A Failure Modes

The design of the prompt plays a pivotal role in our entire process. Specifying instructions precisely can create a significant difference, whether it comes to effectively describing tabular data, generating reliable summaries or inferring accurate predictions (shown in Figures 1 and 2). While soft prompting has been successful at instructing LLMs [23], it cannot be applied to our setting because our classification algorithm learns a human-level summary as a prompt for classifying data, rather than a soft prompt. Instead we chose to write prompts that ask the LLM to perform these tasks. In this way, our prompting method is entirely gradient-free. Hand-written prompts also offer flexibility, aligning well with our core methodology for creating weak learner. While carefully handcrafting prompts this way might seem intensive, we will show that once we identify the ideal hyperparameter settings they can be framed with little effort.

A.1 Data Conversion Challenges

Language models (LLMs) have demonstrated impressive performance on standard tasks with minimal supervision [42, 2]. However, for converting tabular data to text, there were several considerations to meet the requirements of our task. As highlighted in Section 3.1, we will refer to these texts as *data descriptions*.

Ensuring Uniform Length Firstly, the data descriptions should not be too long or short, also be of comparable length. Excessively long descriptions limit the number of examples that can be fit inside the prompt and summarized. We observed in Figure 4 (right bottom) that the summary performance also scales with more examples, so it makes sense to have descriptions of approximately uniform length.

A straightforward way of achieving this uniformity would be by specifying a max word length as part of the conversion prompt itself, as in “Describe in not more than 80 words”. However, we found this approach can falter, sometimes leading to overly simplistic descriptions like "These are annual spendings of a customer." (in the `wholesale-customers` dataset).

Consequently, we adopt more nuanced strategy by first modifying the prompt with the terms “concisely” and “accurately” to emphasize the brevity and preciseness of the generated descriptions (shown in Figure 1). Then, we implement a resampling strategy, that generates descriptions until finding the one with a desired length ranging between 20 to 80 words. This process achieves consistent and uniformly long descriptions.

Including Metadata Prepending metadata to the prompt enhances the contextual awareness of the task, resulting in higher-quality descriptions (shown in Figure 1).

Separating Features from Labels In our method, the data descriptions function dual role, both as training examples and as query for inferring class labels. This suggests that, when converting data to text, the features need to be described separately from the target label as illustrated in Figure 1. The resulting strings are then concatenated to form the data description. Instead, if the whole tabular record were passed to the LLM, it often produces texts that assimilate the classification label information in the meat of the description itself, rendering it difficult to extract a query for doing inference.

While one might cleverly come up with prompts that can allow the LLM to describe the features and target label in separate sentences, we found it to be more sensible to supply just the features for describing and not reveal any information about the target task. Sometimes that can liberate the LLM to hallucinate some facts about the task and form biased data to begin with.

Natural-Sounding Descriptions While the LLM generates a different-styled response every time, to explicitly ensure that the generated descriptions are not template-like by chance, add a directive at the end of the prompt: “Use your creativity”. This encourages the LLM to produce more natural narratives of the record. Alternatively, setting a higher temperature during decoding achieves a similar effect.

A.2 Summarization

There are several aspects worth considering that can contribute to high-quality summaries.

Sampling candidate summaries A well-crafted summary is a one that captures salient information of the dataset, in a way that facilitates inferring predictions off it. However, the process of generating summary using a LLM is inherently stochastic due to temperature sampling, as a result, the generated summary can be noisy. From our experiments with tuning this temperature, we found 0.80 to be ideal through Bayesian optimization. Even at this value, on average only 1 out of 3 summaries were meaningful.

A noisy summary can be distinguished quite easily. For instance, on the `vehicle` dataset, the `tl;dr` prompt elicits summaries as naive as “The given data describes a bus, van, or saab silhouette.” or “The data in this table identifies a vehicle as a bus, saab, opel, or van. The compactness, circularity, distance circularity, radius ratio, hollows ratio, and symmetry are all predictive of the vehicle’s type.” which does not offer actionable insight.

This observation indicates that summaries need to be sampled quite a few times and the best one can be determined based on the validation error. As a result, for the `Summary` learning procedure in Section 3.2, we resample approximately 25 times to find a good summary. Also, given that our datasets are small, it is not unusual for the summaries to have the same validation error. When tied, we pick one having a higher training error rate, i.e. lower generalization gap.

Differently, in our `Summary boosting` procedure explained in Section 3.3, we resample only until finding a summary whose training error is better than random guessing and return immediately.

Ordering examples inside the *summarization* prompt Unlike gradient descent, prompting is not robust to the presentation of the examples to the learning algorithm. While we show via ablation studies in Section 5 that there is no statistically significant difference in performance between either shuffling the examples or listing them by class, we can generally expect that depending on the dataset and the number of target classes, one might be preferred over the other.

For instance, in a multi-class setting, listing examples by class might be more helpful in reaching a weak learner quickly. However, in a two-class setting, the summary might actually benefit from the randomness in shuffled examples.

Customizing the *summarization* prompt The approach of asking the LLM to summarize examples can also give rise to good/bad summaries. For instance, one can prompt the LLM with a simple `tl;dr` or specify the task more elaborately. We will refer to the latter option as `explicit`. As we demonstrated in Figure 4 (left), both are means to the goal and do not statistically differ in terms of performance induced.

