Evaluating Model Performance for Fake News Detection: A Study on Traditional vs Deep Learning Approaches

Ethan Du[1]

Received April 17, 2026
Accepted August 29, 2026
Electronic access September 15, 2026

Political misinformation spreads rapidly online, motivating automated tools to support verification. This study compared four traditional classifiers—Multinomial Naive Bayes, Logistic Regression, Linear Support Vector Machine, and Decision Tree—with a convolutional neural network, a bidirectional long short-term memory network, and DistilBERT on the text-only, six-class LIAR benchmark. All models used the official training, validation, and test partitions and common evaluation criteria, though the model families each retained architecture-specific preprocessing. Candidate configurations were evaluated on the validation split, and checkpoints were selected using validation macro-F1. Performance was assessed through accuracy, macro-F1, micro- and macro-AUC, per-class metrics, within-one accuracy, bootstrap confidence intervals, and recorded fitting/training and inference runtimes. Logistic Regression achieved the highest point estimates for accuracy (0.256) and macro-F1 (0.255), but DistilBERT Config B achieved comparable accuracy (0.253) and macro-F1 (0.246) and the highest micro-AUC (0.643), macro-AUC (0.618), and within-one accuracy (0.613). However, the ordering did not establish statistical superiority, as the leading models' confidence intervals overlapped. Absolute performance remained low: even the strongest classifier misclassified nearly three-quarters of test statements. Logistic Regression required 3.558 seconds for classifier fitting, whereas DistilBERT Config B required 20,570 seconds for five epochs and 153 seconds for test inference. Under this text-only LIAR setting, greater model complexity did not yield a clear overall advantage. Thus, the study finds that these systems are unsuitable as stand-alone fact-checkers, although DistilBERT's ranking and ordinal behavior may merit evaluation in human-review prioritization workflows.

Keywords: Fake News Detection; Text Classification; LIAR Dataset; Natural Language Processing; Benchmark Evaluation; Computational Efficiency

Introduction

Fake news and political misinformation are persistent challenges for public discourse within digital information environments. In today's fast-paced digital world, social-media platforms reduce the cost and time required to publish and redistribute content, allowing misleading claims to reach large audiences before professional verification can occur. Prior research surrounding the 2016 United States election documented the financial and political incentives associated with fabricated stories[1], while a large-scale analysis of Twitter found that false news generally spread farther, faster, deeper, and more broadly than true news[2]. Such conditions have increased interest in automated systems that help identify potentially unreliable content. However, misinformation detection remains difficult as deceptive material may imitate legitimate reporting, selectively present accurate details, or require contextual evidence beyond the wording alone[3].

To address the challenge of identifying fake news, the present study focused on claim-level veracity classification using the LIAR benchmark. The LIAR benchmark contains 12,836 short political statements collected from PolitiFact and labeled across six ordered truthfulness categories: pants-fire, false, barely-true, half-true, mostly-true, and true[4]. Along with the political statements, the dataset also contains metadata covering the speaker, political affiliation, subject, context, and historical credibility associated with every claim. Wang's original experiments demonstrated that combining metadata with statement text improved performance compared to text-only modeling, uncovering the value of contextual information[4]. Thus, LIAR provides a useful and challenging benchmark for evaluating how much truthfulness information may be inferred from short claim text alone.

Traditional machine-learning classifiers serve as valuable baselines for the evaluation. Models like Multinomial Naive Bayes, Logistic Regression, Support Vector Machines, and Decision Trees can operate on sparse bag-of-words or term frequency-inverse document frequency representations.

[1] *Westmont High School; 4805 Westmont Ave, Campbell, CA 95008, USA. Corresponding author. Email: ethandu2027@gmail.com*

Benchmark studies by Khan et al.[5] and Alghamdi et al.[6] compared traditional, neural, and transformer-based models across multiple fake-news datasets, finding that relative performance varied across datasets and text-representation choices. Ahmad et al. further reveal that ensemble combinations of traditional classifiers could improve fake-news classification within their evaluated datasets[7]. Due to the LIAR dataset's imbalance across six classes, macro-F1 is very useful: calculating F1 separately for every class and weighing each class equally prevents larger categories from dominating the result[8]. Other recent studies have also examined large sets of supervised classifiers[9], ensemble feature extraction[10], linguistic features[11], and stacked classification frameworks[12], further demonstrating that classical methods remain relevant components of comparative fake-news research.

Deep learning offers a different approach by learning representations directly from the provided text. FakeBERT, a deep learning model designed to detect fake news, combines contextual BERT representations with convolutional blocks[13], while the hybrid model proposed by Nasir et al. combines convolutional and recurrent layers to capture local and sequential patterns[14]. Additional peer-reviewed studies have examined deep convolutional networks[15], tensor-decomposition models[16], multiscale BERT-convolutional frameworks[17], contextualized text representations[18], pairwise-input architectures[19], sequential deep ensembles[20], and dual-BERT networks[21]. Though these studies reported strong results within their respective datasets, their scores cannot be directly compared with the six-class LIAR classification. Many of the studies use longer articles, choose binary fake-versus-real labels, target balanced datasets, conduct different preprocessing pipelines, or contain auxiliary information unavailable in LIAR's statement-only setting used by this paper.

A separate body of research highlights that statement text is only one possible source of evidence. Raza and Ding utilize a transformer-based early-detection framework for combined news content with social-context features[22]. Meanwhile, FakeNewsNet provides news content together with user engagement and spatiotemporal propagation information[23], while Djenouri et al. incorporated federated learning, user grouping, and convolutional transformers[24]. Similarly, CSI combines article text with temporal user responses and source behavior[25]. These approaches support the value of contextual signals, but they evaluate broader pipelines than the text-only classifiers studied here. The present experiment intentionally excluded metadata and external evidence to isolate the predictive information contained in claim text and ensure that every model family used the same input modality. Thus, only statement text was used as the model input, while the truthfulness label served as the only output or prediction target. Isolating the statement text supports a focused comparison of text-classification pipelines, although it did not represent a complete evidence-based fact-checking system.

Transformer models also introduce substantial computational considerations. DistilBERT attempted to preserve much of BERT's language-representation capability, while also being developed through knowledge distillation as a smaller and faster alternative to BERT[26]. The reduced size makes DistilBERT more accessible for comparative experiments, but it still remains considerably more computationally demanding than sparse linear classifiers, such as TF-IDF-based classifiers like Logistic Regression and Linear SVM.

