LBC: Language-Based-Classifier for Out-Of-Variable Generalization

Anonymous Author(s)

ABSTRACT

Large Language Models (LLMs) have performed well in many natural language process (NLP) tasks. Recently, LLMs have also been used to analyze tabular data. Although LLMs have had great success in natural language processing tasks such as text classification, their use in tabular data classification has been limited due to their inferior performance compared to traditional machine learning models (TMLs) such as XGBoost. However, LLMs have the potential for tabular data classification tasks because they consider the context between variables based on pre-trained knowledge. This implies that LLMs can interpret the context of data that is typically difficult to learn due to the large number of missing values in tabular data or new variables not seen in training. We refer to classification tasks in these situations as out-of-variable (OOV) tasks. We propose a methodology called Language-Based-Classifier (LBC), which has the advantage of solving the OOV tasks in tabular data classification, which is distinct from TMLs. LBC's strength in handling OOV tasks comes from its unique approach to tabular data classification. Firstly, the method of converting tabular data into natural language prompts allows LBC to seamlessly and intuitively handle OOVs for inference. Secondly, the interpretation of OOVs using LBC's pre-trained knowledge base contributes to increasing the probability of the correct answer class. Building on these structural strengths, LBC employs three key methodological strategies: 1) Categorical Changes to adjust data to better fit the model's understanding, 2) Advanced Order and Indicator to enhance data representation to the model, and 3) the use of a Verbalizer to map logit scores to classes during inference to generate model predictions. These strategies, combined with the inherent strengths of LBC, emphasize the model's ability to effectively handle OOV tasks. We empirically and theoretically validate the superiority of LBC. LBC is the first study to apply an LLM-based model to OOV tasks. The source code is at https://github.com/ASDASDanonymous/Language-Based-Classifier-forOOVtasks.

CCS CONCEPTS

• **Computing methodologies** → **Information extraction**.

KEYWORDS

Large Language Model, Tabular Data, Classification, Out of Variable, Verbalizer, Low-Rank Adaption

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

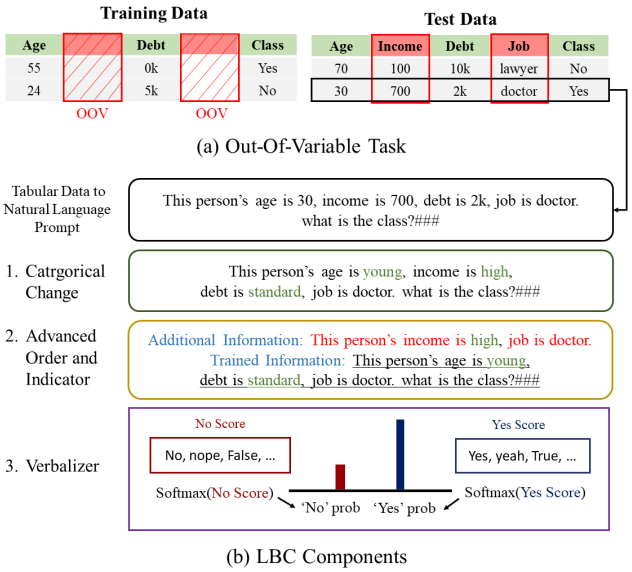

Figure 1: (a) Out-Of-Variable (OOV) task. The variables that were not present in the training data appear in the test data. (b) Key components of LBC to increase performance in OOV tasks. Categorical Change refines data to make it easier for LBC to interpret. Advanced Order and Indicator enhances the prompts that feed into LBC. Verbalizer aggregates the probabilities for a particular class scattered throughout the Logit score and maps them to a specific class. This makes LBC a classifier rather than a language model.

ACM Reference Format:
Anonymous Author(s). 2018. LBC: Language-Based-Classifier for Out-Of-Variable Generalization. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX).* ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The development of language models (LMs) is a major achievement in natural language processing. From early language models, through recurrent neural networks (RNNs) [19], long short-term memory (LSTM) [12], to models based on transformer [30], LMs have continued to evolve and have found success in a variety of natural language processing tasks. Furthermore, these transformer-based Large Language Models (LLMs) [3, 6, 24, 31] have emerged. LLMs consider a wide range of contexts based on extensive pre-trained knowledge and, with fine-tuning, are powerful for many tasks. Recently, applying these LLMs to tabular data has been attempted. Language-Interfaced-Fine-Tuning (LIFT) [7] looked at how well LLMs perform on tabular data tasks, keeping the structure of the LLMs as unchanged as possible. On this basis, we verify

that LLM's pre-trained knowledge can be utilized to solve OOV tasks, which are tasks where a new variable appears in the test that was not learned in training. We propose a new model called Language-Based-Classifier (LBC) to solve the OOV tasks.

In the real world, models often face training restrictions due to various constraints, highlighting the importance of OOV tasks. For instance, in healthcare, data transfer between institutions is impeded by privacy concerns, regulatory limits, or technical barriers. A model trained with patient data from Hospital A cannot directly access or learn from data at Hospital B. However, when deployed at Hospital B, this model must adapt to new variables or unseen data characteristics specific to Hospital B's patient population or data collection practices. Moreover, the relevance of certain data characteristics can shift significantly over time. For example, certain biomarkers or symptoms previously overlooked in medical research or data collection might become crucial for diagnosis [1]. If these were previously underestimated and excluded from the training dataset, they become OOV.

We argue that LBC has strengths in handling OOV tasks, and our rationale is as follows. First, converting tabular data to natural language prompts is intuitive, flexible, and easy. This transformation significantly simplifies the handling of OOVs, allowing us to handle variables that might not have been discovered during training seamlessly, overcoming a common limitation of TMLs. Second, our approach leverages the extensive pre-trained knowledge built into LLMs. Unlike TMLs, which can struggle with data points or scenarios that are not present in the training set, LLMs leverage their inherent knowledge. We empirically verified that LBC can use OOVs to increase the probability of the correct answer class, based on pre-trained knowledge.

In tabular data classification, the output text of LLM is simply the classifier prediction in the previous work [7]. The output text of LLM is highly variable, and it uses a probabilistic sampling method to account for randomness and natural language output generation. We identified the potential for performance improvement by focusing on logit scores instead of the output text of LLM to determine LLM's predictions. We utilize a mapping tool called verbalizer [16]. By applying verbalizer to the logit score of the LLM, we can map the results of the LLM's thought process to the class score we want.

We used the LoRA [15] to fine-tune the classifier. It has been shown that fine-tuning with LoRA can completely approximate an arbitrary target model above a certain rank [34]. We theoretically prove that our model also approximates an arbitrary classifier up to a certain rank.

To the best of our knowledge, LBC is the first study to apply an LLM-based classifier to solve the OOV tasks, and we validate LBC's superiority empirically and theoretically.