However, in our experiments on certain datasets, we would rather be incentivized choosing the `explicit` over the `tl;dr` to attain a weak learner more quickly. This choice becomes important purely for compute reasons as it will take relatively lesser resampling, while the `tl;dr` still works. For instance, this scenario can happen when the LLM cannot decipher what the summary is supposed to say, by just observing the examples. As examples, the `tl;dr` prompt suffices on datasets such as `iris`, `diabetes`, and `wine` that are commonly encountered in prediction context, whereas the LLM might not be very familar with the goals of `vertebra-column` or `somerville-happiness-survey` data, necessitating the use of the `explicit` prompt. For these other datasets, the fact that it is a classification problem based on some features and target classes may not be very apparent from just the examples and metadata. So, providing a directive such as “Summarize in detail how we can tell apart people with normal and abnormal vertebra-column” reduces ambiguity in the task setup and reduces probability of a noisy summary.

While manual intervention is necessary, framing this prompt can be done with little effort. We provide a comprehensive list of these parameters for all datasets in Table 3.

Including Metadata Similar to data conversion, including meta-data information in the prompt offers better penetration into the world of the dataset, as a result improves boosting performance.

Table 3: **Prompt design:** Prompt parameter settings for every dataset.

Dataset	**Prompting hyperparameters**
caesarian	*metadata:* This dataset contains information about caesarian section results of 80 pregnant women with the most important characteristics of delivery problems in the medical field.The goal is to predict whether a woman will undergo normal or caesarian delivery. *classes:* [normal, caesarian] *summary directive:* Tl;dr *inference directive:* Hence this woman's delivery mode is likely to be (normal or caesarian):
iris	*metadata:* This is the iris dataset, perhaps the best known database to be found in the pattern recognition literature. Fisher's paper is a classic in the field and is referenced frequently to this day. (See Duda & Hart, for example.) The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.Predicted attribute- class of iris plant- setosa, versicolor, virginica *classes:* [setosa, versicolor, virginica] *summary directive:* Tl;dr *inference directive:* Based on the above information, predict if this flower will be classified as setosa, versicolor, virginica
tae	*metadata:* The data consist of evaluations of teaching performance over three regular semesters and two summer semesters of 151 teaching assistant (TA) assignments at the Statistics Department of the University of Wisconsin-Madison. The scores were divided into 3 roughly equal-sized categories ("low", "medium", and "high") to form the class variable. *classes:* [low, medium, high] *summary directive:* Tl;dr *inference directive:* Predict whether this class will score low or medium or high:
glass	*metadata:* This is the glass dataset from USA Forensic Science Service; 6 types of glass; defined in terms of their oxide content (i.e. Na, Fe, K, etc). The study of classification of types of glass was motivated by criminological investigation. At the scene of the crime, the glass left can be used as evidence...if it is correctly identified! *classes:* [building_windows_float_processed, building_windows_non_float_processed, vehicle_windows_float_processed, containers, tableware, headlamps] *summary directive:* Tl;dr *inference directive:* There are 6 possible type of glass: building_windows_float_processed, building_windows_non_float_processed, vehicle_windows_float_processed, containers, tableware, headlamps. Predict which one will this sample be:
breast-cancer	*metadata:* This is one of three domains provided by the Oncology Institute that has repeatedly appeared in the machine learning literature. This data set includes 201 instances of one class and 85 instances of another class. The instances are described by 9 attributes, some of which are linear and some are nominal. It contains information about women that had a recurrence or non-relapse of breast cancer after their first time. *classes:* [recurrence, non-relapse] *summary directive:* Based on the above examples, figure out under what conditions will a woman have recurrence or non-relapse of breast cancer? *inference directive:* Predict whether this woman will have a recurrence or non-relapse:
visualizing-environmental	*metadata:* This is the visualizing-environmental dataset, one of the 22 data sets from the book Visualizing Data published by Hobart Press (books@hobart.com). This data describes indicators for a positive/negative environment based on ozone, radiation and temperature. *classes:* [positive, negative] *summary directive:* Tl;dr *inference directive:* There are clear signs of this environment being (positive or negative):
analcatdata-chlamydia	*metadata:* This chlamydia dataset is one of the data sets used in the book "Analyzing Categorical Data" by Jeffrey S. Simonoff, Springer-Verlag, New York, 2003. It contains results of individuals that tested for chlamydia. *classes:* [positive, negative] *summary directive:* Tl;dr *inference directive:* Predict if this person will test positive or negative for chlamydia:
wine	*metadata:* This is the Wine recognition data. Updated Sept 21, 1998 by C.Blake. It contains results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. *classes:* [1, 2, 3] *summary directive:* Using these examples and based on the contents of constituents, summarize what distinguishes wines of type 1 or 2 or 3? *inference directive:* Hence this wine will be classified as ->type
blood-transfusion-center	*metadata:* Data taken from the Blood Transfusion Service Center in Hsin-Chu City in Taiwan - this is a classification problem. The goal is to predict whether a given individual will consent or avoid donating blood. *classes:* [consent, avoid] *summary directive:* Tl;dr *inference directive:* Therefore, this individual is likely to (avoid/consent):
somerville-happiness-survey	*metadata:* This is the Somerville Happiness Survey Data Set. It has ratings collected from a survey of Somerville residents. From the responses of a resident, the goal is to predict whether they feel happy or unhappy about the place. *classes:* [unhappy, happy] *summary directive:* Based on the Somerville happiness survey, how can we predict whether a resident is happy or unhappy with their place? *inference directive:* So this resident is (happy or unhappy):
vehicle	*metadata:* This is the Statlog (Vehicle Silhouettes) Data Set. The purpose is to classify a given silhouette as one of four types of vehicle - bus, saab, opel or a van, using a set of features extracted from the silhouette. The vehicle may be viewed from one of many different angles. *classes:* [bus, saab, opel, van] *summary directive:* Using these examples, summarize how can we differentiate if a silhouette is that of a bus, saab, opel or a van. *inference directive:* Out of saab, bus, van and opel, this vehicle is likely to be a
statlog-heart	*metadata:* This dataset is a heart disease database similar to a database already present in the repository (Heart Disease databases) but in a slightly different form. It has data on individuals having and not having heart disease. *classes:* [present, absent] *summary directive:* Differentiate people with heart disease present from ones absent. *inference directive:* In this case, heart disease is likely to be (present/absent):
verterbra-column	*metadata:* This dataset contains values for six biomechanical features used to classify orthopaedic patients into 3 classes (normal, disk hernia or spondilolysthesis) or 2 classes (normal or abnormal). Biomedical data set built by Dr. Henrique da Mota during a medical residence period in the Group of Applied Research in Orthopaedics (GARO) of the Centre MÃ©dico-Chirurgical de RÃ©adaptation des Massues, Lyon, France. The task is to classify patients as belonging to one out of two categories: Normal (100 patients) or Abnormal (210 patients). *classes:* [abnormal, normal] *summary directive:* Based on the above examples, summarize how will you distinguish patients that have normal vs. abnormal vertebral column. *inference directive:* Therefore, this individual's vertebral column is likely to be (abnormal or normal):

