Columbia Business School

Columbia CaseWorks

ID#260201

PUBLISHED ON
AUGUST 5, 2025

A Tale of Two AIs: Using GenAI to Fuel Machine Learning Churn Models at Blend360

BY SIMEONE DE FREMOND* AND C. DANIEL GUETTA†

Introduction

In 2024, Blend360 was midway through a project with a global manufacturer of packaging and labeling materials. Initially, the client had brought in the consultancy to manage a complex migration of historical order data from several legacy enterprise resource planning (ERP) systems into a modern, cloud-native architecture built on Snowflake. The client had grown through multiple acquisitions, and each acquired factory brought its own data standards, naming conventions, and technical debt. Rather than enforce consistency across all ERP systems, the client had historically opted for a lightweight solution: extracting local data into spreadsheets or reports, leaving inconsistencies unresolved.

But this patchwork approach quickly proved inadequate when the client requested a more ambitious goal: a predictive model that could flag when a customer was likely to churn—i.e., when a customer would stop ordering from this company. Unfortunately, because of the lack of consistency across ERP systems, the order data was severely fragmented. A single customer might appear as "The Coca-Cola Company" in one file, "Coke Puerto Rico" in another, and "CocaCola Inc." in a third. Brand names were inconsistently recorded, or omitted entirely. Item descriptions varied dramatically between factories, often filled with internal shorthand.

Any hope of building a reliable churn model would first require rationalizing the mess.

As they prepared to build a churn model, the Blend360 team were also readying for an internal generative AI hackathon. This case describes the challenge they faced—and ends as they go into the hackathon, considering whether large language models (LLMs) might offer a better solution than their initial attempts at standardizing data.

Author affiliation
* *MBA '25, Columbia Business School*
† *Columbia Business School, Columbia University*

Acknowledgements
This case is based on conversations with Oz Dogan, Joe Foran, Kokila Mallikarjuna, Ismail Mohammed, Mahadevan T Pratheep, and Rahul Sheoran of Blend360. The authors are incredibly grateful for their time and efforts in helping develop this case.

Blend360 Background

Founded in 2015, Blend360 is a data science and analytics consultancy that partners with Fortune 500 companies to unlock the full potential of their data. Headquartered in the United States, the company has grown rapidly, expanding its presence across North America, Europe, India, and Latin America. Blend360 is known for combining deep technical expertise with strategic insight, and for embedding its teams directly within client environments to ensure lasting impact.

The firm works at the intersection of data science, technology, and business performance. Its offerings span AI and machine learning, business intelligence, cloud data engineering, data governance, and modern data stack implementation. Blend360 is particularly focused on helping companies operationalize AI—turning experimentation into scalable, production-grade systems that drive measurable results.

Examples of past projects include migrating data from on-premise ERP systems to cloud-native environments, using machine learning to measure media effectiveness, building composable customer data platforms (CDPs), deploying agentic solutions where AI systems reason and act in clients' environments, forecasting sales and supply for consumer goods companies, and designing AI-enabled tools that allow business users to query complex models.

Blend360 prides itself on its collaborative approach. Rather than building black-box solutions off-site, it works directly in the client's environment, using partners like Snowflake and Databricks. Solutions are designed to be transparent and maintainable as well as scalable.

Client Background and Business Need

Blend 360's client's core business was producing custom labels and packaging materials for large consumer packaged goods companies. In this market, it was common for customers to maintain relationships with multiple vendors and reallocate business at short notice.

The client wanted to proactively detect when a customer might shift volume to a competitor—essentially, to build a churn prediction model tailored to the dynamics of B2B manufacturing. Unlike consumer settings, where churn may be passive or unpredictable, shifts in B2B vendor relationships are often preceded by subtle behavioral cues: reduced order frequency, increased complaints, or changes in product mix. If these signals could be detected early enough, account managers could intervene—reaching out to understand the issue, offer incentives, or escalate operational fixes before the relationship deteriorated further.

To make these interventions meaningful, the model couldn't just output a churn score but rather needed to be interpretable. Sales reps had to understand *why* a customer was at risk—whether it was a string of late deliveries, an unresolved defect report, or an unusually short negotiation—so they could tailor their outreach accordingly. In this context, accuracy alone wasn't enough; the model also had to provide actionable insight. With this knowledge, sales teams could reach out and try to retain the business before it was lost.