Accordingly, this study compared four traditional classifiers—Multinomial Naive Bayes, Logistic Regression, Linear SVM, and Decision Tree—with CNN, BiLSTM, and DistilBERT all using the official LIAR training, validation, and test partitions. All listed models used the same statement text, data partitions, and evaluation framework; however, each model family retained the preprocessing and representation required by its architecture. As a result, the study evaluated complete text-classification pipelines over architecture alone. Performance was assessed using accuracy, macro-F1, micro- and macro-AUC, per-class precision, recall, and F1, and within-one accuracy on the ordered truthfulness scale. Bootstrap confidence intervals characterized uncertainty associated with the held-out test sample. Training and inference runtimes documented computational cost. Ultimately, through this combined evaluation, this study examined whether the additional complexity of neural and transformer-based pipelines produced consistent benefits over tuned traditional methods within the specific setting of text-only, six-class LIAR classification.

Methods

Dataset and Preprocessing

This study utilized the LIAR benchmark dataset, a collection of brief political claims, to explore automated identification of misinformation.[4] Each political remark or statement is placed in one of six truthfulness categories arranged along a spectrum, ranging from true to pants-fire. Of the 12,836 entries marked by human reviewers, 10,269 cases are used to train models. Following that phase, a separate batch of 1,284 samples is used for validation. Finally, an untouched group totaling 1,283 statements measures results after learning finishes. Stored as tab-delimited records, each entry contains multiple fields, including the claim, the label, and additional background context. This analysis only used the statement text and the truthfulness label. The distribution of labels across the dataset, shown in Figure 1, highlights that mid-range categories are substantially more frequent than extreme labels like true or pants-fire (an extreme case of falsehood).

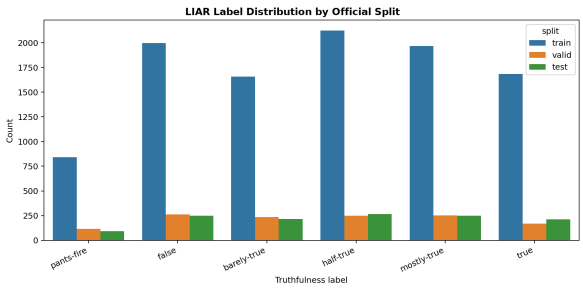

Figure. 1 Distribution of truthfulness labels in the LIAR dataset. The half-true, false, and mostly-true categories contain the most statements, while pants-fire represents the smallest class.

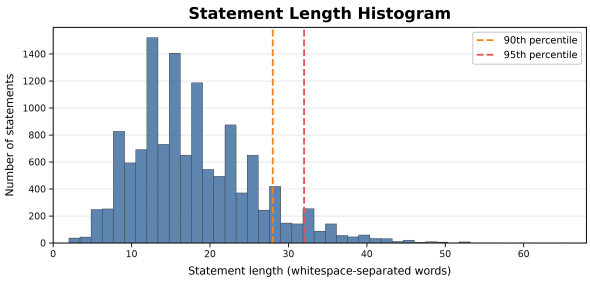

Figure. 2 Distribution of LIAR statement lengths measured using whitespace-delimited word counts.

The LIAR dataset demonstrates visual class imbalance, with the middle truthfulness categories containing higher counts than the border categories. With intermediate categories containing more values, multi-class classification becomes harder, as models may become biased toward larger classes. Thus, macro-averaged metrics were employed throughout the study in order to balance performance assessment across the six categories.

Model-specific tokenization was applied to the statement text. Class labels were encoded as integers ranging from 0 to 5. For traditional machine learning models, statements were converted to numeric feature vectors using term frequency–inverse document frequency (TF–IDF) representations implemented through Scikit-learn's TfidfVectorizer. The TF–IDF settings were tuned separately for each classifier, varying unigram and unigram–bigram features, vocabulary limits from 20,000 to 50,000 terms, minimum document-frequency thresholds of 2 or 3, and standard or sublinear term-frequency scaling. Each resulting TF–IDF matrix was normalized and used as the input feature set for its corresponding classifier.

For the CNN and BiLSTM models, raw text data was tokenized into integer sequences and padded or truncated to a maximum length of 64 tokens. An embedding layer was used to map each token to a dense vector representation before feeding the data into neural network architectures.

To evaluate the suitability of the 64-token maximum length, the statement lengths were measured across all 12,836 LIAR entries using whitespace-delimited word counts. Statements contained an average of 17.9 words and a median of 17 words; the 90th and 95th percentiles were 28 and 32 words, respectively, and only two statements exceeded 64 words, as shown in Figure 2. Because whitespace-delimited words do not correspond exactly to Keras or WordPiece tokens, these statistics support, but do not guarantee, that the selected maximum length preserved most statement content.

Although the same official LIAR train, validation, and test splits were used consistently, individual model family preprocessing pipelines differed due to architectural differences. The traditional machine learning models utilized TF-IDF representations generated from original statement text. However, the neural network models employed tokenized integer sequences padded to a fixed length of 64 tokens, while the DistilBERT required the pretrained DistilBERT tokenizer. The pretrained DistilBERT WordPiece tokenizer (distilbert-base-uncased) automatically performs subword tokenization and generates attention masks. A summary of the preprocessing procedures for each model is provided in Table 1.

For transformer-based modeling, a hyperparameter search was conducted to find an effective DistilBERT configuration using the validation set. By varying maximum sequence length, learning rate, and warmup ratio, various training configurations were evaluated. The two preliminary settings were then evaluated in focused five-epoch runs as Config A and Config B. Their validation macro-F1 values were nearly tied, at 0.2655 and 0.2654, respectively. Config B demonstrated more stable convergence across the focused five-epoch run and therefore was selected as the final configuration before test evaluation. Config B was used for all reported test results. Importantly, the validation and test partitions remained separated throughout the tuning process, preventing information leakage.

Traditional Machine Learning Models

Four traditional machine learning classifiers were implemented as baseline approaches: Multinomial Naive Bayes, Logistic Regression (LR), Linear Support Vector Machine (SVM), and Decision Tree (DT). These models were selected because they are commonly used in text classification tasks and provide interpretable comparisons against the neural and transformer-based models evaluated in this study.