ecoli	*metadata:* This data contains protein localization sites. Reference: "A Knowledge Base for Predicting Protein Localization Sites in Eukaryotic Cells", Kenta Nakai & Minoru Kanehisa, Genomics 14:897-911, 1992. *classes:* [1, 2] *summary directive:* Using these examples, how can we tell apart cells with protein localized in sites 1 and 2? *inference directive:* Hence protein localization will be at site ->
haberman-survival	*metadata:* The dataset contains cases from a study that was conducted between 1958 and 1970 at the University of Chicago's Billings Hospital on the survival of patients who had undergone surgery for breast cancer. *classes:* [survived, died] *summary directive:* Based on these examples, figure out what commonalities are predictive of patients surviving more than 5 years and less. *inference directive:* So, 5 years down the line, this person (survived/died):
diabetes	*metadata:* This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective is to predict based on diagnostic measurements whether a patient has high/low risk of developing diabetes. *classes:* [low, high] *summary directive:* Based on these examples, distinguish patients having low vs. high risk of diabetes. *inference directive:* Based on the reasoning, this patient is likely to have a (low/high):
visualizing-hamster	*metadata:* This is the visualizing-hamster dataset contains 22 data sets from the book Visualizing Data published by Hobart Press (books@hobart.com). It contains examples of hamsters that are ill and healthy. *classes:* [ill, healthy] *summary directive:* Using these examples, identify predictive indicators of ill and healthy hamsters. *inference directive:* Predict whether this hamster will be ill or healthy:
wholesale-customes	*metadata:* The data set refers to clients of a wholesale distributor. It includes the annual spending in monetary units (m.u.) on diverse product categories. This data gives information about spending patterns and region of operations of Retail and Horeca (Hotel/Restaurant/Café) customers of the wholesale distributor. *classes:* [retail, horeca] *summary directive:* Using these examples, summarize how can we differentiate Retail customers and Horeca customers. *inference directive:* Therefore, which one of Retail or Horeca this customer is likely to be:

A.3 Inference

Mapping labels from LLM responses Answer mapping refers to the process of assigning the model's answer to a target output class. This step might be trivial when the answer is the class itself, for example when the LLM responds with "non-relapse" or "recurrence" to a query on the `breast-cancer` dataset. However, in other instances, it can become tricky when the LLM's responses are "will not recur" or "has higher chance of non-relapse than recurrence", requiring a more complex decoding logic to identify the target class.

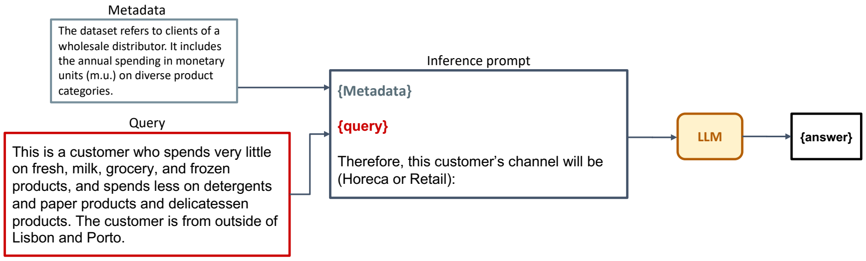
Previous works have handled this problem by disguising the task as a Cloze prompt and learning a verbalizer, i.e. MLP projection of hidden state of the **[MASK]** token, that maps the predicted token to the target classes [4, 19]. By training a verbalizer, one can determinsically go from token vocabulary to the label space. There also exist unsupervised statistical techniques for achieving label mapping [41].

In our method however, we strictly interact with the LLM through prompts and do not access the hidden state nor gradients. As a result, our inference process shown in Figure 2 focusses on inferring the class label solely through prefix prompts, without relying on learning an explicit mapping. Specifically, by conditioning on a suitable prefix, we constrain the LLM to return exactly the class label string. For example, the prefix "Therefore this iris flower is likely to be (setosa, versicolor, virginica):" works for the `iris` dataset. A key observation guiding the design of such a prefix prompt is the fact that specifying the output classes entices the LLM to predict from among these classes. With a rather plain prefix like "Predict what will be the type of this flower.", the LLM's answer space is unconstrained and it might liberally go on to explain a chain of reasoning such as "The flower has short petals and long sepals, hence it is versicolor, and not setosa." preventing a simple keyword search for the class label.