- **Problem** Develop a model that can handle OOV tasks, where new variables not learned in training need to be interpreted at test time. OOV tasks are common in real-world scenarios, and despite the high performance of TMLs, they do not have the capability to handle them effectively.
- **Solution:** We propose a Language-Based-Classifier (LBC) that leverages the inherent ability of LLMs to interpret OOVs based on their vast prior knowledge. The LBC is designed to solve the OOV task by applying three structural improvements to LLM.

2 RELATED WORKS

2.1 Tabular Data Analysis with LLMs

While these LLMs are primarily focused on processing text-based data, recently, these models have begun to be applied to tabular data analysis. One such effort is LIFT [7]. Without modifying the existing LLM structure, LIFT converts tabular data into natural language prompts fed into the LLM, which then analyzes the data using the model's output. While this method did not outperform existing TMLs, such as XGBoost [5] or Decision Tree, it was comparable and opened up new directions for applying LLM to tabular data analysis. Research like LIFT is expanding the use of LLMs and exploring new possibilities for future applications in tabular data analysis. We believe in the potential of LLMs in tabular data analysis based on their performance and several additional grounds, including contextual understanding, OOV tasks, and ease of prompt conversion.

2.2 Out Of Variable

Out-of-Variable Generalization [10] introduces a method aimed at forecasting scenarios where two domains are concurrently observed, addressing situations pertaining to domains not observed in tandem. The study on Missing Data [8, 26] delves into circumstances where a covariate vanishes at a specific data point, exploring the impact of such disappearances on data analysis and interpretation. Marginal Problems research [9] focuses on examining the joint distribution of data originating from diverse sources, aiming to understand how data from different contexts can be integrated and analyzed collectively. Our approach utilizes the pre-trained knowledge embedded within Large Language Models (LLMs) to tackle Out-of-Variable (OOV) tasks in a manner distinct from previous methodologies. We argue that LLMs demonstrate the capability to perform inference based on their pre-trained knowledge, effectively managing the introduction of new variables into the analysis. This assertion underscores the adaptability and potential of LLMs in processing and understanding new, previously unseen information within various domains.

2.3 Classification in tabular data

The classification task of tabular data has been explored in several studies. VIME [33] improved their classification model's performance by introducing feature vector estimation and mask vector estimation processes. SuperTML [28] tries to apply tabular data to a Convolutional Neural Networks (CNNs). XGBoost [5] utilizes Gradient Tree Boosting to gain high classification performance. MLP and ResNet [11] can be used to classify tabular data. In TabTransformer [18], multi-layer transformers and MLP are combined. TabPFN [14] aims to reduce inference time using transformers. Additional models are presented in [2]. We also focus on the tabular data classification task.

2.4 Verbalizer

Verbalizer is a mechanism for mapping the various output forms from an LLM to specific classes [17, 27]. Verbalizer contributed to reducing subjective bias in LLM by using a knowledge base to leverage diverse and comprehensive label words. It is also said that the noise of label words in classification can also be improved with a verbalizer. We argue that even in tabular data classification, we need a particular way to map the output of an LLM to the output of a classifier and that we should apply a verbalizer to it.

2.5 Low-Rank Adaption

Low-rank Adaption has emerged as an important innovation in adapting pre-trained models to specific tasks without extensive re-training of the entire model. LoRA [15] introduces a new approach to fine-tuning large pre-trained models. Instead of updating the whole parameter set, LoRA strategically modifies a small subset of the model's weights through a low-rank matrix. This method allows pre-trained models to adapt efficiently while maintaining their original structure and strengths. Furthermore, the high generalization ability of the fine-tuned model using LoRA is verified theoretically [34]. We also use LoRA to construct an advanced classifier with high generalization ability.

3 PRELIMINARY

3.1 Basic Dataset Conversion

This section describes the process of converting tabular data into prompts for input to LBC. The core of our model is based on the Frozen Pre-Trained Model, LLM. Therefore, how to convert tabular data into prompts is an important issue. Let an instance of tabular data with n features be represented as follows:

$$[[V_1 : x_1], [V_2 : x_2], .., [V_n : x_n], .., [V_N : x_N], [\text{class} : y]] \quad (1)$$

where V_n is nth variable name, x_n is nth variable value. We need a method for the LLM to clearly distinguish between the variables in this dataset as prompts and the class as the output. This involves creating a conversion technique that clearly marks the end of the prompt and the beginning of the response while ensuring that the answer isn't overly lengthy. Therefore, we base our conversions as follows:

prompt: V_1 is x_1, V_2 is $x_2, \ldots, V_N$ is x_N. what is the class? ###

answer: y@@@

This is the scheme developed by OpenAI [22]. The 'prompt' is the input to be fed into LBC, and the 'answer' is the label for the data instance. The '###' helps LBC recognize the end of the prompt, and the '@@@' helps LBC not formulate the answer as a sentence and only output the class. Using this approach, we can clearly define the input fed to LBC and structure our training and inference.

3.2 Fine-tuning LLM

Feeding the converted prompts into LBC yields a vector of vocabulary sizes, which is logit for each word in the vocabulary. We use this logit to fine-tune the LLM. Let *Logit* be the logit vector for a single input vector. During Fine-tuning, L obtained from the model is used to compute the loss against the true labels. Let *Label* be the one-hot encoded vector of the true label for the input. The loss is calculated using a loss function J defined as:

$$J(Logit, Label) = \text{CE}(Logit, Label)$$

where CE is cross-entropy loss function. After calculating the loss, the model's parameters are updated using an optimizer, typically through gradient descent. The update rule in gradient descent can be described as follows:

$$\theta \leftarrow \theta - \eta \nabla_\theta J$$

where θ is the model's parameters, η is the learning rate, and $\nabla_\theta J$ is the gradient of the loss with respect to the model parameters.

3.3 The Structure of LBC Blocks

We fine-tuned a pre-trained LLM, GPT-J 6B [31] as a classifier in our experiments. The following steps **1** through **3** constitute a block, and our model stacks a total of N blocks. Let X be a single input vector, the tokenized prompt. We used LoRA [15] for all of our adapters.

3.3.1 Layernorm. When X is fed into the model, the layernorm is applied to X. $LN(X) = \gamma \frac{X-\mu}{\sqrt{\sigma^2+\epsilon}} + \beta$. Where μ is the mean, σ^2 is the variance, ϵ is a small constant for stability, and γ and β are the trainable parameters.