All traditional machine learning models used only the LIAR statement text and truthfulness label. No LIAR metadata fields, such as speaker, party, subject, state, or context, were included. Statements were converted into sparse numerical feature vectors using Scikit-learn's TfidfVectorizer. The vec-

Table 1 Preprocessing procedures used for each model family.

Preprocessing Detail	Traditional ML	CNN / BiLSTM	DistilBERT
Text Input	Statement text only; TF-IDF fitted on training data	Statement text only; tokenizer fitted on training data	Raw statement text only; Pretrained tokenizer
Lowercasing	Yes	Yes	Yes; uncased tokenizer
Punctuation	Not manually stripped; punctuation acts mainly as token boundaries	Filtered by Keras tokenizer	Not stripped before tokenization
Tokenizer	Scikit-learn TfidfVectorizer	Keras Tokenizer	DistilBERT WordPiece tokenizer
Vocabulary	30k max for NB; 50k max for LR/SVM/DT	10,000	30,522
OOV Handling	Unseen terms ignored	Mapped to <OOV>	Subword splitting; [UNK] fallback
Maximum Length	No fixed sequence length	64 tokens	64 tokens
Lemmatization / POS	None	None	None
N-grams	Unigrams and bigrams	None	None
Numbers / URLs / Special Tokens	Numbers $\geq$ 2 characters kept; URLs split; no special tokens	Numbers retained; URLs split; <OOV>	WordPiece; special tokens automatic
Excluded Metadata	subject, speaker, job, state, party, historical counts, context	subject, speaker, job, state, party, historical counts, context	subject, speaker, job, state, party, historical counts, context

No lemmatization or part-of-speech tagging was applied; therefore, their ordering relative to punctuation treatment was not applicable.

torizer was fitted only on the official LIAR training split and then applied to the validation and test splits. The TF–IDF search space included unigram-only and unigram-bigram feature settings, vocabulary limits of 20,000 to 50,000 terms, minimum document-frequency thresholds of 2 or 3, and both standard and sublinear term-frequency scaling. No lemmatization or part-of-speech processing was applied.

All classical models used the official LIAR train, validation, and test partitions: the training split was used to fit the models, the validation split was used for hyperparameter selection, and the held-out test split was used only for final reporting. By evaluating combinations of TF-IDF settings and model-specific parameters on the validation set, the hyperparameters were tuned. Again, the best model was selected based on validation macro-F1 due to the imbalance in the six LIAR truthfulness labels.

Each traditional classifier received tuning within the same validation-based framework as shown in Table 2. With the optimized configurations, the final comparison reflects both classifier behavior and text-representation differences. This procedure ensured that the final test results reflected performance on the official held-out test data.

Deep Learning Models

Three deep learning architectures were developed using TensorFlow/Keras and PyTorch with Hugging Face Transformers: a convolutional neural network (CNN), a bidirectional long short-term memory network (BiLSTM), and a DistilBERT model. The differences in preprocessing are displayed in Table 1; this section focuses on model architecture and training procedure. Candidate CNN, BiLSTM, and DistilBERT configurations evaluated through the official LIAR validation split were ranked by their validation macro-F1 performance. The complete configurations, results, and selected settings can be found in Appendix A.

The CNN architecture employed an embedding setup: 10,000 possible tokens mapped to 100-dimensional vectors. After that came a one-dimensional convolution stage: 128 filters scanned sequences using windows of five elements, each applying ReLU activation. Instead of averaging across positions, maximum values were pulled out globally through a max-pooling layer. After applying dropout (0.5), half the units shut down randomly during training to reduce overfitting. In the final layers of the CNN, fully connected stages processed those features for distinguishing among six categories. One 64-unit dense layer used ReLU activation, and another assigned class probabilities via softmax.

A similar embedding setup served the BiLSTM, though this model introduced a two-way LSTM layer with 64 internal units instead of convolutions. After that, a dropout of 0.5 was followed by a fully connected segment tuned to 64 nodes, concluding with probabilistic classification through a softmax output layer. Both CNN and BiLSTM models were compiled with the Adam optimizer using a learning rate of 0.001 and trained for up to eight epochs with a batch size of 32. Class weights computed from the official training split were applied during CNN and BiLSTM training. Training and validation loss curves, along with validation macro-F1 curves, were retained to document convergence behavior.

For the transformer-based model, DistilBERT was fine-tuned using the pretrained "distilbert-base-uncased" checkpoint. Input text was tokenized using the Hugging Face tokenizer and truncated or padded to a maximum sequence length of 64 tokens. Following the validation-based search summarized in Appendix A, a focused five-epoch run was used

Table 2 Hyperparameter tuning and selected configurations for the traditional machine learning classifiers. The table contains a summary of the model-specific parameters, including the class-imbalance handling method and the final selected TF-IDF configuration before evaluation.

Category	Multinomial Naive Bayes	Logistic Regression	Linear SVM	Decision Tree
Tuned Parameters	Alpha: 0.25, 0.75, 1.0	C: 0.5, 1.0, 2.0; lbfgs; max_iter = 1200	C: 0.5, 1.0, 2.0; max_iter = 6000	max_depth/min_samples_leaf: 20/2, 50/1, 80/4
Class Imbalance Handling	Sample weights	Balanced class weights	Balanced class weights	Balanced class weights
Selected Configuration	alpha = 0.75; TF-IDF unigrams + bigrams; max_features = 30k; min_df = 3; standard TF	C = 2.0; TF-IDF unigrams + bigrams; max_features = 50k; min_df = 2; sublinear TF	C = 0.5; TF-IDF unigrams + bigrams; max_features = 50k; min_df = 2; sublinear TF	max_depth = 80; min_samples_leaf = 4; TF-IDF unigrams + bigrams; max_features = 50k; min_df = 2; sublinear TF

to train Config B to assess convergence. The final DistilBERT model used AdamW optimization with a learning rate of 5×10^{-5}, a batch size of 8, weight decay of 0.01, a warmup ratio of 0.06, gradient clipping with a maximum norm of 1.0, and weighted cross-entropy loss. A linear warmup-and-decay learning-rate scheduler was also applied across the training steps. Intermediate checkpoints were saved, and the checkpoint with the highest validation macro-F1 was used for final test-set evaluation. Experiments were conducted in Google Colab, with the hardware, settings, and procedures summarized in Appendix B.