For a full list of these inference prompts, refer Table 3.

Two-step prompting, Davinci vs. Curie It is worth mentioning that a two-step prompting trick, by first calling "Lets think step by step" then concatenating the response with the prefix prompt also results in accurate answers, as we have shown in Figure 4 (left). However, it could only be implemented on the larger model `Davinci` but not `Curie` which is primarily used in our experiments. Interestingly `Davinci`'s chain of thought reasoning even outperforms its prefix prompt counterpart. In all our experiments with Curie however, the prefix technique works reasonably well.

The Davinci API also offers a suffix argument which can be invoked for predicting in a more natural way. For instance, for the `breast-cancer` dataset, the prompt can be posed with prefix "All in all, this woman is more likely to have a " and a suffix " of breast cancer." directly expecting the LLM to fill in with "recurrence" or "non-relapse."

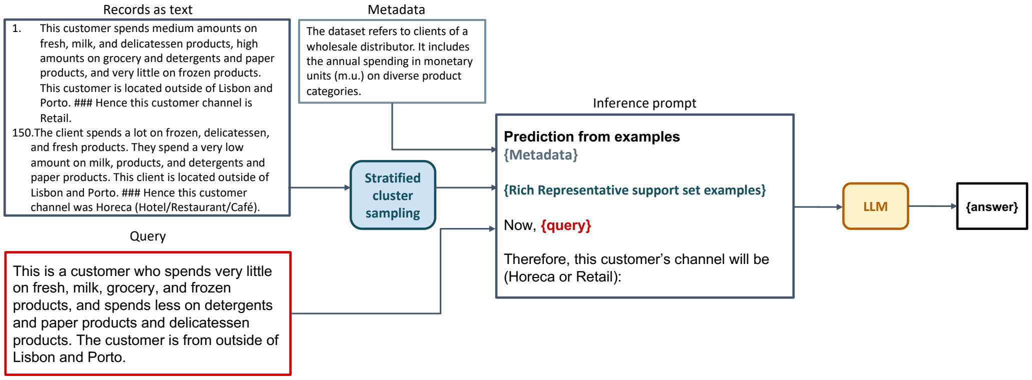


The dataset refers to clients of a wholesale distributor. It includes the annual spending in monetary units (m.u.) on diverse product categories.

This is a customer who spends very little on fresh, milk, grocery, and frozen products, and spends less on detergents and paper products and delicatessen products. The customer is from outside of Lisbon and Porto.

Therefore, this customer's channel will be (Horeca or Retail):

Figure 5: Steps of Zeroshot prompting



1. This customer spends medium amounts on fresh, milk, and delicatessen products, high amounts on grocery and detergents and paper products, and very little on frozen products. This customer is located outside of Lisbon and Porto. ### Hence this customer channel is Retail.
150. The client spends a lot on frozen, delicatessen, and fresh products. They spend a very low amount on milk, products, and detergents and paper products. This client is located outside of Lisbon and Porto. ### Hence this customer channel was Horeca (Hotel/Restaurant/Café).

The dataset refers to clients of a wholesale distributor. It includes the annual spending in monetary units (m.u.) on diverse product categories.

This is a customer who spends very little on fresh, milk, grocery, and frozen products, and spends less on detergents and paper products and delicatessen products. The customer is from outside of Lisbon and Porto.

Therefore, this customer's channel will be (Horeca or Retail):

Figure 6: Workflow in fewshot prompting

A.4 Zeroshot Setting

We extend the analysis of prompting-based methods in Section 4.1 by further delving into the `Zeroshot` experiment. We illustrate this experimental setting in Figure 5. It only consists of the inference prompt, wherein the LLM is presented with the metadata and a query. To facilitate answer mapping, the output classes are also indicated in the prompt. Unlike `Summary`, the `zeroshot` process is not stochastic as there is no learning involved. For inference, the predicted tokens are sampled greedily at temperature $= 0$.

A.5 Few-shot Setting

Extending the analysis from Section 4, we explain the `Fewshot` setting. As illustrated in Figure 6, it is an instance of in-context learning, wherein the support set examples are enlisted in the prompt along with a query in the end. The support set was chosen from the training set through the stratified cluster sampling outlined in Algorithm 1. This results in a semantically diverse collection of examples evenly spread across the classes. Since we observe in Figure 4 (bottom right) that the `Fewshot` performance drops with more examples, we choose approximately 15 examples to fit in the prompt.

Similar to the `Summary` method, this inference prompt carries meta-information about the dataset, and also indicates the output classes. The prompt also injects support set examples that are stringed together as we show in Figure 6. The predictions are made greedily at a temperature of 0.

Again, the `Fewshot` method is a stochastic process whose outcome depends on the selection of the support set. So finding the ideal prompt, requires sampling different support sets from the training set quite a few times. We perform this resampling approximately 25 times and pick the best prompt based on the validation error.

A.6 Preprocessing continuous attributes

Extending from the ablation studies in Section 5, we demonstrate in Table 4, concrete examples of these encoding techniques applied to continuous features. Every numerical column in the dataset was subject to the transformation independently of the rest.

Table 4: Continuous variable transformations applied to an example from the `wholesale-customers` dataset. The raw tabular record is as follows: spending on fresh products: 6353.0, spending on milk products: 8808.0, spending on grocery products: 7684.0, spending on frozen products: 2405.0, spending on detergents and paper products: 3516.0, spending on delicatessen products: 7844.0 and customer's region: Outside Lisbon and Porto.