3.3.2 Multi Head Attention and Feed Forward Network. Let X_1 be the $LN(X)$ derived in step 3.3.1. The Attention mechanism follows the basic structure of the Transformer. Define Q, K, and V as follows:

$$Q = \text{RoPE}(X(W_Q + A_{W_Q}))$$
$$K = \text{RoPE}(X(W_K + A_{W_K}))$$
$$V = X(W_V + A_{W_V})$$

where RoPE is Rotary Positional Encoding, and W_Q, W_K, W_V are the Q, K, and V weights, respectively, and A_{W_Q}, A_{W_K}, and A_{W_V} are the Q, K, and V adapters, respectively. Using the defined Q, K, and V, calculate the Attention value $X_{\text{attention}}$ as follows:

$$X_{\text{attention}} = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where softmax is the softmax function, d_k is the dimension of K. The Feed Forward Network (FFN) layer of LBC consists of a linear transform and an activation. The activation can be a function such as ReLU or GELU. The structure of the FFN can be organized as follows: The input X first passes through the first linear layer with an adapter, where $X_{\text{FFN}} = \text{Activation}((W_{\text{FFN}} + A_{W_{\text{FFN}}})X + b_{\text{FFN}} + A_{b_{\text{FFN}}})$.
The outputs of step 3.3.2 are calculated as follows:

$$X_2 = X_1 + X_{\text{attention}} + X_{\text{FFN}}$$

3.3.3 Multilayer Perceptron. Let X_2 be the output of the step 3.3.2. X_2 is passed through a multilayer perceptron (MLP) to become the final hidden state of a block. First, X_2 is transformed to increase the output dimension.

$$FC_{\text{in}}(X_2) = \text{ReLU}(X_2(W_{\text{in}} + A_{W_{\text{in}}}) + b_{\text{in}} + A_{b_{\text{in}}})$$

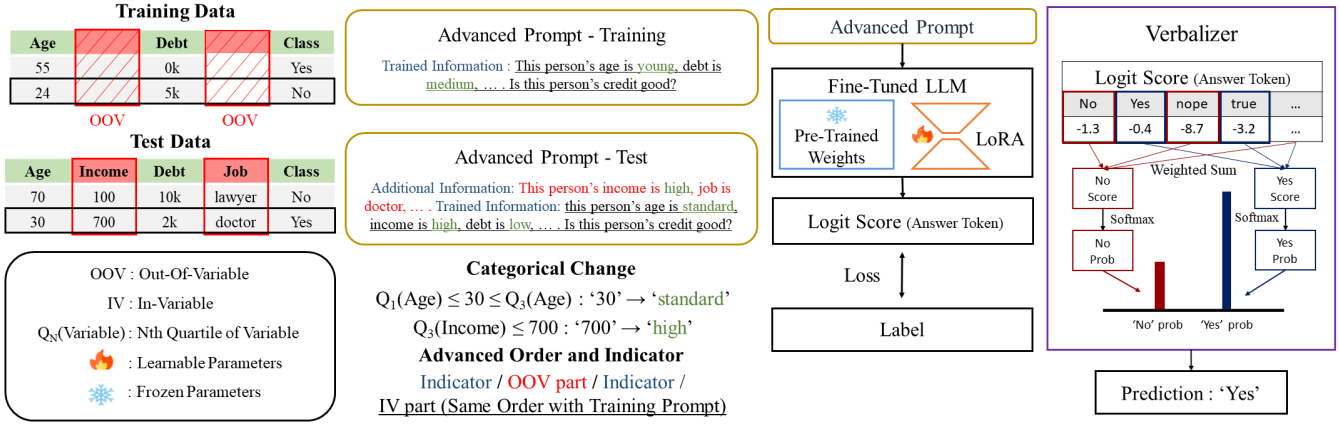

Figure 2: The overall process of an LBC performing an OOV task. LBC transforms tabular data into advanced prompt (AP) utilizing strategies that are 1) Categorical Change and 2) Advanced Order and Indicator. These APs are then input into an LLM that has been fine-tuned with a LoRA adapter, to derive a logit score for the answer token. This logit score is assessed against the label to calculate loss, and during inference, the model prediction is generated by mapping the logit score to a class via a 3) Verbalizer.

where W_{in}, $A_{W_{\text{in}}}$, and b_{in} are the weight of FC_{in} layer, the adapter of FC_{in} layer, and the bias of FC_{in} layer, respectively. Then, activation functions and dropout are applied.

$$\text{GELU}_{\text{Out}} = \text{GELUActivation}(FC_{\text{in}}(X_2))$$
$$\text{DropoutGELU} = \text{Dropout}(\text{GELU}_{\text{Out}})$$
$$Y = FC_{\text{out}}(\text{DropoutGELU})$$

where FC_{out} is the final linear layer. This result is then subjected to a transformation that compresses the output dimension and becomes the output of the MLP.

3.3.4 Logit. Here's how to get the Logit. Let X_3 be the output of the final block's previous step 3.3.3. The Logit of vocab size is as follows:

$$Logit = \text{LT}_{\text{v}}(X_3)$$

where LT is a linear transformation, v is vocab size.

3.4 Prediction of LLM-based tabular data classification

The previous approaches to LLM-based tabular data classification tasks [7] rely on directly comparing the output text generated by the model with class texts such as 'no' or 'yes.' In this approach, if the prediction is an exact match, it is classified with the corresponding class text. Conversely, if the output text differs, the model's prediction is marked as 'None' and automatically classified as incorrect. For example, if the model produces a result of 'yes' for a question with an answer class of 'Yes', this is mapped to 'None'. There is potential for improvement by using the logit score to map directly to a specific class, rather than using the model's output texts. For this mapping process, the probability values for the synonyms of the class text that the logit score has can be further utilized.

4 METHODOLOGY

4.1 Categorical Change

LBC is based on LLM, which has the potential to make more effective inferences on categorical variables than numerical variables. However, this poses a challenge as many key variables in tabular data are numerical. In particular, when LBC deals with OOVs, if the value of the input is numeric, pre-trained knowledge cannot be utilized, unlike categorical type values where the word itself has meaning. Therefore, we experiment by converting numerical type variables in tabular data to categorical variables based on each training and test data columns' quartiles. The quartiles are the values that divide the dataset into four parts. When the dataset is sorted by size, the first quartile (Q1) represents the bottom 25%, the second quartile (Q2) represents the bottom 50%, and the third quartile (Q3) represents the bottom 75%. For example, we converted values less than Q1 to "low" and between Q1 and Q3 to "medium" above Q3 to "high".

4.2 The Order of the Variables

During the tabular data to the prompt conversion process, different prompts are generated depending on the variable order. One instance of tabular data 1 converts to several different types of prompts based on the order of the variables, as follows:

Prompt 1: V_1 is x_1, V_2 is $x_2, \ldots, V_{N-1}$ is x_{N-1}, V_N is $x_N \ldots$.
Prompt 2: V_N is x_N, V_5 is $x_5, \ldots, V_{n-1}$ is x_{n-1}, V_1 is $x_1 \ldots$.
Prompt 3: V_3 is x_3, V_2 is $x_2, \ldots, V_N$ is x_N, V_4 is $x_4 \ldots$.

