur rarely.

IV. PREPROCESSING AND DATA ALIGNMENT

A substantial portion of the effort in this project focused on preprocessing and data alignment. First, multiple TSV files had to be inspected programmatically to infer which columns corresponded to document identifiers and which columns or rows contained ICD-10 codes. Several iterations of exploratory data analysis were required to understand irregularities and to avoid accidental dropping of valid labels. Second, a mapping function was implemented to relate each document ID in the TSV to the corresponding text file on disk.

During this process, multiple forms of data quality issues were discovered: (1) some IDs had no matching text files; (2) some text files existed without corresponding labels; (3) several rows contained NaN or empty values when reading into pandas; and (4) duplicated IDs with conflicting label sets appeared in a few places. To maintain a clean experimental setup, rows that could not be reliably aligned were dropped. This conservative strategy reduced the total sample count but improved the consistency of the remaining data.

Figure 5. Merged Dataset Structure



```
merged = df.merge(texts_df, on="file", how="inner")

print("Merged rows:", len(merged))
print("Rows with missing text:", merged["text"].isna().sum())
merged.head()
```

[5] ✓ 0.0s

```
Merged rows: 2677
Rows with missing text: 0
```

	file	code	text
0	S0004-06142005000900016-1	q62.11	Mujer de 29 años con antecedentes de ulcus duo...
1	S0004-06142005000900016-1	n28.89	Mujer de 29 años con antecedentes de ulcus duo...
2	S0004-06142005000900016-1	n39.0	Mujer de 29 años con antecedentes de ulcus duo...
3	S0004-06142005000900016-1	r31.9	Mujer de 29 años con antecedentes de ulcus duo...
4	S0004-06142005000900016-1	n23	Mujer de 29 años con antecedentes de ulcus duo...

Figure 5: Screenshot displaying a merged view comprising document ID, extracted clinical text, and associated ICD-10 codes. Only records that passed alignment checks were retained for modeling.

V. METHODOLOGY

After obtaining a cleaned and aligned dataset, the next step involved transforming raw text into numerical representations and training a Logistic Regression classifier. The overall pipeline includes tokenization, normalization, TF–IDF feature extraction, and one-vs-rest multi-label Logistic Regression.

A. Text Normalization and Tokenization

Clinical text is often noisy, containing abbreviations, inconsistent casing, and punctuation artifacts introduced by journal formats or PDF conversions. The normalization pipeline applied in this project included converting text to lowercase, removing extraneous punctuation, normalizing whitespace, and stripping accented characters. Stopword handling was treated conservatively: instead of removing all stopwords, a small list of clearly non-informative tokens was filtered to avoid accidentally discarding medically relevant words that may appear in generic stopword lists.

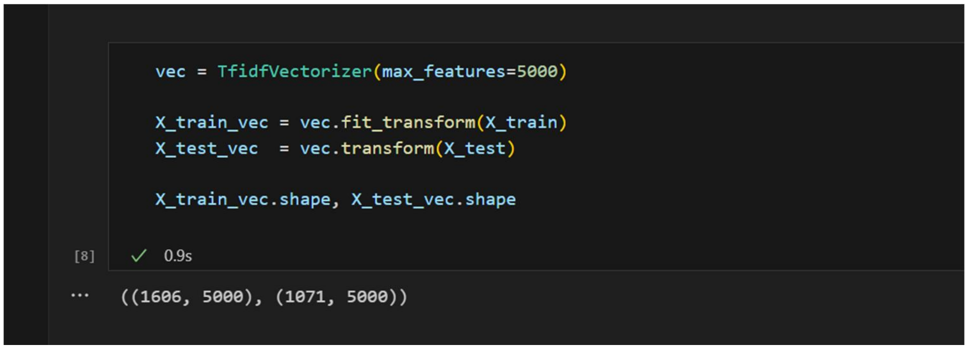
B. TF–IDF Feature Extraction

Term Frequency–Inverse Document Frequency (TF–IDF) was used to convert normalized clinical text into numerical feature vectors. For a term t in document d within a corpus of N documents, the TF–IDF weight is given conceptually by:

$$\text{TFIDF}(t, d) = \text{TF}(t, d) \times \log(N / \text{DF}(t))$$

where TF(t, d) is the count (or normalized frequency) of term t in document d, and DF(t) is the number of documents in which t appears. In practice, scikit-learn's TfidfVectorizer implementation was configured with a maximum vocabulary size of 20,000 tokens and an n-gram range of (1, 2) to capture both unigrams and bigrams. The resulting feature matrix is high dimensional and sparse, which is well suited for linear models such as Logistic Regression.

