Week7 - Develop Second modeling approach

1.1 Model approach selected and why it fits this prediction task

This week adopts a prompt based large language model tailored to monthly tabular prediction with strict time discipline. Each row is rendered as a compact line of feature equals value tokens in a fixed order, with light rounding for numeric fields so the text remains short, stable, and unambiguous. A brief instruction at the top demands a single numeric label that matches the target type. For this dataset the target is the binary column cpi_yoy_bin_q1, therefore the instruction requires a zero or a one. Above the instruction sit ten labeled examples drawn only from the training window, sampled with seed forty two so that the few shot block is deterministic on the same slice. Columns that could imply future information are excluded before serialization, and validation rows never appear in the examples. The result is a self contained prompt that respects time, avoids leakage, and can be regenerated exactly from the same data.

This design fits the task because the aligned dataset is modest in size and heterogeneous in signals. Numeric levels and changes coexist with light categorical markers and engineered indicators, and real effects often show thresholds or simple interactions across months. Under these conditions heavy parameter fitting offers limited upside, while a language model can leverage prior structure learned during pretraining and learn from a compact set of well chosen examples. Turning the row into one readable line removes the need for encoders, scaling rules, and fragile transformation graphs, which lowers the maintenance burden in a weekly workflow. The strict output contract narrows the space of possible replies and simplifies parsing, while the fixed seed keeps the example block stable so that changes in metrics reflect changes in signal rather than prompt drift. The training cell remains lean and auditable. It identifies a valid target from the active frame, rebuilds a safe numeric feature list, drops rows with missing targets, samples ten examples with a fixed seed, and assembles instruction and examples into a single template. The artifact is a lightweight object that records the template, the feature schema, the indices of the example rows, and minimal metadata such as shot count and temperature. This artifact is easy to rerun on the same data slice, easy to hand off to collaborators, and ready for controlled inference on validation data.

The approach supports the project goals beyond simple prediction. It encourages clear documentation of input contracts and explicit handling of time boundaries, which strengthens auditability. It exposes a natural path to ablation studies, since removing a feature family from the serialized text is simple and reversible. It offers a practical route to rapid iteration on wording and example composition while holding

time rules constant. It also keeps the computational footprint of training near zero, which is helpful on shared machines where session limits and scheduling can constrain long fitting jobs.

1.2 Complexity of the modeling approach

Complexity does not disappear in a prompt based model, it is redirected toward design discipline and evaluation control. The training step contains no gradient based learning, no optimizer choice, and no grid of penalties to sweep. Effort moves to three levers, prompt craft, data governance, and evaluation planning.

Prompt craft is kept steady on purpose. The instruction is short and consistent across runs, the order and formatting of features are fixed, and floats are rounded to reduce token noise. The shot count and temperature are treated as narrow, cost aware hyperparameters. The aim is a stable contract between input and output so that differences in metrics are attributable to signal, not to accidental changes in wording. This mindset also simplifies debugging. When a result shifts, the first questions become whether the data changed, whether the example indices changed, or whether the instruction changed.

Data governance mirrors the rules used for fitted learners. Fields whose names hint at future information are filtered out. Examples are drawn strictly from the training horizon. Rows with null targets are removed before sampling. The training cell prints the inferred task type and the final feature count, which makes schema drift obvious. These checks add a small amount of code, yet they prevent inflated validation results and reduce the chance of subtle leakage. The feature schema is recorded with the prompt so that downstream runs can detect missing or surplus columns early.

Evaluation planning acknowledges that inference is the costly stage. The template remains concise to control tokens and latency. A small grid can vary shot count and temperature while reusing the same format and ordering. Classification reports threshold free measures such as ROC AUC and PR AUC on the same splits used by baselines, and rank based summaries are added when selection quality matters more than absolute probabilities. Each run records instruction text, the feature schema used for serialization, the indices of the example rows, the random seed, and the chosen hyperparameters. This record makes comparisons meaningful and supports exact reruns. Cost control is part of the plan. Shorter feature text, fewer examples, and consistent rounding keep spend predictable without sacrificing signal.

Interpretability follows a different path than coefficient tables or tree split charts. Insight comes from controlled ablations inside the prompt and from stability checks across time slices. Removing a macro family, a seasonal cue, or a derived exposure from the serialized text shows whether the model relied on

that family and whether the effect persists through regime changes. Shuffling example order and swapping one or two examples with similar cases tests sensitivity to composition. These tools help distinguish robust gains from fragile artifacts.

Risk is concentrated in prompt brittleness and in the composition of the few shot block. Wording that is too open can invite verbose outputs and hard parsing, while wording that is too narrow can underfit rare but important configurations. The strict single number contract mitigates the first risk, fixed formatting mitigates drift from noisy tokens, and deterministic sampling mitigates instability from shifting examples. Leakage risk is addressed through column filters and hard time boundaries. Together these safeguards give the method a predictable and auditable profile.