Unfortunately, building a churn model on top of the fragmented data in the client's system would prove an enormous challenge.

Building a Churn Model on Dirty Data

The fragmented state of the data wasn't an accident. Over the years, the client had expanded rapidly, acquiring several regional and international competitors. Each acquisition brought with it new customers, unique product lines, and, most problematically, its own ERP system. Rather than attempt an expensive and risky integration of all these systems, the company chose to maintain them separately, planning instead to harmonize reports at the summary level when needed. This approach worked well enough for financial rollups and inventory snapshots. But when Blend360 was asked to develop a churn prediction model at the customer-brand level, the cracks began to show.

Names of customers and brands had been entered manually in each system. There was no master data governance in place. Some names were formal legal entities like "Coca-Cola Inc.," others were shorthand used by plant managers, such as "Coke PR" or "CocaCola." In some systems, brand names weren't even captured in their own fields. Instead, they appeared sporadically within product descriptions—often abbreviated or surrounded by technical terms.

Blend360 knew that traditional churn models required accurate customer history over time. But with inconsistent naming, it was impossible to tie orders back to a reliable customer or brand profile. Before modeling could begin, the team had to find a way to clean and rationalize the data.

Their first attempts followed a familiar path. They tried to group close matches using string similarity measures like the Levenshtein distance—a standard approach in fuzzy matching that quantifies how "different" two pieces of text are (see Appendix A for details). But the inconsistencies were too great. While "Powerade" and "PowerAde" were caught by the algorithm, it failed on abbreviations like "PWRD," which bore little resemblance in terms of character sequences. Likewise, natural language processing libraries such as NLTK and spaCy could not reliably extract brand names from item descriptions full of acronyms, sizes, and packaging specifications.

In some cases, the problem was even more thorny—brand names were not even provided in the data and had to be extracted from long-form "item description" text.

For example, looking at a set of hypothetical label orders from Pepsi for Gatorade bottles, and Coca-Cola for Powerade bottles, we may encounter the following examples:

Variants that Could Be Resolved with String Similarity

Customer Name	Brand (Raw)	Item Description
The Coca-Cola Company	Powerade	Labels for Powerade bottles, English only
Coca Cola Inc.	PowerAde Zero	Diet Powerade bottle wraps
PepsiCo Ltd.	Gatorade	Gatorade lemon-lime shrink sleeves
Pepsi Bottling Co.	GATORADE	Gatorade shrink labels, 500ml

Variants that String Similarity May Miss

Customer Name	Brand (Raw)	Item Description
PBV Canada	GTDE	Summer 6-pack bottle sleeves for GTDE orange
Coke Puerto Rico	PWRD	Limited edition labels for Powerade, 12oz
Pepsi Intl	G.A.	Electrolyte drink wrapper—Berry Blast flavor
CocaCola Intl	P-Ade	International run of pink lemonade drink wrappers

Missing Brand Field, Inference Required

Customer Name	Brand (Raw)	Item Description
Pepsi Bottling Ventures		Sports drink label—mango flavor hydration line
CocaCola		Fruit punch drink wrapper (Powerade)
PepsiCo Ltd.		Frostberry electrolyte wrap – 500ml bottle

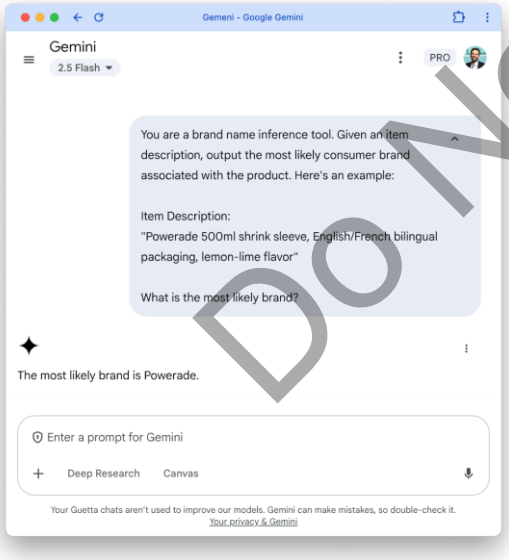
Ambiguous and Inconsistent Naming in Descriptions