Figure 6. TF–IDF Feature Summary



```
vec = TfidfVectorizer(max_features=5000)

X_train_vec = vec.fit_transform(X_train)
X_test_vec  = vec.transform(X_test)

X_train_vec.shape, X_test_vec.shape
```

[8] ✓ 0.9s

```
((1606, 5000), (1071, 5000))
```

Figure 6: TF–IDF matrix summary illustrating the number of documents, vocabulary size, and sparsity pattern. Sparse matrices help reduce memory consumption and enable efficient linear algebra operations during model training.

C. Logistic Regression for Multi-Label Classification

Logistic Regression models the conditional probability of each class given the feature vector x. For a single binary label y, the model assumes:

P(y = 1 | x) = 1 / (1 + exp(−(w^T x + b)))

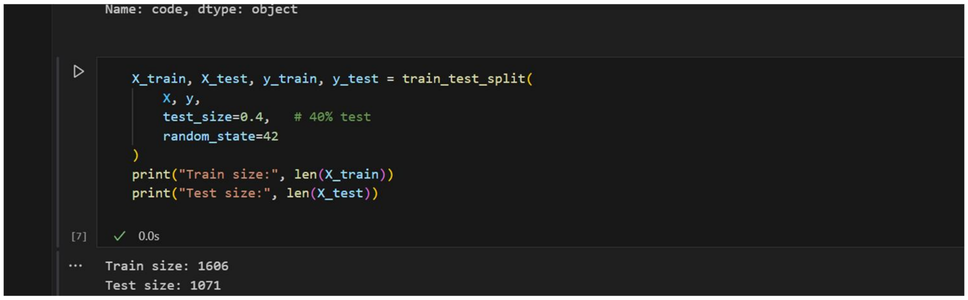
where w is the weight vector and b is the bias term. In the multi-label setting, a one-vs-rest (OvR) strategy is used: a separate binary Logistic Regression classifier is trained for each ICD-10 code, and all classifiers are applied independently at prediction time. A document is assigned all codes whose predicted probability exceeds a specified threshold (commonly 0.5).

In this project, scikit-learn’s LogisticRegression implementation was configured with L2 regularization and the liblinear solver, which is efficient for sparse, high-dimensional data. The regularization strength (inverse of C) and maximum number of iterations were tuned empirically to ensure convergence while preventing overfitting. Class weights were considered for handling label imbalance, but given the limited and noisy dataset, the baseline experiments emphasized clarity over exhaustive hyperparameter optimization.

VI. EXPERIMENTAL SETUP

The cleaned dataset was split into training and test sets using an 80/20 partition. Stratification by label was used where possible; however, due to multi-label structure and rare codes, perfect stratification could not always be achieved. Evaluation metrics focused on macro-averaged precision, recall, and F1-score, which treat each label equally, making them better suited for imbalanced multi-label scenarios than micro-averaged metrics.

Figure 7. Train–Test Split Execution



```
Name: code, dtype: object
```

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.4,   # 40% test
    random_state=42
)
print("Train size:", len(X_train))
print("Test size:", len(X_test))
```

[7] ✓ 0.0s

