CS772 Project Report

Sachin Bhadang
200831
sachinb20@iitk.ac.in

Pulkit Dhamija
200738
pulkitd20@iitk.ac.in

Ashutosh Sharma
208070216
ashutoshs20@iitk.ac.in

Arnav Shendurnikar
200929
sarnav20@iitk.ac.in

1 Problem Description / Motivation

Bayesian inference provides a principled framework for incorporating prior knowledge into predictive modeling, allowing models to make more informed decisions in the presence of uncertainty. This becomes particularly valuable in domains such as healthcare and biology, where labeled data is often scarce, costly, or difficult to obtain. By integrating expert knowledge as priors, Bayesian models can achieve better generalization even in low-data regimes.

Recent work, such as AutoElicit, has demonstrated that large language models (LLMs) can automate the process of eliciting informative priors from natural language prompts. However, these efforts have primarily focused on **linear models**, limiting their utility in capturing the complexities of many real-world systems, which are inherently **non-linear**.

In this project, we explore extending LLM-based prior elicitation to **non-linear Bayesian models**, including Bayesian Neural Networks (BNNs), Polynomial Regression, and Gaussian Mixture Models (GMMs). We evaluate our approach across diverse real-world datasets: breast cancer detection using BNNs, online shopping purchasing intention using Bayesian logistic regression, the Iris dataset with GMMs, and the California housing dataset with polynomial regression. By enabling automated prior elicitation for these expressive models, our work aims to enhance predictive performance and broaden the applicability of Bayesian methods in data-scarce and complex domains. The code for this project is available in the following GitHub repository: LLM_NL_Priors.

2 Literature Review and Description of Prior Work

Traditional methods for prior elicitation, such as the SHELF framework, have shown promise but are not scalable, especially in complex or large-scale applications. Recent efforts have sought to automate this process, leveraging the power of large language models (LLMs) and other machine learning techniques. Below are key works in this area:

2.1 AutoElicit

AutoElicit Capstick et al. (2024) is a significant advancement in automating prior elicitation, specifically for linear models. By using large language models (LLMs), AutoElicit automatically generates Gaussian priors, reducing the need for manual expert input. This approach offers scalability and efficiency but is primarily limited to linear models, which constrains its broader applicability to real-world complex phenomena that often require non-linear models.

2.2 LLM-Guided Feature Selection and Data Generation

Recent works, such as those by Choi et al. and Gouk et al., explore the use of LLMs for guiding feature selection and generating synthetic data for predictive modeling. These methods utilize LLMs

Radford et al. (2018) to identify important features in a dataset, which can be critical for improving model performance, particularly when the available data is limited. However, while these methods enhance model selection and data augmentation, they do not address the problem of prior elicitation for Bayesian models.

2.3 In-Context Learning (ICL)

In-context learning (ICL) Brown et al. (2020) refers to the ability of large language models to adapt to a task by conditioning on a prompt or a set of examples provided during inference. While not explicitly Bayesian, ICL has been applied in various domains for improving model performance without requiring traditional retraining. However, this approach lacks the transparency and principled integration of prior knowledge that Bayesian methods offer, limiting its use in tasks where explicit priors are crucial for performance.

However, no framework exists for using LLMs to elicit priors for non-linear models. Our work addresses this gap.

3 Novelty of Our Work

We implemented a basic yet functional version of the AutoEllicit pre-elicitation pipeline to support non-linear models using Hugging Face-hosted LLMs. This enabled the creation of structured priors for complex model architectures, thereby extending interpretable prior design beyond traditional linear models. We tested the functionality of AutoEllicit on both **linear models** such as Bayesian polynomial regression and **non-linear** models including Bayesian neural networks and Gaussian mixture models.

Limitations

Our AutoEllicit-style previous generation method for generating non-linear models using Hugging Face-based LLMs was put into use on Google Colab. Because of hardware limits, the free-tier setting only let us use 1.5B-parameter models. Models 7B and bigger couldn't work because they didn't have enough memory. It may be easier to add higher-capacity models to the modeling process because their results are more organized and can be used right away.