Training Configuration and Evaluation

For all traditional machine learning models, accuracy and macro-averaged F1 scores were calculated using the same evaluation function to ensure consistent metrics across baselines. For multi-class classification, the CNN and BiLSTM models used categorical cross-entropy loss, while DistilBERT used weighted cross-entropy loss to account for class imbalance. Accuracy, macro-F1, macro-AUC, and micro-AUC were reported for final test performance. Since the LIAR labels formed an ordered truthfulness scale, accuracy could not fully describe the classification, so within-one accuracy was additionally calculated as the proportion of predictions whose encoded class differed from the true class by no more than one level. The ordered encoding was pants-fire = 0, false = 1, barely-true = 2, half-true = 3, mostly-true = 4, and true = 5. Predictions were generated using argmax over predicted class probabilities or output logits. Scores were calculated for per-class precision, recall, and F1.

A combined results table was generated for all models: traditional, neural, and transformer-based. Two non-trained baselines were included: a majority-class baseline labeling all test statements with the most frequent training label (half-true), and an empirical uniform-random prediction set, created for main performance and ordinal error analysis. In addition to model-level metrics, confusion matrices were created for each classifier to visualize correct predictions and misclassification patterns. These class-level outputs were used to support later error analysis and comparison among model families.

Micro-averaged and macro-averaged receiver operating characteristic (ROC) curves and area-under-the-curve (AUC) values were computed using Scikit-learn's one-vs-rest implementation. In this approach, each truthfulness label was treated in turn as the positive class, while the other five labels were treated as the negative class. Label binarization converted the six-class target into six binary indicator columns, allowing a ROC curve and AUC to be calculated for each class. Class labels were binarized, and ROC/AUC calculations used predicted class probabilities for probabilistic models and one-vs-rest decision-function scores for Linear SVM. Both measures are reported in Table 3.

For the deep learning models, CNN, BiLSTM, and DistilBERT, training history was also recorded across epochs. Training loss, validation loss, and validation macro-F1 were plotted to document convergence behavior in order to identify validation performance: improvement, plateau, or decline.

To quantify test-set uncertainty, 95% bootstrap confidence intervals for accuracy and macro-F1 were calculated. Calculated from paired resamples of the final 1,283-example test labels and predictions using the percentile method and random seed 42, the intervals measure uncertainty associated with the finite test sample, not the variation across repeated training runs. DistilBERT Config B used 1,000 resamples; Logistic Regression, Linear SVM, BiLSTM, Naive Bayes, CNN, and Decision Tree used 2,000 bootstrap resamples.

Runtime Measurement

To evaluate computational efficiency across models, runtime measurements were collected during training and inference. Each model was wrapped with timing functions that recorded the total wall-clock time required to complete each model's final training procedure after hyperparameter selection. Timing was implemented using Python's built-in time module, captur-

ing the start and end of each training block.

For traditional machine learning models, the reported runtime measured classifier fitting after TF-IDF feature construction; TF-IDF vectorization time was not included. CNN and BiLSTM runtime calculations included the neural model training time in the runtime calculations across the selected epochs. For DistilBERT, the cumulative epoch runtimes were used to calculate the time for the five-epoch Config B run.

Inference runtime was also measured for the official LIAR test split. The total time required to generate the predictions on the test set was recorded per test statement, allowing for comparison not only by training cost, but also by prediction speed during deployment.

Runtime values were reported in seconds, and per-statement inference time was reported in milliseconds. Performance-runtime trade-offs were used to assess whether improvements in classification performance justified the additional computational cost of neural and transformer-based models. Representing one recorded final run rather than an average across repeated hardware-controlled trials, the reported runtimes describe the observed Google Colab environments and should not be generalized as universal runtime ratios.

Results

Model Performance Comparison

Each model was evaluated on the official LIAR test split using identical partitions. Performance highlights appear in Table 3, covering both classic machine learning and neural network methods through various measures. Logistic Regression held the highest point estimates for accuracy and macro-F1, with DistilBERT Config B trailing closely behind. Linear SVM and BiLSTM also produced similar macro-F1 values, but Decision Tree finished with the lowest non-baseline macro-F1.

The calculated 95% confidence intervals for the leading models overlapped for both accuracy and macro-F1, meaning the small differences among the models should be interpreted as descriptive rankings, not as evidence that one model was statistically superior.

Figure 3 provides a visual summary of the same performance pattern, showing that the strongest models clustered closely in accuracy and macro-F1 rather than separating cleanly by model family.

Training Stability and Convergence

Training stability was examined for the CNN, BiLSTM, and DistilBERT Config B models using the saved loss and validation macro-F1 curves. Figure 4 shows that CNN and BiLSTM both steadily decrease in training loss, while validation loss increases across the later epochs. The increasing gap between

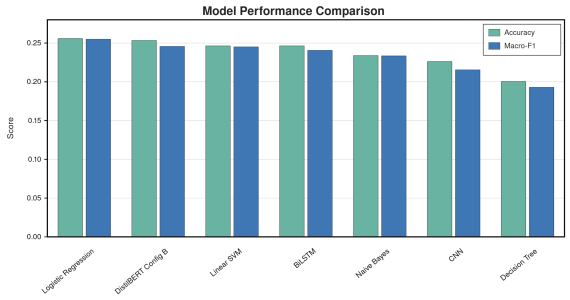

Figure. 3 Model performance comparison of accuracy and macro-F1 across all trained classifiers.

the validation loss and the training loss indicates overfitting pressure: the models continued fitting the training data even as their validation loss became less stable. However, validation macro-F1 did not immediately decline. The CNN's validation macro-F1 continued improving across the later epochs, while the BiLSTM reached its strongest validation macro-F1 around epoch 6 before fluctuating slightly afterward.

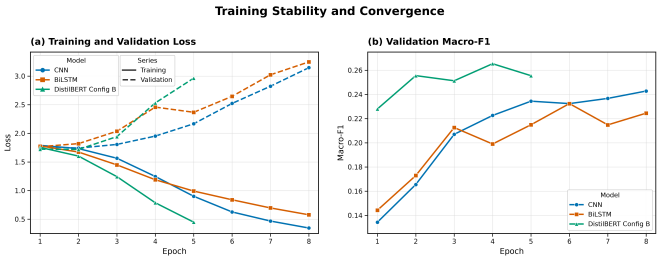

Figure. 4 Training stability and convergence curves for CNN, BiLSTM, and DistilBERT Config B. Panel (a) shows training and validation loss by epoch. Panel (b) shows validation macro-F1 across epochs.