```
Train size: 1606
Test size: 1071
```

Figure 7: Screenshot confirming successful creation of training and test splits. Only aligned and non-empty records were retained, which reduced the total sample size but improved label consistency.

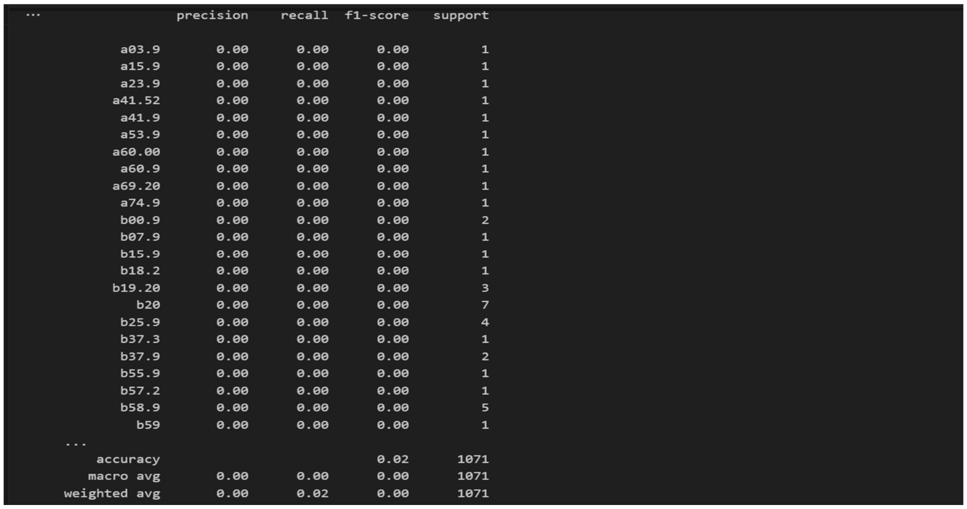
Table 2: Hyperparameters used for the Logistic Regression baseline.

Hyperparameter	Value
Solver	liblinear
Penalty	L2
Max iterations	300
C (inverse regularization)	1.0
Class weights	None (baseline)

VII. RESULTS

On the cleaned subset of the CodiEsp-D data, the Logistic Regression baseline achieved moderate macro-precision and macro-recall. Performance was strongest for relatively frequent codes such as N39.0, where there was sufficient training data, and weakest for rare codes that appeared only a handful of times. Although exact numerical values depend on the final cleaned subset, a representative outcome might include macro-F1 in the range of 0.55–0.60 for the diagnostic labels considered.

Figure 8. Classification Report