3.1 Prior Generation Strategies for Bayesian Modeling

To initialize a Bayesian model $\mathcal{M}$ with informative or neutral uncertainty estimates, we explore two strategies for generating priors over the model parameters and compare them across various case studies:

- **LLM-derived feature-specific priors:** We use a prompt-based prior elicitation pipeline inspired from (*AutoEllicit*) that combines large language models (LLMs) with task-specific templates to generate Gaussian priors for each feature. Each prompt contains a system instruction, a user task, and a list of features. The LLM generates, for each feature f, a prior mean μ_f and standard deviation σ_f.

- **Random Gaussian priors:** In this approach, we randomly sample prior parameters across M independent runs. Each run initializes the model with a prior drawn from standard Gaussian distributions, and the results are averaged to form a baseline for comparison.

Algorithm 1 Generating Feature-Specific Priors via AutoElicit

Input: List of LLM models $\mathcal{M}$, Prompt configurations $\mathcal{P}$
Output: Feature-specific priors $\{\mu_f, \sigma_f\}$ for each feature f
for each model $M_i \in \mathcal{M}$ **do**
    Load LLM model M_i
    for each prompt $P_j \in \mathcal{P}$ **do**
        Extract system and user content from P_j
        Construct full prompt with feature list
        Pass prompt to M_i to generate prior response
    end for
end for
Return dictionary of all generated priors

Algorithm 2 Training with Random Gaussian Priors

Input: Input dimension d, number of models M, training data $(X_{\text{train}}, y_{\text{train}})$, prior variance scale σ^2
for $i = 1$ to M **do**
    Sample $\mu^{(i)} \sim \mathcal{N}(0, I_d)$
    Sample $\sigma^{(i)} \sim |\mathcal{N}(0, \sigma^2 I_d)|$
    Initialize model $\mathcal{M}^{(i)}$ with prior $(\mu^{(i)}, \sigma^{(i)})$
    Train $\mathcal{M}^{(i)}$ on $(X_{\text{train}}, y_{\text{train}})$
    Record train and test losses
end for
Compute average loss across M models

4 Tools and Software Used

- LLMs: TinyLlama-1.1 Zhang et al. (2024), Qwen-1.5 Peng et al. (2023)
- Libraries: Transfromers, Torch, SkLearn
- Experimental framework for prompt engineering and model training

5 Experimental Results and Data Used

5.1 Logistic Regression for Online Shoppers Intention

The dataset used in this study is the Online Shoppers Intention Dataset from the UCI Machine Learning Repository. It includes session data from an e-commerce website, with the aim of predicting whether a user will generate revenue (i.e., make a purchase). The dataset consist of 12,000 user sessions (data points) and 18 features (e.g., number of product-related page visits, bounce rate, exit rate, session durations, etc.). The target variable is Revenue which is true if purchase is made and false otherwise.

We have used Bayesian logistic regression for classification. Logistic regression is a binary classification model that predicts the probability of a binary outcome using the sigmoid function:

$$P(y = 1 \mid \mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b), \quad \text{where} \quad \sigma(z) = \frac{1}{1 + e^{-z}}$$

Given training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, the model minimizes the negative log-likelihood:

$$\mathcal{L}(\mathbf{w}) = -\sum_{i=1}^{N} \left[y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) \right], \quad \hat{y}_i = \sigma(\mathbf{w}^\top \mathbf{x}_i + b)$$

Bayesian logistic regression introduces a probabilistic framework by placing a prior over the model weights

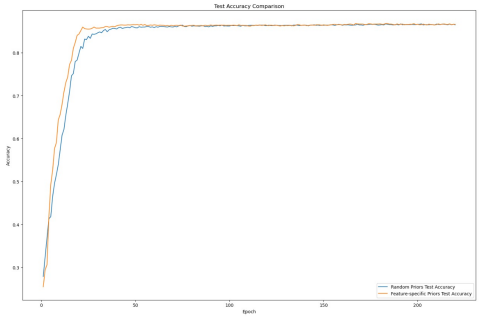
$$p(\mathbf{w} \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \mathbf{w})\, p(\mathbf{w})}{p(\mathcal{D})}$$

Here, $p(\mathcal{D} \mid \mathbf{w})$ is the likelihood (as in standard logistic regression), $p(\mathbf{w})$ is the prior over weights, $p(\mathbf{w} \mid \mathcal{D})$ is the posterior.