The total number of prompts that can be generated by changing the order of the variables is $N!$. Every prompt is a transformation of a single instance of tabular data, but the order of the variables gives it a different form, which causes LBC to interpret it differently. Therefore, the order of the variables is a factor that directly affects the performance of LBC.

4.3 The Advanced Order and Indicator

As shown in 4.2, for a single instance of data, different prompts are generated depending on the order of the variables. The same problem occurs in the OOV task, where the number of variables increases due to the addition of OOVs, resulting in more variability in the prompts. This hinders LBC's ability to learn the relationships between tokens. Therefore, LBC performs structured prompt transformations using advanced ordering and indicators to facilitate learning and inference. The format of the training and test prompts with both methods is as follows.

Training Prompt: IV Indicator + IV part + Question

Test Prompt: OOV Indicator + OOV part + IV Indicator + IV part + Question

By positioning the OOV part at the front of the prompt and matching the variable order of the IV part exactly as in training, the IV part in the test prompt has the exact same structure as the IV part in the training prompt. This allows LBC to apply the relationships between variables captured during training to the test as well. Also, since the indicator is always fixed in the same position, it allows LBC to distinguish between the OOV part and the IV part in training and inference. A prompt with both Categorical Change and Advanced Order and Indicator applied is referred to hereafter as an **Advanced Prompt (AP)**. An example of an AP can be found in Fig 2.

4.4 Generalization Ability of LBC: LoRA

According to Zeng and Lee [34], an arbitrary model fine-tuned with LoRA approximates the target model. We extend this theory and theoretically prove that, under certain assumptions, LLMs are fine-tuned with LoRA approximate arbitrary classifiers. Theorem 1 supports the idea that LBC has a high generalization performance in tabular data classification.

Theorem 1. Let $f(\boldsymbol{x})$ represents the ReLU neural network to which LoRA is applied, with no activation function in the last layer, and $\bar{f}(\boldsymbol{x})$ represents the target single-layer linear network. Let $g(x)$ is the logistic function $(1+e^{-x})^{-1}$. $\sigma(\boldsymbol{W})_i$ is the i-th greatest singular value of $\boldsymbol{W}$. $\boldsymbol{W}_l$ and $\overline{\boldsymbol{W}}$ are l-th layer weight matrix of the frozen model and the weight matrix of the target model, respectively.

$$\mathbb{E}\left\|g(f(\boldsymbol{x}))-g(\bar{f}(\boldsymbol{x}))\right\|_2^2 \leq \frac{1}{16}\|\mathbb{E}(\boldsymbol{x}\boldsymbol{x}^T)\|_F\sigma^2\left(\overline{\boldsymbol{W}}-\prod \boldsymbol{W}_l\right)_{\min(\sum_{l=1}^{L} R_l, R_E)+1}.$$

where R_l, R_E are $Rank(W_l), Rank(\overline{\boldsymbol{W}}-\prod \boldsymbol{W}_l)$, respectively. L is the number of layers in f.

4.5 Verbalizer

Verbalizer is applied to LBC for two reasons. The first is to compensate for a weakness in the prediction derivation method of previous work mentioned in 3.4, that is, to obtain direct output from *Logit* so that LBC can be used as a classifier, rather than having the output text of LLM as the prediction of LBC. LBC slightly modifies the structure of traditional LLMs in training and inference to use a verbalizer.

Given a Logit vector $Logit = \{l_{w_1}, l_{w_2}, \ldots, l_{w_V}\}$, where V is the vocabulary size and l_{w_i} is the score for the word w_i in the vocabulary, LBC's score for a single class C_k is calculated as follows:

$$\text{Score}(C_k) = \alpha_1 l_k + \alpha_2 \sum_{w \in S_k} l_w$$

where k is the central word representing class C_k, α_1 and α_2 are the weights for the central word and synonyms, and S_k is the set of synonyms of central word k. For example, if k = 'Yes', then S_k = {'yes', 'yeah', 'true'...}. The probability for C_k is computed using a softmax function:

$$P(C_k) = \frac{\exp(\text{Score}(C_k))}{\sum_{k' \in K}\exp(\text{Score}(C_{k'}))}$$

where K is the set of central words of all classes. And, modify the existing loss function as follows:

$$J = \alpha_1 \text{CE}(Logit, L_k) + \alpha_2 \sum_{w \in S_k} \text{CE}(Logit, L_w)$$

5 EXPERIEMNTS

5.1 Experiments Settings

5.1.1 Dataset. To experiment with reliable datasets used in many studies, we only selected datasets that have been run a number of times in OpenML [29], Kaggle, or used in other benchmarks. Table 1 provides information about the eight datasets we used in our experiments.

Table 1: Dataset Statistics

Dataset	#Variable	#Class	#Instance
Blood [32]	4	2	583
Breast Cancer [35]	31	2	569
Creditcard [23]	15	2	690
German Credit [13]	20	2	1000+
ILPD [25]	11	2	583
Loan [21]	10	2	615
Salary [20]	14	2	1000+
Steel Plate [4]	34	2	1000+

5.1.2 Evaluation. Three main evaluation metrics were used to validate the model: Accuracy, F1 score, and AUC score. Collectively, these metrics ensure a general evaluation of LBC and TMLs.

Accuracy measures the proportion of correct predictions and is defined as Accuracy = $\frac{n_{correct}}{n_{samples}}$. Here, $n_{correct}$ is the number of correct predictions, and $n_{samples}$ is the total number of samples.

F1 score, a harmonic mean of Precision and Recall, is calculated as F1 score = $2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision}+\text{Recall}}$, where Precision = $\frac{TP}{TP+FP}$ and Recall = $\frac{TP}{TP+FN}$.

AUC score represents the area under the ROC curve, which plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings.

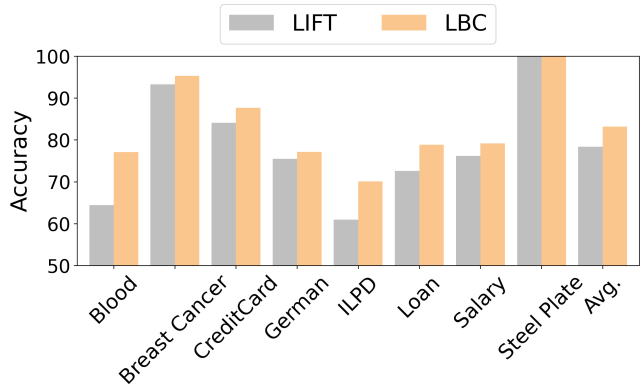

Figure 3: Performance comparison between LIFT and LBC in the non-OOV context. LBC outperforms LIFT on all datasets in non-OOV tasks, i.e., all variables are learned in train and appear in test.