The DistilBERT Config B curves showed a similar divergence between training and validation loss. Similar to the other two models, training loss decreased consistently while validation loss rose across the five-epoch run. However, validation macro-F1 peaked before the final epoch, justifying the decision to select the checkpoint with the highest validation macro-F1.

ROC/AUC and Class-Level Performance

The ROC curves in Figure 5 compare the discriminative ability of all the evaluated models using micro-averaged AUC scores across the six truthfulness classes. Meanwhile, Table 3 additionally reports macro-AUC to give equal weight to each truthfulness class. Despite lower accuracy, DistilBERT Con-

Table 3 Model performance comparison on the LIAR set displaying the model's accuracy and macro-F1 with 95% bootstrap confidence intervals, micro-AUC, macro-AUC, and training/inference runtimes. Micro-AUC pools one-vs-rest decisions, whereas macro-AUC weights all classes equally. Micro-AUC reveals a model's overall discriminative ability across one-vs-rest class decisions. Confidence intervals reflect test-sample uncertainty. The uniform random baseline represents the realized results from an empirical equal-probability prediction set.

Model	Accuracy (95% CI)	Macro-F1 (95% CI)	Micro-AUC	Macro-AUC	Training Runtime (s)	Inference Runtime (s)
Logistic Regression	0.256 [0.231–0.280]	0.255 [0.229–0.279]	0.620	0.607	3.558	0.005
DistilBERT Config B	0.253 [0.230–0.277]	0.246 [0.220–0.271]	0.643	0.618	20,570	153
Linear SVM	0.246 [0.221–0.272]	0.245 [0.219–0.270]	0.605	0.595	0.258	0.003
BiLSTM	0.246 [0.223–0.270]	0.241 [0.217–0.263]	0.610	0.592	289	1.246
Naive Bayes	0.234 [0.210–0.257]	0.233 [0.210–0.256]	0.621	0.615	0.009	0.008
CNN	0.226 [0.202–0.249]	0.215 [0.192–0.238]	0.596	0.575	114	0.584
Decision Tree	0.200 [0.180–0.221]	0.193 [0.172–0.214]	0.538	0.529	5.622	0.010
Uniform Random	0.163 [—]	0.158 [—]	0.500	0.500	—	—
Majority-Class	0.208 [—]	0.057 [—]	0.525	0.500	—	—

fig B had the highest observed micro-AUC of 0.643. Because micro-AUC and macro-AUC summarize ranking performance rather than final hard-label assignments, Table 4 reports per-class precision, recall, and F1 to show how performance varied across the six truthfulness classes. Table 5 then presents exact and within-one accuracy, while Figure 6 illustrates where Logistic Regression and DistilBERT Config B made their final classification errors.

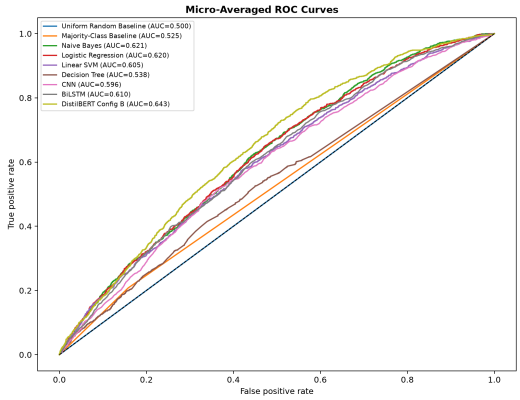

Figure. 5 Micro-averaged ROC curves for all models. Each curve shows the trade-off between true positive rate and false positive rate across all six truthfulness classes. The area under the curve (AUC) indicates discriminative ability.

Ordinal Error Analysis

Because the six LIAR labels form an ordered truthfulness scale, the exact-match accuracy does not distinguish an adjacent-category error from a distant error, justifying the use of within-one accuracy as a supplement. The within-one measure counted predictions that either matched the true label or differed from it by one adjacent category. Table 5 displays both measures.

DistilBERT Config B achieved the highest within-one accuracy at 0.613. Yet, the Majority-Class Baseline also reached a high within-one accuracy of 0.569 because it assigned every statement to the central half-true category; therefore, it is necessary to interpret within-one accuracy alongside the exact accuracy rather than replacing one for the other. The Uniform Random Baseline recorded in Table 5 has an empirical random-prediction accuracy of 0.163 and a within-one accuracy of 0.454.

Figure 6 compares the row-normalized confusion matrices for Logistic Regression and DistilBERT Config B. Despite Logistic Regression displaying a slightly higher concentration along the main diagonal, DistilBERT Config B had more concentrated predictions in neighboring categories rather than the most extreme distance classes. In Figure 6, when the true label was true, Logistic Regression assigned 6.6% cases to pants-fire, while DistilBERT Config B assigned only 0.9%, reducing extreme confusion. The confusion matrices expand on the previous within-one accuracy by showing exactly where each model's misclassifications occur.

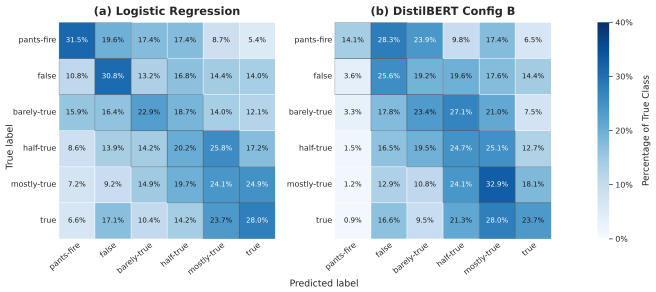

Figure. 6 Row-normalized confusion matrices for (a) Logistic Regression and (b) DistilBERT Config B. Each cell reports the percentage of a true class assigned to each predicted class. Cells on the diagonal indicate exact classification, while the diagonal and the immediately adjacent cells indicate within-one accuracy.

Table 4 Each cell reports per-class precision, recall, and F1 scores.