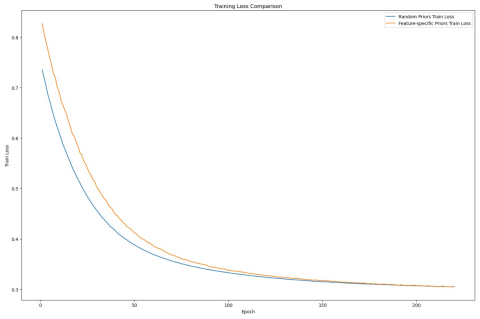
The predictive distribution is given by:

$$p(y^* \mid \mathbf{x}^*, \mathcal{D}) = \int \sigma(\mathbf{w}^\top \mathbf{x}^*)\, p(\mathbf{w} \mid \mathcal{D})\, d\mathbf{w}$$

This integral is intractable and approximated using Monte Carlo sampling. We build two models, where the first model is trained using Algorithm 1 and the second model is trained using Algorithm 2.



(a) Training loss comparison between structured and random priors.



(b) Test accuracy comparison between structured and random priors.

Figure 1: Comparison between structured and random priors for training loss and test accuracy.

Figure 1 shows the training loss comparison where structured priors (likely the green line) converge slightly faster than random priors (orange line). The test accuracy plot shows minimal difference in final performance - just a 0.1% improvement with feature-specific priors (86.6% vs 86.5%). The learning curves appear to follow similar patterns, indicating that while LLM-derived priors may provide some initial guidance during training, their impact on final model performance is negligible for this particular task.

5.2 Bayesian Polynomial Regression

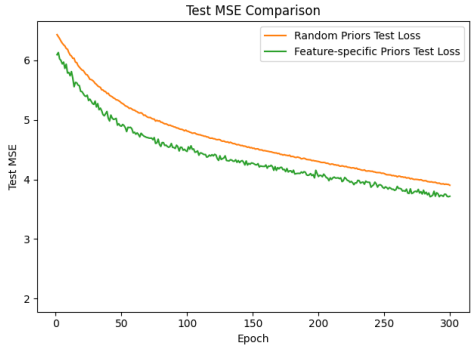
We employ the **California Housing dataset**, which includes standardized attributes such as `MedInc`, `HouseAge`, `AveRooms`, and `Latitude`, among others. The dataset consists of 20,640 data elements. A second-degree polynomial regression model is implemented.

To model the relationship between housing features and the target variable, `MedHouseVal`, we present a Bayesian linear model that employs **polynomial regression**. The general form of a polynomial regression model is as follows:

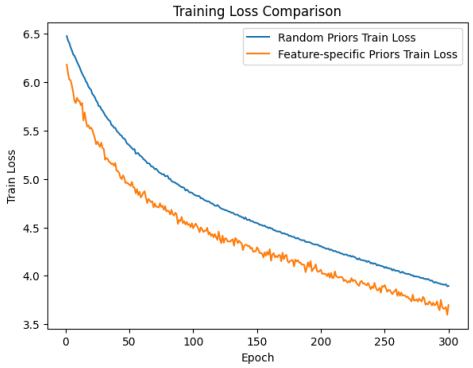
$$y = w_0 + \sum_{i=1}^{n} \sum_{j=1}^{d} w_{ij} x_i^j + \varepsilon$$

where w_{ij} denotes the regression coefficients (weights), x_i signifies the input features, d indicates the polynomial degree, and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ represents Gaussian noise.

We visualize and compare the performance of these models using training loss and test accuracy. These metrics demonstrate the impact of structured priors on both convergence and generalization.



(a) Training loss comparison between structured and random priors.