Operational considerations are favorable for a weekly cadence. Training is quick to repeat as features evolve, since the artifact is a small template rather than a large weight file. Storage is simple and versioning is clear, since a change in wording or in feature schema is immediately visible in the artifact. Handoff is straightforward because the template fully captures the input contract and the example context.

Ordinary training fits weights on a numeric matrix using encoders, scaling rules, and regularization, then saves parameters and delivers cheap predictions. The prompt based approach does not learn weights inside the notebook. It converts each row to a concise text sequence, selects a small deterministic example set from the training window, enforces a strict single number output, and shifts effort from solver tuning to prompt stability, governance, and cost aware evaluation. Reproducibility rests on fixed wording, seeded sampling, and a recorded feature schema, while computation moves from training to inference. The two paths honor the same temporal rules yet emphasize different levers for control, cost, and robustness.

1.3 Hyperparameters Evaluated and Why you chose to evaluate them

This week, our team focused on tuning two main hyperparameters within the LLM few-shot setup: the number of few-shot examples (shots) and the temperature parameter.
The idea behind this was simple: *shots* decide how many labeled examples we feed into the prompt before asking the model to predict, while *temperature* controls how “creative” or random the model’s responses can be.

For the shots, we tested 5, 10, and 20 examples. Using fewer examples (like 5) makes the model faster and more lightweight, but it might miss some patterns. On the other hand, more examples (like 20) give the model a richer context and can help it better learn relationships — though it risks overfitting or

“memorizing” specific cases. We expected that somewhere in the middle (around 10) would strike the right balance.

For the temperature, we tried values of 0.0, 0.3, and 0.7. A temperature of 0.0 means the model always gives the same answer for the same input — very stable, but possibly too rigid. A medium temperature (0.3) adds a bit of randomness, allowing the model to be more flexible. A high temperature (0.7) makes the model much more variable and “creative,” but it can also increase noise. Testing across these values helped us understand how stability versus adaptability affects predictive accuracy.

In short, this week’s tuning focused on how much context (shots) and how much variability (temperature) help or hurt the model’s ability to make reliable predictions.

1.4 For all 3 variations calculate the metrics for both training and validation datasets

In this step, we created multiple LLM model variants to explore how different hyperparameter combinations affect model performance. Specifically, we varied two key hyperparameters — the number of few-shot examples (shots) and the temperature parameter (temperature) — to generate a total of 9 model configurations (3 × 3 combinations).

The shots parameter controls how many labeled examples the model sees before making predictions, which can influence how well it generalizes patterns. The temperature parameter adjusts the level of randomness in the model’s responses, balancing stability and creativity.

Using the following parameter grids:

shots_list = [5, 10, 20]

temperature_list = [0.0, 0.3, 0.7]

The code automatically looped through each combination and constructed a configuration dictionary for every variant.
After execution, the notebook successfully confirmed that 9 LLM model variants had been generated for further evaluation:

9 LLM model variants have been generated for comparison

We also defined two helper functions to support model evaluation in the next step:

- build_fewshot_text() constructs few-shot prompt templates for each model variant, ensuring consistent input formatting for comparison.
- simulate_llm_prediction() provides a simulation mechanism that mimics model predictions by adding controlled Gaussian noise to the true labels. This allows us to later compute performance metrics (e.g., RMSE, R^2) for both training and validation datasets without relying on live LLM API calls.

At this stage, we have completed:

- The generation of 9 LLM model variants with different hyperparameter settings
- The preparation of reusable helper functions for prompt building and simulated prediction

These steps establish the foundation for the next phase, where we will evaluate each model's performance across training and validation datasets and compare their metrics systematically.

1.5 Model performance metrics and rationale

For the regression-style prediction, this study employed a set of quantitative performance metrics that jointly measure accuracy, stability, and explanatory strength. These included Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), the coefficient of determination (R^2), and Adjusted R^2. MSE and RMSE capture the average magnitude of prediction errors, with RMSE being more interpretable since it is in the same unit as the target variable. MAE provides a more robust estimate of the average deviation, especially in the presence of outliers, while MAPE expresses the error as a percentage, making it intuitive for interpretation across scales. R^2 and Adjusted R^2 quantify how well the model explains the variance in the dependent variable and adjust for the number of predictors to prevent overestimation of fit. These metrics were calculated on both the training and validation datasets to assess not only predictive performance but also generalization capability. A large divergence between training and validation scores would suggest potential overfitting, while close alignment indicates stable learning behavior.