Class	Naive Bayes	Logistic Regression	Linear SVM	Decision Tree	CNN	BiLSTM	DistilBERT Config B
pants-fire	0.165 / 0.391 / 0.232	0.200 / 0.315 / 0.245	0.197 / 0.272 / 0.228	0.105 / 0.196 / 0.137	0.149 / 0.163 / 0.155	0.178 / 0.261 / 0.211	0.342 / 0.141 / 0.200
false	0.269 / 0.180 / 0.216	0.341 / 0.308 / 0.324	0.312 / 0.288 / 0.299	0.240 / 0.236 / 0.238	0.272 / 0.348 / 0.305	0.293 / 0.340 / 0.315	0.268 / 0.256 / 0.262
barely-true	0.239 / 0.224 / 0.231	0.251 / 0.229 / 0.240	0.260 / 0.220 / 0.238	0.201 / 0.252 / 0.224	0.192 / 0.210 / 0.201	0.212 / 0.229 / 0.220	0.228 / 0.234 / 0.231
half-true	0.269 / 0.184 / 0.218	0.234 / 0.202 / 0.217	0.225 / 0.206 / 0.215	0.197 / 0.228 / 0.212	0.222 / 0.191 / 0.205	0.225 / 0.191 / 0.206	0.230 / 0.247 / 0.238
mostly-true	0.223 / 0.237 / 0.230	0.237 / 0.241 / 0.239	0.225 / 0.233 / 0.229	0.241 / 0.141 / 0.178	0.222 / 0.145 / 0.175	0.280 / 0.161 / 0.204	0.262 / 0.329 / 0.292
true	0.252 / 0.299 / 0.273	0.253 / 0.280 / 0.266	0.244 / 0.280 / 0.260	0.210 / 0.142 / 0.169	0.237 / 0.265 / 0.251	0.261 / 0.318 / 0.286	0.267 / 0.237 / 0.251

Table 5 Exact and within-one accuracy across the evaluated models and baselines.

Model	Exact Accuracy	Within-One Accuracy
Uniform Random Baseline	0.163	0.454
Majority-Class Baseline	0.208	0.569
Naive Bayes	0.234	0.568
Logistic Regression	0.256	0.584
Linear SVM	0.246	0.571
Decision Tree	0.200	0.509
CNN	0.226	0.559
BiLSTM	0.246	0.551
DistilBERT Config B	0.253	0.613

Runtime Measurement

Timing data supplements accuracy measures to assess how efficiently each model runs. In Table 3 and Figure 7, fitting and training durations reflect the recorded Google Colab setup found in Appendix B. Naive Bayes and Linear SVM took only 0.009 and 0.258 seconds for classifier fitting, while Logistic Regression needed 3.558 seconds and Decision Tree, 5.622 seconds. For the neural networks, CNN took 114 seconds to train, and BiLSTM, 289 seconds. DistilBERT Config B took 20,570 seconds (about 5.7 hours) to train the model for 5 epochs.

A similar tendency can be observed during the inference stage. Specifically, Linear SVM and Logistic Regression classified the whole test set in 0.003 and 0.005 seconds, compared to 0.584 and 1.246 seconds for CNN and BiLSTM, respectively. Meanwhile, DistilBERT took 153 seconds, i.e. 119 milliseconds per test statement.

Performance-Runtime Comparison

Figure 8 combines macro-F1 and classifier-fitting runtime for the traditional models and training time for the neural and transformer models. In particular, Logistic Regression and Linear SVM were shown to achieve a good balance between low computational overhead and high macro-F1, while DistilBERT Config B required substantially longer training time. The implications are covered in the Discussion.

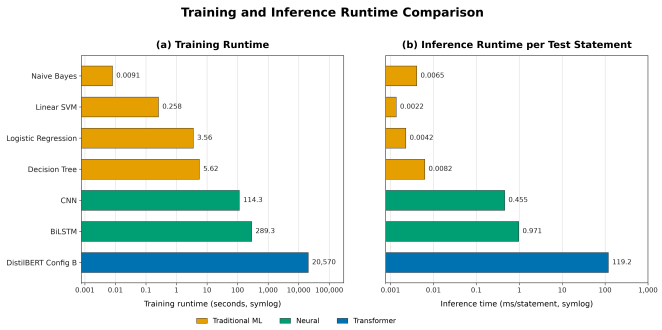

Figure. 7 Fitting/training and inference runtime comparison across the evaluated models. Panel (a) presents total training runtime in seconds, while panel (b) presents inference time per test statement in milliseconds. Symlog scaling is used to display the large differences between traditional, neural, and transformer-based models.

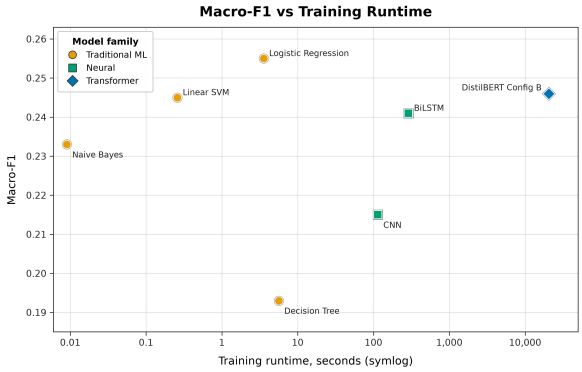

Figure. 8 Macro-F1 versus fitting/training runtime trade-off across all trained models. Each point represents a model's final test macro-F1 plotted against its total training runtime. The runtime axis is displayed on a symmetric logarithmic scale.

Discussion

This study examined whether greater model complexity improved text-only, six-class classification on the LIAR dataset. The results did not clearly identify one model family as consistently superior compared to the rest. Logistic Regression

produced the highest point estimates for exact accuracy and macro-F1, whereas DistilBERT Config B produced the highest micro-AUC, macro-AUC, and within-one accuracy. Because the leading models were closely grouped and their bootstrap confidence intervals overlapped, the rankings should be interpreted descriptively rather than as evidence of statistical superiority. More importantly, even the highest-accuracy model misclassified nearly three-quarters of the test statements. Therefore, none of the evaluated systems can reliably function alone as an independent political fact-checker with the current training. A more realistic function for these models would be to support human review.

Within this experiment, Logistic Regression and Linear SVM remained competitive with the neural and transformer pipelines. Sparse TF-IDF representations may suit short claims because they directly capture recurring words and phrases without requiring task-specific contextual representations to be learned from a modest dataset like LIAR. However, the study did not isolate the effects of dataset size, statement length, or representation. Since the model families utilized different preprocessing, features, and optimization procedures, the comparison evaluated complete text-classification pipelines. Therefore, the findings do not establish that traditional methods are universally superior, instead, they are consistent with earlier studies showing that LIAR is difficult and that relative performance varies across datasets and modeling pipelines[4,6].