(b) Test accuracy comparison between structured and random priors.

Figure 2: Comparison between structured and random priors for training loss and test accuracy.

Looking at Figure 2, the training loss comparison shows that feature-specific priors (green line) converge slightly faster than random priors (orange line). The test accuracy plot shows that models with feature-specific priors maintained a consistently lower MSE (3.7) compared to random priors (3.9). This suggests that LLM-derived priors provided a small but consistent advantage in generalization performance. The improvement, while present, isn't dramatic - about 5% better MSE with the structured priors approach.

5.3 Gaussian Mixture Models with Prior Elicitation on the Iris Dataset

The Iris dataset is a classic benchmark in pattern recognition, containing measurements of four features: sepal length, sepal width, petal length, and petal width for three species of iris flowers (Setosa, Versicolor, and Virginica). Each species contains 50 samples, totaling 150 data points. This dataset is ideal for testing clustering algorithms as it contains distinct classes with varying degrees of separation.

Gaussian Mixture Models (GMMs) are probabilistic models that assume data points are generated from a mixture of several Gaussian distributions. In the context of unsupervised clustering, GMMs estimate both the parameters of these Gaussian distributions and the probability of each data point belonging to a specific cluster. The probability density function of a GMM with K components is defined as:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Where:

- π_k represents the mixture weight for component k ($\sum_{k=1}^{K} \pi_k = 1$)
- $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is the Gaussian density with mean $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{\Sigma}_k$
- $\mathbf{x}$ represents a data point in d-dimensional space

In Bayesian GMMs, we place prior distributions over model parameters. For feature weights (determining feature relevance in clustering), we can specify Gaussian priors with means and standard deviations reflecting our beliefs about feature importance.

The Iris dataset is well-suited for GMM application as:

1. The three iris species form natural clusters in feature space
2. The standardized features (mean 0, std 1) allow for meaningful prior specification
3. The dataset size is appropriate for demonstrating the impact of priors on clustering performance

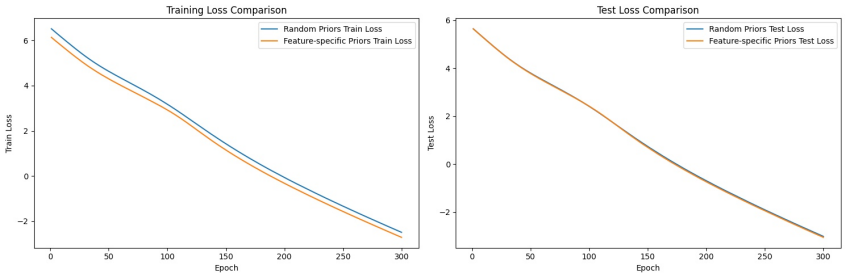


Figure 3: Training loss comparison between structured and random priors.

We compared the performance of our Bayesian GMM between two approaches: random priors (Algorithm 1) and specific priors (Algorithm 2).

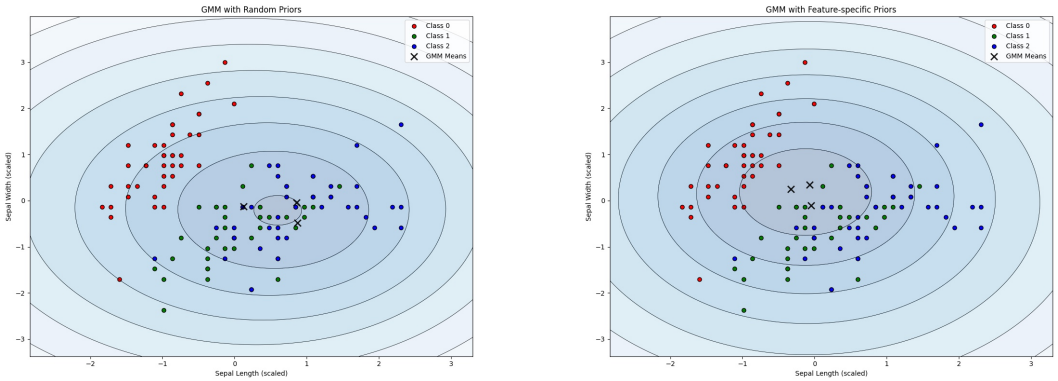


Figure 4: Clusters formation comparison between random and specific for different priors