5.1.3 Baseline. We selected five models as baselines to compare their performance with LBC in tabular data classification, and we call them TMLs. Each model performs well on tabular data classification. Details of the TMLs are in Appendix B.

5.2 OOV Setting

To experiment with the performance of LBC on OOV tasks, it is essential to create scenarios where variables that do not exist in training appear in testing. However, we faced a problem because no existing tabular datasets fulfill this requirement. We randomly deleted 50% of the variable columns in the original tabular dataset. As a result, variables that are deleted become OOV, not learned by the model during training, and emerge as new variables in the test. This allows for the assessment of LBC's ability to interpret OOVs. We compare the performance of TMLs and LBC with the data generated by this method.

6 RESULTS

6.1 The Previous Work vs LBC

Before evaluating LBC's performance on OOV tasks, we compare LBC's performance with our previous work, LIFT [7], on non-OOV tasks. Non-OOV tasks refer to situations where all variables are learned without OOV in training, and all variables also appear in the test. Since this task does not need to distinguish between OOVs and IVs, indicators are excluded from test prompt generation. Figure 3 shows that LBC outperforms the traditional method, LIFT, on all eight datasets. These results emphasize the superiority of LBC's approach, which employs a verbalizer as a mapping tool for class mapping, over the method adopted by LIFT, which directly converts the output text into the model's prediction. The use of a verbalizer in LBC demonstrates a more effective strategy by focusing on class mapping rather than a straightforward conversion of output text to predictions. In other words, the verbalizer allows LBC to interpret the role of LLM as a classifier rather than a text generator.

Table 2: LBC vs TMLs in 50% randomly selected OOV situation. The models are trained with 50% IVs, and LBCs add 50% OOVs in the test prompts. The LBC w/o AO column is the result of LBC's inference for the prompts that are written exactly as the variables are ordered in the tabular data, shown in '1. Categorical Change' in Figure 1, and the LBC column is the result of LBC's inference for Advanced Prompt (AP). LBC outperforms the five TMLs on three evaluation scores.

Accuracy	DT	KNN	LogReg	SVM	XGBoost	LBC w/o AO	LBC
Blood	72.67	69.33	75.33	75.33	74.67	**76.00**	**76.00±0.00**
Breast Cancer	93.86	93.86	92.98	92.98	92.98	92.98	**94.15±1.01**
Creditcard	76.81	73.91	72.46	77.54	76.09	76.09	**83.81±0.42**
German	71.00	71.50	77.50	71.50	70.50	76.50	**78.50±0.86**
ILPD	70.94	60.68	72.65	70.94	64.86	69.23	**75.05±0.84**
Loan	69.11	66.67	69.92	69.11	59.35	77.75	**80.59±1.22**
Salary	**85.00**	83.00	83.00	81.50	83.00	82.50	84.00±0.86
Steel Plate	80.21	79.69	73.78	78.15	81.23	81.74	**81.83±1.62**
Avg.	77.53	74.83	77.18	75.01	76.38	78.17	**81.74±0.85**

F1	DT	KNN	LogReg	SVM	XGBoost	LBC w/o AO	LBC
Blood	0.68	**0.73**	0.68	0.63	**0.73**	0.68	0.67±0.00
Breast Cancer	**0.94**	**0.94**	0.93	0.93	0.93	0.93	0.93±0.00
Creditcard	0.67	0.59	0.62	0.62	0.67	0.79	**0.87±0.02**
German	0.73	0.77	0.77	0.73	**0.78**	0.76	0.71±0.01
ILPD	**0.76**	0.71	0.73	0.74	0.75	0.75	0.75±0.00
Loan	0.70	0.70	0.71	0.70	0.69	0.73	**0.76±0.01**
Salary	0.55	0.55	0.55	0.5	**0.59**	0.52	0.52±0.01
Steel Plate	0.8	0.79	0.72	0.79	**0.81**	0.79	0.80±0.01
Avg.	0.72	0.71	0.70	0.68	0.74	0.74	**0.75±0.00**

AUC	DT	KNN	LogReg	SVM	XGBoost	LBC w/o AO	LBC
Blood	0.67	0.61	0.67	0.68	0.68	**0.70**	0.67±0.00
Breast Cancer	0.97	0.98	0.98	**0.99**	**0.99**	0.98	**0.99±0.00**
Creditcard	0.79	0.8	0.83	0.84	0.80	0.84	**0.92±0.02**
German	0.67	0.69	**0.80**	0.67	0.69	0.79	0.79±0.00
ILPD	0.71	0.57	0.68	0.71	0.71	0.73	**0.75±0.01**
Loan	0.56	0.57	0.63	0.51	0.53	0.77	**0.79±0.01**
Salary	0.84	0.85	0.86	0.87	0.86	0.87	**0.88±0.01**
Steel Plate	0.87	0.89	0.89	0.89	0.89	0.89	**0.90±0.00**
Avg.	0.76	0.73	0.78	0.78	0.78	0.82	**0.84±0.00**

6.2 Performance in OOV tasks

Table 2 compares the Accuracy, F1, and AUC scores of TMLs and LBC on eight datasets after conducting 50% OOV conversion. In the Avg. rows for the three evaluation metrics, LBC outperforms the five TMLs. This provides empirical evidence that LBC effectively utilizes pre-trained knowledge to make interpretations about OOV. We explore this in more depth in 6.3.

Comparing LBC w/o AO and LBC columns in Table 2, the difference in AO widened the performance gap across the seven datasets. In the Blood dataset, there is not much difference between LBC w/o AO and LBC because the Blood dataset has the smallest variability for variable order in prompt generation with 4 variables. We empirically investigate these findings in 6.4.

To validate the ability of LBC to perform well on OOV tasks, we conduct experiments on four datasets with different OOV ratios. In each dataset, we vary the OOV ratio to 0%, 30%, 50%, and 70% and observe the model's accuracy change. Figure 4 shows that for traditional TMLs, the performance decreases significantly as the OOV ratio increases. In contrast, LBC shows no decrease in accuracy as the OOV ratio increases or it is small compared to TMLs. These findings suggest that LBC can effectively utilize the pre-trained knowledge of LLMs to outperform traditional machine learning methods even as the percentage of OOVs increases.