DistilBERT's results demonstrate why exact accuracy alone does not fully describe model behavior. Its higher AUC values indicated stronger score-based ranking, while its higher within-one accuracy showed that more predictions fell in the correct or an adjacent truthfulness category. Its confusion matrix also contained fewer extreme cross-scale errors in some cases. These properties could be useful for prioritizing claims for human review, but that workflow and the practical consequences of adjacent versus extreme errors were not evaluated. Despite the benefits, within-one accuracy must also remain supplementary, as the Majority-Class Baseline achieved a comparatively high score by always predicting the central half-true category. Thus, ordinal proximity provides more insight about error distance but cannot replace exact accuracy, macro-F1, per-class metrics, or confusion-matrix analysis.

The convergence curves showed overfitting pressure during later epochs, but validation-based model selection reduced its influence on the reported test results. CNN and BiLSTM restored the weights with the strongest validation macro-F1, while DistilBERT Config B used its best saved checkpoint from epoch 4. Traditional models also had lower recorded fitting and inference times, although these comparisons remain environment-specific because the runs were not repeated under identical hardware and traditional-model timing excluded TF-IDF construction.

Several limitations restrict the conclusions. The study used one imbalanced dataset of short political statements and excluded speaker, party, subject, context, historical credibility, and external evidence. Though this decision supported a focused text-only comparison, it removed information used in professional fact-checking. The six labels also contain subtle and potentially subjective boundaries. Bootstrap intervals represented finite-test-sample uncertainty rather than variation across repeated training runs, and no paired significance tests were conducted. The low absolute performance is consistent with the difficulty of assigning six truthfulness levels from context-limited claims, but the study cannot isolate a single cause. These findings should not be generalized to longer documents, larger datasets, metadata-enhanced systems, evidence-retrieval systems, or other misinformation domains.

The study's objective was therefore met in a qualified manner: greater architectural complexity did not produce a clear overall advantage in this text-only LIAR setting, but it affected different dimensions of performance. Future research should evaluate additional datasets, metadata and evidence retrieval, ordinal-aware loss functions, repeated-seed experiments, paired significance tests, calibration, and complete end-to-end timing. To reach a better balance between prediction and computation, future research may utilize hybrid lexical-contextual models. Overall, findings suggest the most appropriate model is the model whose error profile, computational requirements, and intended role best match the application.

Acknowledgements

I would like to express my deepest gratitude to my research mentor, Sophia Chen, for her guidance throughout the project.

References

1 H. Allcott, M. Gentzkow. Social media and fake news in the 2016 election. Journal of Economic Perspectives. Vol. 31, pg. 211–236, 2017. https://doi.org/10.1257/jep.31.2.211.

2 S. Vosoughi, D. Roy, S. Aral. The spread of true and false news online. Science. Vol. 359, pg. 1146–1151, 2018. https://doi.org/10.1126/science.aap9559.

3 K. Shu, A. Sliva, S. Wang, J. Tang, H. Liu. Fake news detection on social media: a data mining perspective. ACM SIGKDD Explorations Newsletter. Vol. 19, pg. 22–36, 2017. https://doi.org/10.1145/3137597.3137600.

4 W. Y. Wang. "Liar, liar pants on fire": a new benchmark dataset for fake news detection. Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). pg. 422–426, 2017. https://doi.org/10.18653/v1/P17-2067.

5 J. Y. Khan, M. T. I. Khondaker, S. Afroz, G. Uddin, A. Iqbal. A benchmark study of machine learning models for online fake news detection. Machine Learning with Applications. Vol. 4, pg. 100032, 2021. https://doi.org/10.1016/j.mlwa.2021.100032.

6 J. Alghamdi, Y. Lin, S. Luo. A comparative study of machine learning and deep learning techniques for fake news detection. Information. Vol. 13, pg. 576, 2022. https://doi.org/10.3390/info13120576.

7 I. Ahmad, M. Yousaf, S. Yousaf, M. O. Ahmad. Fake news detection using machine learning ensemble methods. Complexity. Vol. 2020, pg. 8885861, 2020. https://doi.org/10.1155/2020/8885861.

8 M. Sokolova, G. Lapalme. A systematic analysis of performance measures for classification tasks. Information Processing & Management. Vol. 45, pg. 427–437, 2009. https://doi.org/10.1016/j.ipm.2009.03.002.

9 F. A. Ozbay, B. Alatas. Fake news detection within online social media using supervised artificial intelligence algorithms. Physica A: Statistical Mechanics and its Applications. Vol. 540, pg. 123174, 2020. https://doi.org/10.1016/j.physa.2019.123174.

10 S. Hakak, M. Alazab, S. Khan, T. R. Gadekallu, P. K. R. Maddikunta, W. Z. Khan. An ensemble machine learning approach through effective feature extraction to classify fake news. Future Generation Computer Systems. Vol. 117, pg. 47–58, 2021. https://doi.org/10.1016/j.future.2020.11.022.

11 A. Choudhary, A. Arora. Linguistic feature based learning model for fake news detection and classification. Expert Systems with Applications. Vol. 169, pg. 114171, 2021. https://doi.org/10.1016/j.eswa.2020.114171.

12 T. Jiang, J. P. Li, A. U. Haq, A. Saboor, A. Ali. A novel stacking approach for accurate detection of fake news. IEEE Access. Vol. 9, pg. 22626–22639, 2021. https://doi.org/10.1109/ACCESS.2021.3056079.

13 R. K. Kaliyar, A. Goswami, P. Narang. FakeBERT: fake news detection in social media with a BERT-based deep learning approach. Multimedia Tools and Applications. Vol. 80, pg. 11765–11788, 2021. https://doi.org/10.1007/s11042-020-10183-2.

14 J. A. Nasir, O. S. Khan, I. Varlamis. Fake news detection: a hybrid CNN–RNN based deep learning approach. International Journal of Information Management Data Insights. Vol. 1, pg. 100007, 2021. https://doi.org/10.1016/j.jjimei.2020.100007.