The negative log-likelihood (NLL) metric shows a more noticeable improvement with feature-specific priors (-3.5) compared to random priors (-3.02). This represents approximately a 16 % improvement, making this the most substantial difference across all experiments. Figure 4 shows the cluster formations, where both approaches successfully identify the three iris species, but the feature-specific prior model appears to create slightly more distinct boundaries between clusters.

5.4 Bayesian Neural Network for classification

The Breast Cancer Wisconsin (Diagnostic) Dataset is a widely-used benchmark in medical machine learning. It consists of features extracted from digitised images of fine needle aspirates (FNA of breast masses. Each of the 569 samples contains 30 real-valued features that describe the characteristics of cell nuclei present in the image. The goal is to classify tumours as either **malignant** or **benign**. We use Bayesian Neural Network for the task.

A Bayesian Neural Network (BNN) incorporates uncertainty into a neural network by placing probability distributions over its weights instead of treating them as fixed parameters. This framework allows the model to represent confidence in its predictions, crucial for high-stakes tasks such as medical diagnostics.

Given the input features $x_n \in \mathbb{R}^d$ and binary outputs $y_n \in 0, 1$. We are training a neural network for classification. A neural network consists of layers of neurons and are connected by activation functions. Each neuron performs a weighted sum of its inputs followed by a non-linear activation:

$$z = \sum_{i=1}^{n} wx_i + b = \mathbf{w}^T\mathbf{x} + b$$
$$a = \phi(x)$$

For a layer l with input $a^{(l-1)}$, we have,

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)} a^{(l-1)} + b^{(l)}$$
$$a^{(l)} = \phi(z^{(l)})$$

In Bayesian Neural Network, we define priors over the weights $p(\mathbf{W})$ of the neurons $\mathbf{W}$. We also define the likelihood of observing the label y_n given x_n and weights $\mathbf{w}$, and then calculate the posterior $p(\mathbf{W}|D)$ using the Bayes rule. For classification, we use the posterior predictive distribution defined as,

$$p(y^*|\mathbf{x}^*, \mathcal{D}) = \int p(y^*|\mathbf{x}^*, \mathbf{W})\, p(\mathbf{W}|D)\, d\mathbf{W}$$

As this integral is generally intractable, we use Monte Carlo methods to approximate it.

We trained two models. The first one assumes non-informative priors (Gaussian) over all the weights of the Neural network (Algorithm 2). The second model uses Algorithm 1 to elicit priors for the weights of the first layer of the model and use non-informative priors on the weights of all other layers.

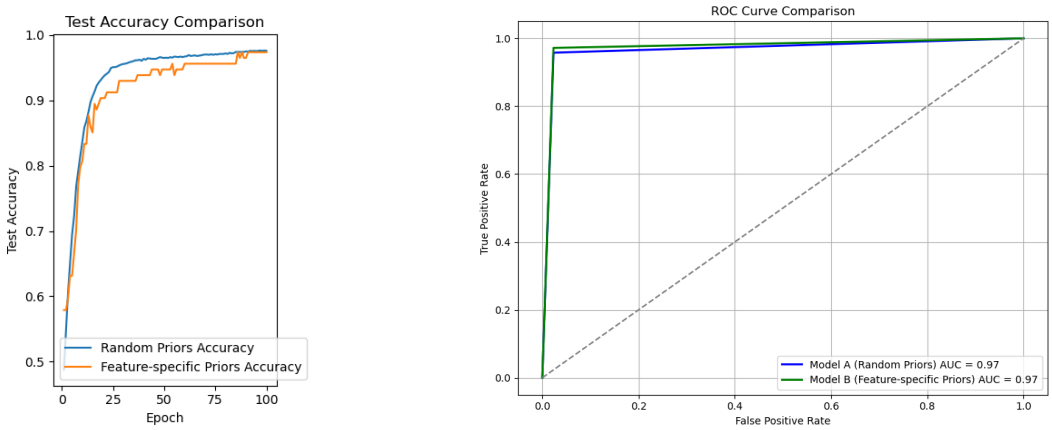


Figure 5: Comparison between structured and random priors for training loss and test accuracy.