Method	Data Representation	Example as text
`4 bins + quantifiers` {very low, low, high, very high}	- *spending on fresh products :* low - *spending on milk products :* very high - *spending on grocery products :* high - *spending on frozen products :* high - *spending on detergents and paper products :* high - *spending on delicatessen products :* very high - *customer's region :* Outside Lisbon and Porto	This customer spends low amounts on fresh products, very high amounts on milk products, high amounts on grocery products, frozen products, detergents and paper products, and very high amounts on delicatessen products. They are located outside of Lisbon and Porto.
`5 bins + quantifiers` {very low, low, medium, high, very high}	- *spending on fresh products :* medium - *spending on milk products :* very high - *spending on grocery products :* high - *spending on frozen products :* high - *spending on detergents and paper products :* high - *spending on delicatessen products :* very high - *customer's region :* Outside Lisbon and Porto	This customer from outside Lisbon and Porto spends medium on fresh products, very high on milk products, high on grocery products, high on frozen products, high on detergents and paper products, and very high on delicatessen products.
`7 bins + quantifiers` {extremely low, very low, low, medium, high, very high, extremely high}	- *spending on fresh products :* low - *spending on milk products :* very high - *spending on grocery products :* high - *spending on frozen products :* high - *spending on detergents and paper products :* very high - *spending on delicatessen products :* extremely high - *customer's region :* Outside Lisbon and Porto	This customer situated outside Lisbon and Porto spends low on fresh products, very high on milk products, high on grocery products, high on frozen products, very high on detergents and paper products, and extremely high on delicatessen products.
`9 bins + quantifiers` {lowest, extremely low, very low, low, medium, high, very high, extremely high, highest}	- *spending on fresh products :* low - *spending on milk products :* extremely high - *spending on grocery products :* high - *spending on frozen products :* high - *spending on detergents and paper products :* very high - *spending on delicatessen products :* highest - *customer's region :* Outside Lisbon and Porto	This customer spends low amounts on fresh products, extremely high amounts on milk products, high amounts on grocery products, frozen products, detergents and paper products, and highest amounts on delicatessen products. They are located outside Lisbon and Porto.
`10 bins`	- *spending on fresh products :* falls in the first out of ten bins of values - *spending on milk products :* falls in the second out of ten bins of values - *spending on grocery products :* falls in the first out of ten bins of values - *spending on frozen products :* falls in the first out of ten bins of values - *spending on detergents and paper products :* falls in the first out of ten bins of values - *spending on delicatessen products :* falls in the second out of ten bins of values - *customer's region :* Outside Lisbon and Porto	This customer spends relatively little on fresh, grocery, frozen and detergents/paper products, and more on milk and delicatessen products. They are based outside Lisbon and Porto.
`Percentile`	- *spending on fresh products :* falls in the forty-first percentile - *spending on milk products :* falls in the eighty-second percentile - *spending on grocery products :* falls in the sixty-fifth percentile - *spending on frozen products :* falls in the sixty-third percentile - *spending on detergents and paper products :* falls in the seventy-second percentile - *spending on delicatessen products :* falls in the ninety-eighth percentile - *customer's region :* Outside Lisbon and Porto	This customer has an annual spending of 41st percentile on fresh products, 82nd percentile on milk products, 65th percentile on grocery products, 63rd percentile on frozen products, 72nd percentile on detergents and paper products, and 98th percentile on delicatessen products, and is located outside of Lisbon and Porto.
`Standard deviation`	- *spending on fresh products :* is within one std-dev below the mean value - *spending on milk products :* is within one std-dev above the mean value - *spending on grocery products :* is within one std-dev below the mean value - *spending on frozen products :* is within one std-dev below the mean value - *spending on detergents and paper products :* is within one std-dev above the mean value - *spending on delicatessen products :* is two std-dev above the mean value - *customer's region :* Outside Lisbon and Porto	The customer has annual spending on fresh products, milk products, grocery products, frozen products, detergents and paper products, and delicatessen products within one standard deviation of the mean, except for delicatessen products which is two standard deviations above the mean. The customer is located outside Lisbon and Porto.
`Quartiles`	- *spending on fresh products :* is between the first quartile and median values - *spending on milk products :* is more than the third quartile value - *spending on grocery products :* is between median and third quartile values - *spending on frozen products :* is between median and third quartile values - *spending on detergents and paper products :* is between median and third quartile values - *spending on delicatessen products :* is more than the third quartile value - *customer's region :* Outside Lisbon and Porto	This customer spends more than the third quartile value on milk, delicatessen and detergents and paper products. The customer's spending on fresh, grocery, and frozen products falls between the median and third quartile values, while the customer is located outside of Lisbon and Porto.

We applied several encoding techniques for continuous features, including binning, percentiles, and standard deviations. Our approach involved using technical language terms to describe these ranges, such as *"falls in the nth bin/nth percentile or n deviations above/below mean"*. We also characterize them in a more naturalistic way by assigning quantifiers such as *low*, *medium*, and *high* to each level in the binning technique.

To create effective textual descriptions, we examined three high-level approaches: 1. presenting only numerical values, 2. using solely textual encodings, and 3. concatenating both. We observed that utilizing textual encoding alone outperformed the other methods. As a result, we focused on mainly comparing textual encoding methods as shown in Figure 4 (right bottom). Through Bayesian optimization, we found that binning with "5" quantifiers was ideal for generating high-quality summaries.