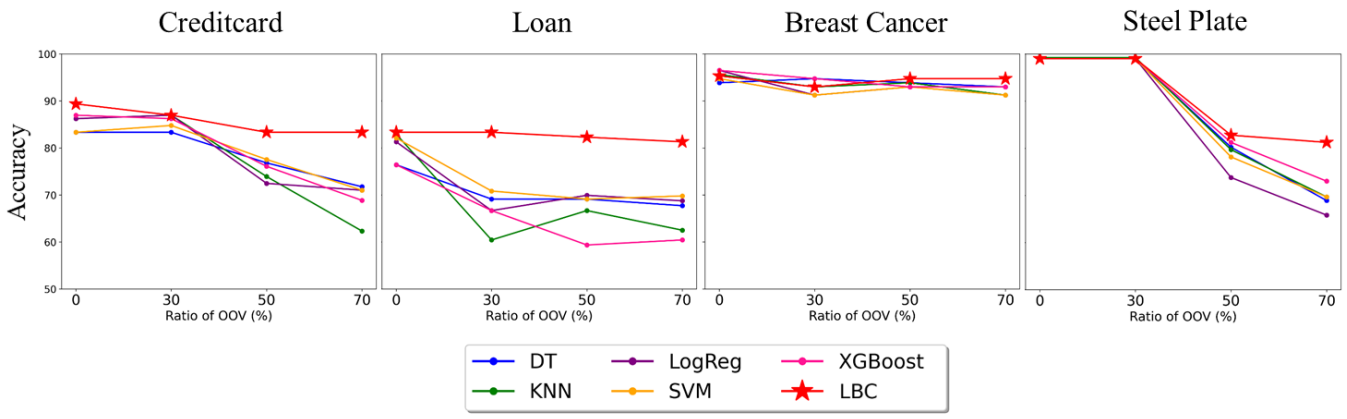

Figure 4: Graph of accuracy changing over OOV ratio (%): We observed the accuracy change of TMLs and LBC by increasing the OOV ratio from 0, 30, 50, and 70 (%) for four datasets. Comparing the accuracy reduction of TMLs and LBC, the reduction of LBC is smaller compared to TMLs. It demonstrates that LBC interpret OOVs, unlike TMLs.

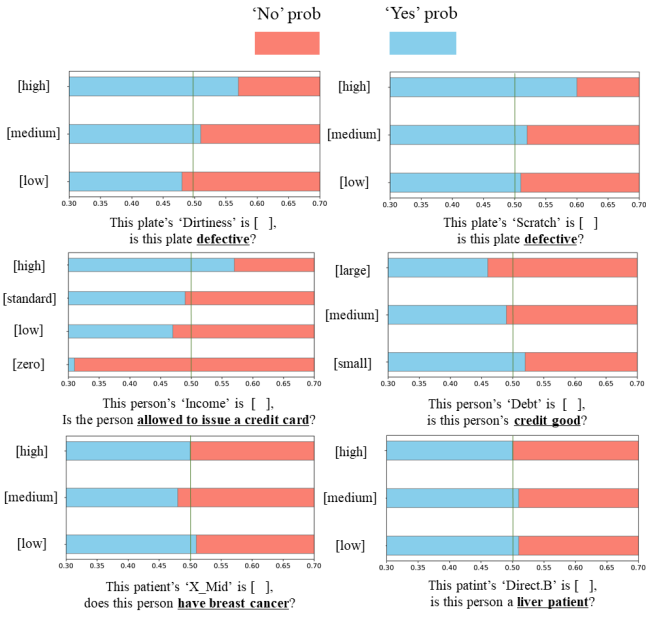

Figure 5: Observing how LBC applies its pre-trained knowledge to prompts about OOVs, thereby revealing biases in its pre-trained knowledge. Intuitively, LBC has a bias toward making its predictions closer to the correct answer. However, it is not responsive to special variable names that do not have a word meaning.

6.3 LBC's Ability to utilize Pre-Trained Knowledge

In this section, we specifically investigate how LBC use their pre-trained knowledge to interpret OOVs in the OOV tasks. We conduct an experiment to observe the pre-trained vias for variables using an LBC that has been trained with the structure of data without any information about the variables. For datasets with a "Yes" or "No" answer, The structure of the data is as follows:

$$\begin{aligned}\text{Prompt} = {} & \text{string(Start of sentence)+} \\ & \text{string('Variable name' is [Variable value])+} \\ & \text{string(Question\#\#\#)} \\ \text{Answer} = {} & \text{Yes@@@ or No@@@}\end{aligned}$$

In the training process, the prompt structure is utilized as it is, with experimental adjustments made to balance the likelihood of the trained LBC predicting 'Yes' or 'No'. During testing, we replace the names and values of various variables in the 'Variable name' and 'Variable value' placeholders within the prompts to evaluate the pre-trained biases of LBC towards those variables.

Figure 5 presents the outcomes for several variables of high importance in this experiment. It is evident that LBC leverages pre-trained knowledge to approximate the probabilities for variables not learned during training closer to the correct answers. Notably, the interpretation that the graph for "income" shows a high risk of issuing a credit card to an individual with an income of "0" matches with the actual distribution in the Creditcard dataset. However, for the unique variables in the table dataset, such as "Direct.B", which does not have the meaning of a common word, LBC showed almost balanced results and tended to make predictions without any clear bias. This shows that LBC maintains a neutral approach to uninterpretable variables, and maintains an even probability distribution without any particular tendency. These results support the high performance of LBC in handling out-of-variable (OOV) tasks.

Additionally, we experiment with the same conditions as TMLs without providing the LBC with any information about OOVs. Using the Creditcard dataset with

Table 3: The performance of LBC under the same conditions as TMLs.

Metric	Xgboost	LBC w/o OOV	LBC w/ OOV
Accuracy	76.09	74.63 ± 0.90	**83.81 ± 0.42**
F1	0.67	0.72 ± 0.01	**0.87 ± 0.02**
AUC	0.80	0.81 ± 0.00	**0.92 ± 0.02**

the same settings as Table 2, we feed the LBC with test prompts that excluded OOV information and measure its performance. The results shows that the LBC's performance on prompts without OOV information was significantly lower than when OOVs were included, highlighting the fact that the LBC gets information from OOVs.

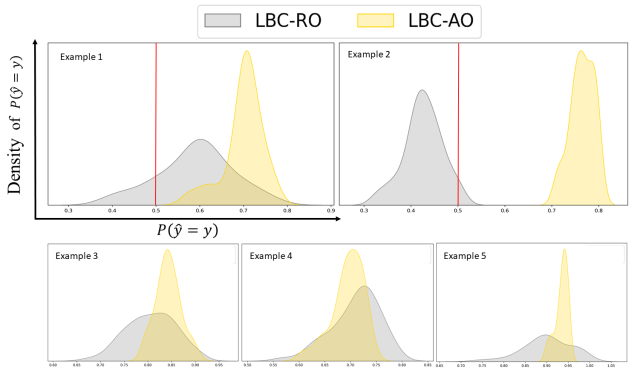

Figure 6: Density of $P(\hat{y} = y)$ for the two prompt generation methods. We repeated the prompt generation 100 times for each of the five randomly selected examples from the Creditcard dataset in two ways: random order (RO) and advanced order (AO). Although it is one and the same row in the tabular data, different prompts can be generated depending on the two methods. The horizontal axis represents the model's probability for the correct class y across prompts, and the vertical axis shows frequency. The AO method yields more consistent probabilities than RO. In the samples with the red vertical line, which is the prediction boundary, the difference between the two methods results in different predictions.