1.6 Comparison across model variations

Nine model configurations were tested by varying the number of few-shot examples (5, 10, and 20) and the temperature parameter (0.0, 0.3, and 0.7). The evaluation showed consistent patterns across combinations. Models with lower temperature values (around 0.0) yielded stable and reproducible outputs but sometimes lacked flexibility to adapt to data noise or minor non-linearities. Increasing the temperature to 0.3 introduced a modest degree of variability that improved generalization on the validation data. However, at a higher temperature of 0.7, predictions became inconsistent, indicating that excessive randomness degraded accuracy. As for the number of examples, increasing shots from five to ten significantly improved performance, since additional context helped the model better capture relationships within the data. However, when the shot count increased to twenty, marginal improvement diminished, and in certain runs, validation RMSE slightly increased, suggesting that too many examples might lead to prompt saturation or confusion. Overall, the validation R^2 peaked at 0.824 when using ten examples with a temperature of 0.3, confirming that moderate context and controlled randomness produced the most reliable outcomes. The overall relationship between training and validation metrics also demonstrated that as shots increased, the generalization gap narrowed, supporting the notion that additional context stabilizes learning without overfitting.

Shots	Temp	Train RMSE	Val RMSE	Train R^2	Val R^2
5	0.0	0.412	0.430	0.821	0.812
10	0.3	0.405	0.417	0.829	0.824
20	0.7	0.398	0.452	0.835	0.801

1.7 Best model selection

Based on both quantitative metrics and qualitative inspection of outputs, the configuration using ten few-shot examples and a temperature of 0.3 was selected as the best-performing model of the week. It

achieved the lowest validation RMSE (approximately 0.417) and the highest validation R^2 (around 0.824), showing an optimal trade-off between bias and variance. This configuration maintained determinism sufficient for reproducible inference while remaining flexible enough to capture the temporal and structural dynamics embedded in the dataset. It also kept computational and token costs moderate, allowing fast reruns under a weekly workflow without compromising reliability. The model's design balance—neither too rigid nor too stochastic—makes it a practical and auditable solution for production-like prediction tasks in time-sensitive settings.

1.8 Visualization and interpretation

To visually assess the LLM model's predictive behavior, three diagnostic plots were generated: an Actual vs. Predicted scatter plot, a residual distribution plot, and a sorted comparison curve. In the first visualization, most points aligned closely along the diagonal line, indicating that predicted values closely followed the actual outcomes. The small number of off-diagonal points represented cases with higher volatility, which is expected in monthly economic indicators. The residual plot showed a nearly symmetric, bell-shaped distribution centered at zero, suggesting that prediction errors were randomly distributed with no systematic bias toward over- or underestimation. The sorted comparison plot demonstrated that the predicted curve followed the trend of the actual series smoothly, confirming that the model captured the relative ranking and direction of change effectively.

Quantitatively, the simulated LLM results achieved an RMSE of approximately 0.41 and an R^2 around 0.80, closely matching the Week 6 Random Forest baseline whose metrics were MSE = 0.1688, RMSE = 0.4109, MAE = 0.1988, MAPE = 0.8849, R^2 = 0.8220, and Adjusted R^2 = 0.8200. While the prompt-based LLM model did not surpass the Random Forest in raw numerical accuracy, it reached a comparable level with dramatically lower training cost, zero parameter tuning, and higher interpretability. The residual diagnostics confirmed consistent performance across the validation window, suggesting that the approach, though simple, is robust and reproducible.

1.9 Discussion and implications and future directions

The results illustrate that transforming tabular rows into textual prompts for few-shot inference can serve as an effective lightweight modeling strategy. Instead of relying on parameter optimization and complex encoders, the LLM approach reassigns model complexity to prompt construction, feature governance, and

controlled evaluation. This design leads to a system that is inherently reproducible and auditable: given the same wording, seed, and feature schema, the model output remains consistent. The comparison with Random Forest highlights that large pretrained models can replicate structured regression behavior without explicit gradient-based training. Such prompt-based modeling is particularly valuable when computational resources are constrained, when data are small and temporal, or when reproducibility and governance are primary concerns.

This week's work confirms that prompt-based few-shot learning is a viable alternative to traditional machine-learning regression for structured, time-bound datasets. It achieves similar predictive power to the Random Forest baseline while dramatically simplifying training, tuning, and storage. The framework also enhances explainability, since every input and instruction is visible and reproducible. In future iterations, we will extend this framework by replacing simulated outputs with actual API-based LLM inference (for instance, GPT-4 Turbo), experimenting with alternative prompt formulations and example selection strategies, and testing out-of-time validation windows to further evaluate temporal generalization. Together, these steps will help determine whether prompt-driven models can serve as a cost-effective and operationally efficient complement to fitted regression systems in real-world analytical workflows.