15 R. K. Kaliyar, A. Goswami, P. Narang, S. Sinha. FNDNet—a deep convolutional neural network for fake news detection. Cognitive Systems Research. Vol. 61, pg. 32–44, 2020. https://doi.org/10.1016/j.cogsys.2019.12.005.

16 R. K. Kaliyar, A. Goswami, P. Narang. DeepFakE: improving fake news detection using tensor decomposition-based deep neural network. The Journal of Supercomputing. Vol. 77, pg. 1015–1037, 2021. https://doi.org/10.1007/s11227-020-03294-y.

17 M. Choudhary, S. S. Chouhan, E. S. Pilli, S. K. Vipparthi. BerConvoNet: a deep learning framework for fake news classification. Applied Soft Computing. Vol. 110, pg. 107614, 2021. https://doi.org/10.1016/j.asoc.2021.107614.

18 M. Samadi, M. Mousavian, S. Momtazi. Deep contextualized text representation and learning for fake news detection. Information Processing & Management. Vol. 58, pg. 102723, 2021. https://doi.org/10.1016/j.ipm.2021.102723.

19 D. Mouratidis, M. N. Nikiforos, K. L. Kermanidis. Deep learning for fake news detection in a pairwise textual input schema. Computation. Vol. 9, pg. 20, 2021. https://doi.org/10.3390/computation9020020.

20 A. M. Ali, F. A. Ghaleb, B. A. S. Al-Rimy, F. J. Alsolami, A. I. Khan. Deep ensemble fake news detection model using sequential deep learning technique. Sensors. Vol. 22, pg. 6970, 2022. https://doi.org/10.3390/s22186970.

21 M. Farokhian, V. Rafe, H. Veisi. Fake news detection using dual BERT deep neural networks. Multimedia Tools and Applications. Vol. 83, pg. 43831–43848, 2024. https://doi.org/10.1007/s11042-023-17115-w.

22 S. Raza, C. Ding. Fake news detection based on news content and social contexts: a transformer-based approach. International Journal of Data Science and Analytics. Vol. 13, pg. 335–362, 2022. https://doi.org/10.1007/s41060-021-00302-z.

23 K. Shu, D. Mahudeswaran, S. Wang, D. Lee, H. Liu. FakeNewsNet: a data repository with news content, social context, and spatiotemporal information for studying fake news on social media. Big Data. Vol. 8, pg. 171–188, 2020. https://doi.org/10.1089/big.2020.0062.

24 Y. Djenouri, A. N. Belbachir, T. P. Michalak, G. Srivastava. A federated convolution transformer for fake news detection. IEEE Transactions on Big Data. Vol. 10, pg. 214–225, 2024. https://doi.org/10.1109/TBDATA.2023.3325746.

25 N. Ruchansky, S. Seo, Y. Liu. CSI: a hybrid deep model for fake news detection. Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. pg. 797–806, 2017. https://doi.org/10.1145/3132847.3132877.

26 V. Sanh, L. Debut, J. Chaumond, T. Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv. arXiv:1910.01108, 2019. https://doi.org/10.48550/arXiv.1910.01108.

Appendix A. Deep-Model Hyperparameter Search

Table A1 All CNN & BiLSTM trials used early stopping based on validation macro-F1 with patience = 2. Trial 1 was used for the final reported CNN & BiLSTM model. All DistilBERT trials used batch size 8, weight decay 0.01, and gradient clipping at 1.0. Config B was used for the final reported test results.

(a) CNN

Parameter	Trial 1*	Trial 2	Trial 3
Vocabulary	10,000	20,000	15,000
Maximum length	64	96	80
Embedding dimension	100	128	100
Filters	128	128	96
Kernel size	5	3	5
Dropout	0.50	0.40	0.50
Dense units	64	96	64
Learning rate	0.001	0.0005	0.001
Batch size	32	32	64
Maximum epochs	8	8	7
Best validation macro-F1	0.2428	0.2332	0.2346

(b) BiLSTM (continued table)

Parameter	Trial 1*	Trial 2	Trial 3
Vocabulary	10,000	20,000	15,000
Maximum length	64	96	80
Embedding dimension	100	128	100
LSTM units	64	64	96
Dropout	0.50	0.40	0.50
Dense units	64	96	64
Learning rate	0.001	0.0005	0.0008
Batch size	32	32	64
Maximum epochs	8	8	7
Best validation macro-F1	0.2324	0.2169	0.2161

(c) DistilBERT

Parameter	Trial 1	Trial 2	Config A	Config B*
Maximum length	96	64	96	64
Learning rate	2×10^{-5}	5×10^{-5}	2×10^{-5}	5×10^{-5}
Epochs	3	3	5	5
Warmup ratio	0.10	0.06	0.10	0.06
Early stopping	Patience 1	Patience 1	None	None
Best validation macro-F1	0.2674	0.2666	0.2655	0.2654

Appendix B. Experimental Environment and Reproducibility

Table B1 Verified hardware, software, seed, validation, and stopping settings for the final model runs.

Item	Traditional ML	CNN/BiLSTM	DistilBERT Config B
Platform / hardware	Google Colab; CPU only; x86_64 (exact model not recorded); 1 physical / 2 logical CPUs; 12.67 GB RAM; no GPU	Same Colab session as traditional ML; CPU only; 1 physical / 2 logical CPUs; 12.67 GB RAM; no GPU	Separate Colab session; CPU-only PyTorch build; x86_64 processor (model/count not recorded); 12.67 GB RAM; no GPU used
Software	Python 3.12.13; scikit-learn 1.6.1	Python 3.12.13; TensorFlow 2.20.0; scikit-learn 1.6.1	Python 3.12.13; PyTorch 2.11.0+cpu; Transformers 5.10.2; scikit-learn 1.6.1
Random-seed control	Global seed 42; SVM/DT random_state = 42; LR/NB had no model-specific seed	Global seed 42; CNN and BiLSTM were not reseeded separately	Python, NumPy, PyTorch, and CUDA seed calls = 42
Early stopping	None	Validation macro-F1; patience = 2; min_delta = 1e-4; best weights restored; both completed 8 epochs	Preliminary trials: patience = 1; Config B: none; all 5 epochs completed
Checkpoint / selection	Highest-validation-macro-F1 candidate retained for each classifier	Best validation-macro-F1 weights restored in memory	Best checkpoint loaded for test: epoch 4, validation macro-F1 = 0.2654