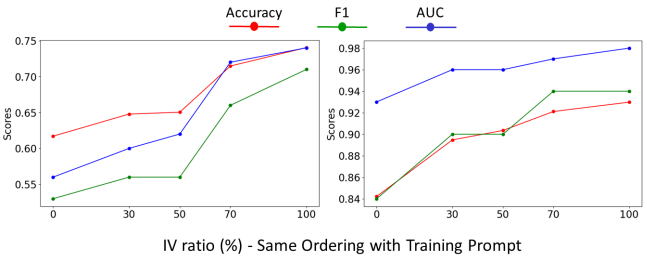

Figure 7: The changes in three scores according to the ratio of IVs in the test prompt that maintains the same order as the IVs in the training prompt. Graphs for the Steel Plate on the left and the Breast Cancer dataset on the right. Both the training and test prompts consist of only the IVs used in table 2. As the ratio of the same IV order increases between the training and test prompts, all three scores improve. This outcome demonstrates the importance of applying the same order of IVs in the test prompt as in the training prompt.

6.4 Importance of Advanced Prompt

In this section, we'll take a closer look at how Advanced Prompts, such as "Consider the order of variables" or "Add an Indicator," used to generate test prompts, affect LBC's probability output.

To verify the importance of the variables' order, we experiment with repeatedly generating two types of prompts by randomly selecting an instance from the tabular data: One, where the order of all variables is randomized (LBC-RO), and the other, where the order of the IVs matches to the IV of the training data, and only the order of the OOVs are randomized (LBC-AO). We randomly select five instances from the Creditcard dataset and generate 100 different prompts for each instance with the RO and AO methods, respectively, to compare the probability distributions generated by LBC for the two methods. Figure 6 illustrates the performance difference between prompts where the order of variables is matched with the training data and those where it is not. LBC-RO exhibits a large variance in the probability distribution, leading to variations in the model's predictions for a single data instance. In contrast, LBC-AO shows a small variance in the probability distribution, which means that the model makes consistent predictions.

To further investigate the benefits of matching the order of IVs of test prompts with the training prompts, we conduct additional experiments. We compose the training and test data using only the IVs, excluding the OOVs selected from the Steel Plate dataset used in Table 2. Then, for the variables that make up the test prompt, we experiment with increasing the ratio of variables in the same order as the variable order of the training prompt to check the scores for the three evaluation metrics. Figure 7 illustrates the scores for the three evaluation metrics. As the IVs ratio increases, the performance improves on all three metrics. From this, we can see that LBC performs best when the variables in the training data are delivered to the test in the same order, which means that LBC is learning the entire prompt that reflects the order of the variables, rather than simply learning information about each variable in training.

7 CONCLUSION

In this work, we propose LBC to solve OOV tasks. Although TMLs have shown outstanding performance, they are limited in OOV tasks due to their inability to handle the variables they never learned in training. LBCs, on the other hand, utilize prompt-based inference, which allows information about OOVs to be added to prompts in a straightforward way and enables understanding of the new information through pre-trained knowledge. To utilize LLM's reasoning capabilities on tabular data, LBC takes the three steps we propose. First, we apply Categorical Change, which converts numeric data types to string types, prompting LLM to interpret the meaning of features as sentences. Second, in Advanced Ordering, our proposed variable ordering scheme places OOVs before IVs and maintains the order of IVs with the training phase. This method is simple but yields significant performance gains. Third, a class mapping method from logit scores using a verbalizer allows the LBC to function as a classifier rather than a language model. LBC is the first approach to apply pre-trained LLM to OOV tasks.

REFERENCES

[1] Ali Bodaghi, Nadia Fattahi, and Ali Ramazani. 2023. Biomarkers: Promising and valuable tools towards diagnosis, prognosis and treatment of Covid-19 and other diseases. *Heliyon* 9, 2 (2023).

[2] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. 2022. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[4] Massimo Buscema, Stefano Terzi, and William Tastle. 2010. Steel Plates Faults. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5J88N.

[5] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[7] Tuan Dinh, Yuchen Zeng, Ruisu Zhang, Ziqian Lin, Michael Gira, Shashank Rajput, Jy-yong Sohn, Dimitris Papailiopoulos, and Kangwook Lee. 2022. Lift: Language-interfaced fine-tuning for non-language machine learning tasks. *Advances in Neural Information Processing Systems* 35 (2022), 11763–11784.

[8] Allan Donner. 1982. The relative effectiveness of procedures commonly used in multiple regression analysis for dealing with missing values. *The American Statistician* 36, 4 (1982), 378–381.

[9] Luigi Gresele, Julius Von Kügelgen, Jonas Kübler, Elke Kirschbaum, Bernhard Schölkopf, and Dominik Janzing. 2022. Causal inference through the structural causal marginal problem. In *International Conference on Machine Learning*. PMLR, 7793–7824.

[10] Siyuan Guo, Jonas Wildberger, and Bernhard Schölkopf. 2023. Out-of-Variable Generalization. *arXiv preprint arXiv:2304.07896* (2023).

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[12] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[13] Hans Hofmann. 1994. Statlog (German Credit Data). UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5NC77.

[14] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. 2022. Tabpfn: A transformer that solves small tabular classification problems in a second. *arXiv preprint arXiv:2207.01848* (2022).

[15] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* (2021).

[16] Shengding Hu, Ning Ding, Huadong Wang, Zhiyuan Liu, Jingang Wang, Juanzi Li, Wei Wu, and Maosong Sun. 2022. Knowledgeable Prompt-tuning: Incorporating Knowledge into Prompt Verbalizer for Text Classification. arXiv:2108.02035 [cs.CL]

[17] Shengding Hu, Ning Ding, Huadong Wang, Zhiyuan Liu, Jingang Wang, Juanzi Li, Wei Wu, and Maosong Sun. 2022. Knowledgeable Prompt-tuning: Incorporating Knowledge into Prompt Verbalizer for Text Classification. arXiv:2108.02035 [cs.CL]

[18] Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. 2020. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678* (2020).

[19] M I Jordan. 1986. Serial order: a parallel distributed processing approach. Technical report, June 1985-March 1986. (5 1986). https://www.osti.gov/biblio/6910294

[20] Ron Kohavi. 1996. Census Income. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5GP7S.