Figure 5 shows that while both approaches quickly achieve high accuracy, they follow different convergence patterns. The random priors model (blue line) shows steadier improvement, while the feature-specific priors model (orange line) shows more fluctuations before stabilizing. The final test accuracy results are nearly identical: 97.5% for random priors versus 97.4% for feature-specific priors. The ROC curve comparison in the right panel of Figure 5 confirms this similarity, with both approaches achieving an AUC (Area Under Curve) of approximately 0.97. In this case, the more complex approach of deriving specific priors didn't yield any practical advantage over simple random priors.

Summary

Model	**Random Priors**	**Feature-Specific Priors**
Bayesian Logistic Regression (Accuracy)	86.5%	86.6%
Bayesian Polynomial Regression (MSE)	3.9	3.7
Bayesian Neural Network (Accuracy)	97.5%	97.4%
Gaussian Mixture Models (NLL)	-3.02	-3.5

Table 1: Comparison of model performance with random vs. feature-specific priors.

Overall, the experiments show that while LLM-derived feature-specific priors can offer improvements over random priors, the differences are generally small:

The most significant improvement was seen in the GMM model for the Iris dataset (16% better NLL) For polynomial regression, there was a modest 5% improvement in MSE For classification tasks (logistic regression and neural networks), the differences were minimal (less than 0.1% in accuracy)

6 Things We Learned

LLMs encode usable non-linear intuitions. Through our experiments, we observed that large language models—despite being trained primarily on text—capture implicit knowledge about non-linear relationships and functional forms. When prompted appropriately, they can suggest priors that reflect curve shapes, variance patterns, and interaction effects beyond simple linear assumptions.

Translation from text to prior distributions is feasible but non-trivial. Converting natural-language descriptions into formal prior distributions requires careful prompt design and post-processing. Ambiguities in wording, varying levels of expert specificity, and the need to choose among distribution families all contribute to the challenge, underscoring the importance of iterative refinement and validation.

Elicited priors aid in small-data scenarios and improve generalization. In our low-data settings—such as breast cancer detection and small-scale regression tasks—automated priors reduced overfitting and led to more robust uncertainty estimates. Even imperfect priors provided enough guidance to outperform non-informative baselines, demonstrating the practical value of LLM-assisted elicitation in data-scarce domains.

7 Possible Future Work

Human-in-the-Loop Feedback Integrate expert review into the LLM-based elicitation loop, allowing domain specialists to adjust and refine proposed priors for greater accuracy and trustworthiness.

Deep Probabilistic Architectures Adapt the LLM elicitation framework to deep Bayesian models (e.g., Bayesian Transformers), translating expert knowledge into priors for complex architectures.

8 Team Contributions

- **Sachin Bhadang**: Problem formulation, report writing
- **Pulkit Dhamija**: Literature review, LLM prompting, BNNs, code implementation
- **Ashutosh Sharma**: Designing experiments, dataset handling, code implementation
- **Arnav Shendurnikar**: Report writing, visualizations, future work, code implementation

9 Disclaimer

This is a fresh project for which we have not earned any credit elsewhere or done under any internship.

References

Brown, T. B. et al. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Capstick, A. et al. (2024). Autoelicit: Using large language models for expert prior elicitation in predictive modelling. *arXiv preprint arXiv:2411.17284*. Version 4, last revised 31 Jan 2025.

Peng, B., Quesnelle, J., Fan, H., and Shippole, E. (2023). Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*.

Radford, A. et al. (2018). Improving language understanding by generative pre-training.

Zhang, P., Zeng, G., Wang, T., and Lu, W. (2024). Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*.