We describe each encoding technique as follows:

- **Binning**: It involves creating a histogram with the given number of bins. As outputs, the values are directly described as *"falling in the n-th bin"* as illustrated in the `10 bins` experiment. However, in the presence of *degree quantifiers* which are categorical names assigned to the bins, these tags are used instead. We found that as opposed to calling out the bin number, describing in terms of these quantifiers further aids the LLM in comparing the relative extent to which features match and improving the estimation of similarities. This led us to tune the number of bins against these degree quantifiers, selecting values in the range of 4, 5, 7, and 9 bins. The first four rows in Table 3 show how these tags get translated into the record.
- **Percentile**: It is given by computing the percentile rank of a value relative to that series of values. Then, the value is described as falling in that percentile rank in words. This is closer to representation of the true numerical values per se, but helps the LLM draw comparisons on a scale of 1-100.
- **Standard deviations**: In this procedure, the values are segmented into six ranges based on distance from the mean, given by one/two/three standard deviations above/below the mean.
- **Quartiles**: Here, we consider the first and third quartiles, and the median as landmarks to bucketize the values into four partitions.

Among these methods, the "5 bins with quantifiers" strikes a balance in granularity scale. It is not excessively fine-grained as "percentile", nor overly abstract, as the "4-bin" approach. This balance ultimately leads to optimal performance.

A.7 Clustering Sampling components

We discuss more of the functions in Algorithm 1.

GPT-Embedding is OpenAI's text similarity model `text-embedding-ada-002` that takes a maximum input size of 8191 tokens. It returns a 1536-dimensional embedding for text. OpenAI recommends cosine distance for comparing *ada* embeddings in downstream tasks.

As a result, the **AgglomerativeClustering** algorithm applies hierarchical clustering over these features using cosine distance, average linkage and a heuristically selected distance threshold of 0.05. It yields a set of clusters C and each C_j contains a list of indices of data points that belong to that cluster j.

A.8 Adaboost Optimizations

We additionally apply several run-time optimizations to the boosting algorithm described in 3.3. Thus we present its full version in Algorithm 3.

- **Raising the bar for a weak learner:** Our goal was to create high-quality summaries that dramatically reduce the validation error rate and significantly accelerate the convergence of the boosting procedure. Thus we raise the performance threshold to a notch slightly higher than random guessing probability (see Step 15 in Algorithm 3), provoking insightful summaries.

  We resample until finding a weak learner that satisfies this threshold.

  The positive quantity μ is a hyperparameter that typically takes values 0.08 for 2-class problem and 0.16 for 3-class problem, and so on.

  Although this step increases compute, it yields better weak learners and improves convergence overall.
- **Permuting the predicted class label assignments:**

  We harness the potential of permuting the class assignments by exploring $K!$ different mappings of predictions to classes using the **PermutedLabelMappings** function in steps 11-14. This process helps us identify the mapping that minimizes the training error to the greatest extent.

  By considering multiple permutations of predictions across the label space, as outlined in Steps 11-14 of Algorithm 3, we obtain a hashmap p from the **PermutedLabelMappings** function. This hashmap maps the predictions $\hat{y}$ to the permuted label space. Selecting

Algorithm 3 Summary Boosting

```
1: Input: $X$, all training data; $y$, all training label; T: maximum number of rounds; s: size of the sampled subset.
2: h, P, $\epsilon$, $\alpha \leftarrow$ empty array of size T.    ▷ h holds the hypotheses, P are the corresponding label mappings, $\epsilon$ gathers the weighted train errors, and $\alpha$ are coefficients of the hypotheses.
3: N $\leftarrow$ len($X$)
4: c $\leftarrow$ set of target classes
5: K $\leftarrow$ len(c)
6: w $\leftarrow$ new array of size N filled with $\frac{1}{\mathtt{N}}$.    ▷ w is the weighted data distribution
7: for r = 1 to T do
8:     $(X_s, y_s) \leftarrow$ Cluster-sample $s$ examples from training data using distribution w.
9:     h[r] $\leftarrow$ Summary $(X_s, y_s)$    ▷ h[r] is the weak learner in the current round
10:    $\hat{y} \leftarrow h[r](X[i])$    ▷ $\hat{y}$ refers to predictions on training set
11:    $\xi \leftarrow$ empty hashmap    ▷ $\xi[p]$ will have error rate of the corresponding label mapping $p$
12:    for $p$ in PermutedLabelMappings(c) do
13:        $\xi[p] \leftarrow \frac{\sum_{i=1}^{N} \mathtt{w}[i] \times \mathbb{1}\{p[\hat{y}[i]] \neq y[i]\}}{\sum_{i=1}^{N} \mathtt{w}[i]}$
14:    end for
15:    $p^* \leftarrow \arg\min_p \xi[p]$
16:    if $\xi[p^*] > 1 - \frac{1}{K} - \mu$ OR AllSame($\hat{y}$) then
17:        Resample $(X_s, y_s)$ and Goto Step 8.
18:    else
19:        P[r] $\leftarrow p^*$; $\epsilon$[r] $\leftarrow \xi[p^*]$
20:    end if
21:    if $\epsilon$[r] == 0 then
22:        Break
23:    end if
24:    $\alpha[\mathtt{r}] \leftarrow \log\left(\frac{1-\epsilon[\mathtt{r}]}{\epsilon[\mathtt{r}]}\right) + \log(\mathtt{K} - 1)$
25:    for i = 1 to N do
26:        $\mathtt{w}[i] = \mathtt{w}[i] \times \exp(\alpha[\mathtt{r}]\mathbb{1}\{\mathtt{P[r]}[h[\mathtt{r}](X[i])] \neq y[i]\})$
27:    end for
28:    w $\leftarrow$ Normalize(w)
29: end for
30: Return h, $\alpha$
```

the mapping that results in the lowest training error effectively diminishes the cumulative training error during boosting iterations and proves to be an effective strategy for generating strong weak learners. This technique is particularly advantageous in scenarios involving more than two classes.