[21] Mazaharul Hasnine Mirza. 2023. Loan Data Set. https://doi.org/10.34740/KAGGLE/DSV/5149638

[22] OpenAI. 2021. Fine-tuning. https://platform.openai.com/docs/guides/fine-tuning Accessed on 01-02, 2024.

[23] J. R. Quinlan. [n. d.]. Credit Approval. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5FS30.

[24] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).

[25] Bendi Ramana and N. Venkateswarlu. 2012. ILPD (Indian Liver Patient Dataset). UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5D02C.

[26] Donald B Rubin. 1976. Inference and missing data. *Biometrika* 63, 3 (1976), 581–592.

[27] Timo Schick and Hinrich Schütze. 2021. It's Not Just Size That Matters: Small Language Models Are Also Few-Shot Learners. arXiv:2009.07118 [cs.CL]

[28] Baohua Sun, Lin Yang, Wenhan Zhang, Michael Lin, Patrick Dong, Charles Young, and Jason Dong. 2019. Supertml: Two-dimensional word embedding for the precognition on structured tabular data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 0–0.

[29] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. 2013. OpenML: Networked Science in Machine Learning. *SIGKDD Explorations* 15, 2 (2013), 49–60. https://doi.org/10.1145/2641190.2641198

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[31] Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax.

[32] I-Cheng Yeh. 2008. Blood Transfusion Service Center. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5GS39.

[33] Jinsung Yoon, Yao Zhang, James Jordon, and Mihaela van der Schaar. 2020. VIME: Extending the Success of Self- and Semi-supervised Learning to Tabular Domain. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 11033–11043. https://proceedings.neurips.cc/paper_files/paper/2020/file/7d97667a3e056acab9aaf653807b4a03-Paper.pdf

[34] Yuchen Zeng and Kangwook Lee. 2023. The expressive power of low-rank adaptation. *arXiv preprint arXiv:2310.17513* (2023).

[35] Matjaz Zwitter and Milan Soklic. 1988. Breast Cancer. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C51P4M.

A PROOF OF THEOREM 1

According to Zeng and Lee [34], an arbitrary model fine-tuned with LoRA approximates the target model. We extend this theory and theoretically prove that, under certain assumptions, LLMs are fine-tuned with LoRA approximate arbitrary classifiers. Theorem 1 supports the idea that LBC has a high generalization performance in tabular data classification.

Lemma 1. The logistic function $g(x) = (1 + e^{-x})^{-1}$ is Lipschitz continuous with a Lipschitz constant of $1/4$.

Proof of Lemma 1. A function $f : \mathbb{R} \to \mathbb{R}$ is Lipschitz continuous if

$$\exists K > 0, \forall x_1, x_2 \in \mathbb{R}, |f(x_1) - f(x_2)| \leq K|x_1 - x_2|.$$

By substituting f with g, and considering that g is a monotonic function, we can obtain the following expression:

$$\frac{g(x_1) - g(x_2)}{x_1 - x_2} \leq K.$$

By the mean value theorem,

$$g'(c) = \frac{g(x_2) - g(x_1)}{x_2 - x_1} \leq K, \text{and}$$
$$0 < g'(c) \leq \frac{1}{4}$$
$$(g'(c) = g(c)(1 - g(c)) \wedge 0 < g(c) < 1)$$
$$\rightarrow K \geq \frac{1}{4}.$$

A new theorem, which is a variation of Lemma 11 of [34], can be proposed using Lemma 1 above.

Theorem 1. Let $f(\boldsymbol{x})$ represents the ReLU neural network to which LoRA is applied, with no activation function in the last layer, and $\bar{f}(\boldsymbol{x})$ represents the target single-layer linear network. Let $g(x)$ is the logistic function $(1+e^{-x})^{-1}$. $\sigma(\boldsymbol{W})_i$ is the i-th greatest singular value of $\boldsymbol{W}$. $\boldsymbol{W}_l$ and $\overline{\boldsymbol{W}}$ are l-th layer weight matrix of the frozen

model and the weight matrix of the target model, respectively.

$$\begin{aligned}
&\mathbb{E}\left\|g(f(\boldsymbol{x})) - g(\bar{f}(\boldsymbol{x}))\right\|_2^2 \\
&\quad \leq \frac{1}{16}\mathbb{E}\left\|(f(\boldsymbol{x}) - \bar{f}(\boldsymbol{x}))\right\|_2^2 \\
&\qquad\qquad\qquad\qquad \text{(g is 1/4 Lipschitz by Lemma 1)} \\
&\quad \leq \frac{1}{16}\|\mathbb{E}(\boldsymbol{x}\boldsymbol{x}^T)\|_{\mathrm{F}}\,\sigma^2\left(\overline{\boldsymbol{W}} - \prod \boldsymbol{W}_l\right)_{\min(\sum_{l=1}^{L} R_l, R_E)+1}.
\end{aligned}$$

where R_l, R_E are $Rank(W_l), Rank(\overline{\boldsymbol{W}} - \prod \boldsymbol{W}_l)$, respectively. L is the number of layers in f.

B TRADITIONAL MACHINE LEARNING MODELS

For Traditional Machine Learning Models, we selected 5 models. For tree-based models, we chose Decision Tree and XGBoost. Tree-based models have strong performance in tabular data classification. We also included K-Nearest Neighbor, Logistic Regression, and Support Vector Machine to increase the diversity of the models.

Decision Tree. A Decision Tree (DT) is a model used for classification and regression tasks. The model trains on data to make predictions based on simple decision rules. The advantage of decision trees is that they can capture non-linear patterns in your data, and the results of the model are easy to interpret.

K-Nearest Neighbor The K-Nearest Neighbor (KNN) algorithm is used in classification and regression to make predictions based on the data of the K closest neighbors. This model's key parameters are the number of neighbors (K) and how the distance is measured.

Logistic Regression Logistic regression (LogReg) is a statistical model often used for classification problems. This model is often used when the outcome is binary, and it estimates probabilities to perform classification based on decision boundaries.

Support Vector Machine A Support Vector Machine (SVM) is a machine learning model used for classification and regression problems. The model finds the optimal decision boundary to divide a given set of data into categories, and in this study, we used a Radial Basis Function (RBF) kernel.

XGBoost XGBoost is a high-performance machine learning model based on the Gradient Boosting algorithm, which is a decision tree-based ensemble learning method that combines multiple tree models to improve prediction accuracy. At each step, XGBoost adds a new model to reduce the error of the previous model and uses the Gradient Boosting technique in the process. XGBoost is a model that is state-of-the-art on many benchmarks.

All 5 models were imported and used from scikit-learn. We also used scikit-learn's HalvingGridSearchCV class to explore the optimal hyperparameters.