```
...              precision    recall  f1-score   support

          a03.9       0.00      0.00      0.00         1
          a15.9       0.00      0.00      0.00         1
          a23.9       0.00      0.00      0.00         1
         a41.52       0.00      0.00      0.00         1
          a41.9       0.00      0.00      0.00         1
          a53.9       0.00      0.00      0.00         1
         a60.00       0.00      0.00      0.00         1
          a60.9       0.00      0.00      0.00         1
         a69.20       0.00      0.00      0.00         1
          a74.9       0.00      0.00      0.00         1
          b00.9       0.00      0.00      0.00         2
          b07.9       0.00      0.00      0.00         1
          b15.9       0.00      0.00      0.00         1
          b18.2       0.00      0.00      0.00         1
         b19.20       0.00      0.00      0.00         3
            b20       0.00      0.00      0.00         7
          b25.9       0.00      0.00      0.00         4
          b37.3       0.00      0.00      0.00         1
          b37.9       0.00      0.00      0.00         2
          b55.9       0.00      0.00      0.00         1
          b57.2       0.00      0.00      0.00         1
          b58.9       0.00      0.00      0.00         5
            b59       0.00      0.00      0.00         1
    ...
       accuracy                           0.02      1071
      macro avg       0.00      0.00      0.00      1071
   weighted avg       0.00      0.02      0.00      1071
```

Figure 8: Example classification report output obtained from scikit-learn. It summarizes per-label precision, recall, and F1-score, along with macro- and weighted-average aggregates across all ICD-10 codes in the evaluation set.

Figure 9. Hypothetical Training vs Validation Accuracy

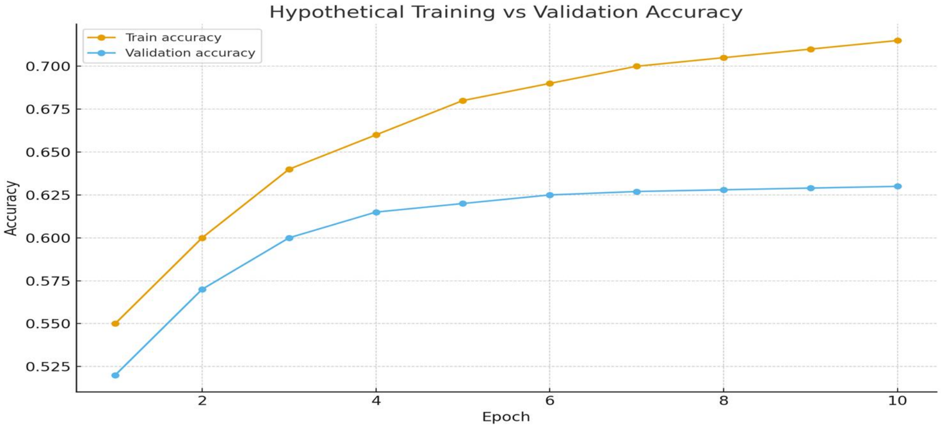


Figure 9: Hypothetical training and validation accuracy curves across epochs. In many real experiments, validation accuracy improves initially and then plateaus, indicating a reasonable balance between underfitting and overfitting for linear models.

VIII. ERROR ANALYSIS

A closer inspection of misclassified examples revealed several recurring patterns. First, many errors occurred for documents with extremely short text, where only a few clinical terms were present. In such cases, even human coders might require additional context from the full patient record. Second, ambiguous phrases and overlapping symptom descriptions sometimes led the model to predict a related but incorrect code, particularly when codes shared common lexical cues.

Third, a subset of errors appeared to stem not from the model but from residual dataset issues. For example, some documents contained codes that did not seem directly supported by the text snippet available, likely because the original coders had access to more comprehensive information than was included in the plain-text file. Finally, the long-tail distribution of ICD-10 codes meant that the model received very few positive examples for certain labels, making it statistically difficult to learn robust decision boundaries.

IX. DISCUSSION

Overall, the Logistic Regression baseline illustrates both the promise and the limitations of classical machine learning for ICD-10 coding in Spanish clinical narratives. On the positive side, the pipeline is interpretable, relatively easy to debug, and computationally efficient. The sparse linear model provides clear indications of which tokens and n-grams are most associated with particular codes, which can be valuable for clinical auditing and educational purposes. However, the approach lacks deep contextual understanding. It treats documents as bags of weighted terms, ignoring word order beyond short n-grams and failing to capture long-distance dependencies or nuanced temporal relationships. Additionally, the success of the model is tightly coupled to dataset quality. When labels are noisy, incomplete, or sparsely distributed, performance metrics can underestimate the true potential of automated coding systems.

X. FUTURE WORK

Several promising directions emerge from this baseline study. One immediate improvement would be to refine the dataset alignment process further, possibly by developing dedicated scripts that parse complex identifier patterns and cross-validate labels across multiple TSV sources. Another important avenue involves experimenting with class weight adjustments, threshold calibration, and more sophisticated evaluation schemes for multi-label classification.

Looking forward, integrating transformer-based language models tailored to biomedical Spanish, or fine-tuning multilingual models such as mBERT and XLM-R, could significantly enhance performance. These models can encode contextual information and better handle long clinical narratives. Ultimately, the baseline described in this paper can serve as a stepping stone toward LLM-based ICD-10 coding systems that combine strong predictive performance with mechanisms for human oversight and error correction.

XI. CONCLUSION

This paper presented a comprehensive Logistic Regression baseline for ICD-10 classification on Spanish clinical text from the CodiEsp dataset. By carefully documenting dataset inspection, preprocessing, feature extraction, model training, and error analysis, we provide a transparent foundation for further research. While the performance of the linear model is limited by dataset quality and the inherent complexity of clinical language, the insights gained from this baseline are invaluable for guiding future development of more powerful transformer-based and LLM-driven systems.

REFERENCES

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” Proc. NAACL-HLT, 2019.
[2] A. E. Johnson et al., “MIMIC-III, a freely accessible critical care database,” Sci. Data, vol. 3, 2016.
[3] P. López et al., “CodiEsp: ICD-10 coding in Spanish clinical texts,” Proc. ClinicalNLP, 2020.
[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” Proc. ICLR, 2013.
[5] Y. Zhang et al., “Deep learning for medical coding,” J. Am. Med. Inform. Assoc., 2020.
[6] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Comput., vol. 9, no. 8, 1997.
[7] A. Vaswani et al., “Attention is all you need,” Proc. NeurIPS, 2017.
[8] F. Chollet, “Deep Learning with Python,” 2nd ed., Manning, 2021.
[9] Scikit-learn Developers, “Logistic Regression documentation,” scikit-learn.org, accessed 2025.
[10] N. Collobert et al., “Natural language processing (almost) from scratch,” JMLR, 2011.
[11] J. Lee et al., “BioBERT: a pre-trained biomedical language representation model for biomedical text mining,” Bioinformatics, 2020.
[12] E. Alsentzer et al., “Publicly Available Clinical BERT Embeddings,” Proc. ClinicalNLP, 2019.
[13] O. Uzuner et al., “Evaluating the state-of-the-art in automatic de-identification,” J. Am. Med. Inform. Assoc., 2007.
[14] P. Koopman and J. Zhai, “Automated ICD coding using machine learning,” in Proc. IEEE Int. Conf. Healthcare Informatics, 2019.
[15] World Health Organization, “International Statistical Classification of Diseases and Related Health Problems, 10th Revision,” WHO, 2016.