- **Sanity checks:** Finally, to ensure robustness of the weak learner when faced with skewed datasets, we have implemented a policy that disallows a naive all-ones classifier. The condition calling **AllSame** in Step 15 of Algorithm 3) performs this check.

A.9 Text Templates

In the ablation study which involves masking the attribute names, as illustrated in Figure 3 (second from left), we transform the descriptive attributes into the textual format by applying a pre-defined template. In Table 5 we provide examples of these templates for selected datasets.

A.10 Complexity Analysis

We provide the time complexity analysis comparing our boosting procedure to finetuning the LLM.

For finetuning, the complexity is $\mathcal{O}(TNf)$, where f is runtime of the LLM, T is number of epochs, N is the number of data points.

For summary boosting, the complexity is $\mathcal{O}(TRf)$, where f is runtime of the LLM, T is number of boosting rounds and R is the number of resampling per round.

Table 5: **Templatized descriptions:** Templates used to format examples for the ablation study between LLM-created data descriptions vs. template descriptions

Dataset	**Descriptive attribute values**	**Template**
caesarian	*age:* [very young, young, middle-aged, old, very old] *delivery_number:* [first, second, third, fourth, fifth] *delivery_time:* [timely, premature, latecomer] *blood_pressure:* [low, normal, high] *heart_problem:* [has, doesn't have] *delivery_mode:* [normal, caesarian]	This *{age}* woman is in her *{delivery_number}* delivery and it is *{delivery_time}*. She has a *{blood_pressure}* blood pressure and *{heart_problem}* heart problems. ### Based on these attributes, this woman is likely to deliver by *{delivery_mode}*
iris	*sepal_length*, *petal_length*: [very short, short, medium length, long, very long] *sepal_width*, *petal_width*: [very narrow, narrow, medium width, wide, very wide] *flower_type:* [setosa, versicolor, virginica]	This iris flower has *{sepal_length}* and {sepal_width} sepals. It also has *{petal_length}* and *{petal_width}* petals. ### Hence this flower is a *{flower_type}*
vertebral-column	*pelvic_incidence*, *pelvic_tilt*, *lumbar_lordosis_angle*, *sacral_slope*, *pelvic_radius, grade_of_spondylolisthesis:* [very low, low, medium, high, very high] *result:* [normal, abnormal]	This patient has a *{pelvic_incidence}* pelvic incidence, *{pelvic_tilt}* pelvic tilt, and *{lumbar_lordosis_angle}* lumbar lordosis angle, *{sacral_slope}* sacral slope, *{pelvic_radius}* pelvic radius and *{grade_of_spondylolisthesis}* grade of spondylolisthesis. ## As a result, the patient's vertebral-column is likely to be *{result}*
statlog-heart	age: [very young, young, middle-aged, old, very old] *sex:* [male, female] *chest_pain_type:* [asymptomatic, nonanginal pain, atypical angina, typical angina] *bp, cholesterol, st_depression, heart_rate, num_major_vessels:* [very low, low, medium, high, very high] *fasting_blood_sugar:* [high, low] *electrocardiographic_results:* [having left ventricular hypertrophy, normal, having ST-T wave abnormality] *slope_st_segment:* [flat, upsloping, downsloping] *exercise_induced_angina:* [has, do not have] *defect_type:* normal, reversible, fixed *presence_of_heart_disease:* [present, absent]	This individual is a/an *{age}* *{sex}* with *{chest_pain_type}* chest pain, *{bp}* resting blood pressure, and *{cholesterol}* serum cholesterol. Their fasting blood sugar *{fasting_blood_sugar}* >120 mg/dl, they are *{electrocardiographic_results}* and a *{heart_rate}* maximum heart rate. They *{exercise_induced_angina}* exercise-induced angina, and have a *{st_depression}* ST depression induced by exercise relative to rest. Their peak exercise ST segment has a *{slope_st_segment}* slope, and they have a *{num_major_vessels}* number of major vessels. The defect type is *{defect_type}*. ### Hence heart disease is likely to be *{presence_of_heart_disease}*.
haberman-survival	*age_at_time_of_op:* [very young, young, middle-aged, old, very old] *year_of_op:* [1964, 1962, 1965, 1959, 1958, 1960, 1966, 1961, 1967, 1963, 1969,1968] *num_pos_axillary_nodes:* [very low, low, medium, high, very high] *survival_status:* [survived, died]	This patient was *{age_at_time_of_op}* at the time of operation in *{year_of_op}*. They had a *{num_pos_axillary_nodes}* number of positive axillary nodes detected. ### Therefore 5 years down the line, the patient *{survival_status}*

Concretely, for a dataset with 175 examples, finetuning takes 20 epochs × 175 examples × 2 = 7000 passes through the LLM. 2 stands for both forward and backward passes through the model.

For the same dataset boosting requires 50 rounds × 25 resampling on average = 1250 passes through the LLM.

Thus, we believe the complexity of our algorithm is at least comparable to, if not better than, that of finetuning (without considering the cost of the actual API calls).

A.11 Estimating the cost of API calls

While our method is applicable to any large language model (LLM), we primarily conducted experiments using GPT-3. Each API call to GPT-3 incurs a specific dollar cost.

After analyzing the running time complexity of summary boosting, which is $\mathcal{O}(TRf)$, we can provide a rough estimation of the cost associated with training a classifier on any given dataset.

To begin, when making a call to summarize examples, the prompt is filled up to the maximum context length, which is 2048 tokens for the query prompt and completion. We'll refer to these summary tokens as $S_t = 2048$.

Additionally, if N represents the size of the dataset and we allocate (50 + 10)

Now, to obtain a weak learner at boosting round r, we may need to resample up to R candidate summaries. Furthermore, we calculate the training error for each candidate summary to determine if it performs better than random guessing. Once the desired weak learner is found, we compute the validation error for that round only once. Therefore, each round requires querying $R \times (S_t + 0.5N \times P_t) + 0.1N \times P_t$ tokens.

Considering that the maximum number of rounds is denoted as T, the total number of tokens exchanged would be $T \times [R \times (S_t + 0.5N \times P_t) + 0.1N \times P_t]$.

For instance, let's consider a dataset with 175 examples. In this case, the cost would be 30 rounds × [20 resampling × (2048 summary tokens + (0.5 × 175 training examples) × 210 prediction tokens) +

Table 6: Comparing test error rate of `Summary Boosting` backended by Curie and ChatGPT on all datasets (↓). Refer to caption of Table 1 for the notations.

Dataset	**Data Type**	**Size**	`Curie`	`ChatGPT`
caesarian [cae] (42901)	`1c4d`	80	$0.300_{\pm 0.04}$	$0.406_{\pm 0.03}$
iris (61)	`4c0d`	150	$0.193_{\pm 0.03}$	$0.083_{\pm 0.01}$
tae (48)	`1c4d`	151	$0.454_{\pm 0.03}$	$0.443_{\pm 0.04}$
glass (41)	`9c0d`	214	$0.370_{\pm 0.02}$	$0.492_{\pm 0.02}$
breast-cancer [bc] (13)	`7c5d`	277	$0.288_{\pm 0.02}$	$0.360_{\pm 0.01}$
visualizing-environmental [ve] (678)	`3c0d`	111	$0.268_{\pm 0.03}$	$0.333_{\pm 0.04}$
analcatdata-chlamydia [ac] (535)	`2c2d`	100	$0.170_{\pm 0.01}$	$0.300_{\pm 0.06}$
wine (43571)	`13c0d`	178	$0.320_{\pm 0.01}$	$0.250_{\pm 0.01}$
blood-transfusion-center [btc] (1464)	`4c0d`	748	$0.240_{\pm 0.04}$	$0.433_{\pm 0.01}$
somerville-happiness-survey [shs] [21]	`0c7d`	143	$0.350_{\pm 0.02}$	$0.430_{\pm 0.02}$
vehicle (54)	`18c0d`	846	$0.410_{\pm 0.04}$	$0.350_{\pm 0.16}$
statlog-heart [stath] [8]	`6c7d`	270	$0.430_{\pm 0.01}$	$0.370_{\pm 0.17}$
verterbra-column [vc] (1524)	`6c0d`	310	$0.262_{\pm 0.01}$	$0.669_{\pm 0.03}$
ecoli (1011)	`7c0d`	336	$0.270_{\pm 0.03}$	$0.193_{\pm 0.03}$
haberman-survival [hs] (43)	`3c0d`	306	$0.250_{\pm 0.01}$	$0.415_{\pm 0.03}$
diabetes [dia] (37)	`8c0d`	768	$0.344_{\pm 0.01}$	$0.297_{\pm 0.04}$
visualizing-hamster [hams] (708)	`5c0d`	73	$0.207_{\pm 0.00}$	$0.400_{\pm 0.08}$
wholesale-customers [wc] (1511)	`6c1d`	440	$0.330_{\pm 0.00}$	$0.199_{\pm 0.04}$

(0.1 × 175 validation examples) × 210 prediction tokens] = 12364050 tokens, which approximately costs $25 for Curie at a rate of $0.002/1K tokens.

A.12 Can ChatGPT function as a weak learner?

One would expect that it is more advantageous to try newer LLMs such as ChatGPT that produce increasingly more human-like text and are far more sample-efficient, i.e. can summarize more examples since they come with a larger context length. To investigate this, we conduct experiments by feeding ChatGPT with the same tabular data descriptions and using identical prompts to create weak learners. The results are presented in Table 6.

Surprisingly ChatGPT outperforms Curie in classifying datasets with more numerical features, such as `wine`, `wholesale-customers`, and `iris`. This observation suggests that LLMs are becoming more adept at quantitative reasoning from finetuning with more data. However, the reinforcement learning from human feedback (RLHF) [30] poses a limitation as it still ensures the generated text does not deviate too much from its prior. The generated text distribution adheres closely to the behavior programmed into the LLM induced by optimizing with such a reward model. Consequently it becomes challenging to bias the LLM with adversarial examples that might occasionally emerge in the training set.

For example, ChatGPT does not mostly generalize well on datasets with medical information such as `verterbra-column`, `breast-cancer`, `caesarian` and `blood-transfusion-center` where there can be examples contrary to common medical beliefs. In these cases, the RLHF is more restrictive due to its conformity to human preferences and does not neutrally summarize examples at hand from a classification standpoint. However, boosting imposes a significantly higher penalty on examples that the model fails to classify correctly, causing ChatGPT to not decrease training error after a few epochs. While these models exhibit promise in terms of higher-order problem-solving skills, their capabilities can also be limited by their alignment with human preferences.