Customer Name	Brand (Raw)	Item Description
Coke Puerto Rico	PWD	SLM STRW-LM wrap, 24 pack
Pepsi Intl	GatorAid	FR BERRY HYD DRNK LBL (Strwbrry)
The Coca-Cola Company		Label wrap: P-Ade STRB-LMD, Canada, EN/FR

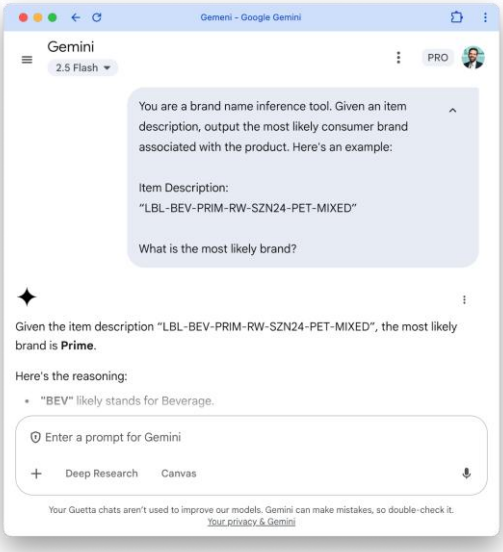
Large Language Models to the Rescue?

In the face of such a difficult data resolution problem, the Blend360 team decided to turn to large language models as a potential solution. They hoped that these models' abilities to understand and process text would make them perfectly suited to the task of cleaning the fragmented dataset.

The results were unreliable. In some cases—the easiest ones in the data—the models could successfully derive a brand from an item description:



Unfortunately, with more complex examples containing too many internal codes, technical terms, and abbreviations, the model lacked enough context to correctly identify the brand name:



This is—in retrospect—not particularly surprising. The description is so convoluted that, without additional context, it seems difficult to conceive of *any* model that could identify the correct brand.

Coincidentally, the Blend360 team were preparing to enter an internal company hackathon focused on generative AI. With this real-world challenge fresh in their minds, the team wondered: Could a large language model, if prompted the right way, offer a better solution?

They went into the hackathon determined to find out.

Appendix A - A Simplified Example

To support this case and help you experience the process the Blend360 team went through as part of building this churn resolution model, we have provided you with a synthetic dataset based on the data described above.

To simplify the in-class exercise, we will only be considering the *customer name* resolution process. For example, we will work on using a large language model to resolve names like "IBM" and "International Business Machines." It will allow us to learn how to use large language models in this context without excessive complication—the techniques for brand name resolutions are very similar.

The dataset includes the following tables:

1. **Customer Relationship Management Dataset**; a table containing one row for every active customer
    - `company_id`: a unique identifier for the company; unfortunately, this column is not provided in any other table, and therefore cannot be used to match customers across tables
    - `company_name`: the name of the company (note that different datasets may use different variations of this name to identify the company)
    - `company_address`
    - `company_contact`
    - `tel`
    - `churn`: a variable containing a 1 if the company churned (i.e., stopped being a customer) in the 60 days *following* the order, contacts, defective orders, or negotiation data, which are described below. If this column contains a 0, the customer was still a customer 60 days after the period described by the data below.
2. **Order Dataset**; a table containing one row for every order placed by a customer from January to March of a given year
    - `order_id`: a unique identifier for the order
    - `customer_name`: the raw customer name, as recorded in the original ERP system.
    - `order_date`: the date of the order
    - `order_amount`: the dollar value of the order
    - `late_flag`: a flag that indicates whether the order was late (Yes/No)
3. **Contacts Dataset**; a table containing one row for every instance in which a customer contacted the company
    - `customer_name`: the raw customer name, as recorded in the original ERP system.
    - `date_time`: the date and time of the contact
    - `type_of_contact`: the channel used for contact (e.g., phone, email, chat)

- `reason`: the purpose of contact (e.g., complaint, inquiry, reorder)

4. **Defective Orders Dataset**; a table containing one row for every order that was reported as defective by the customer
   - `order_id`: the unique identifier of the order that was defective; this will match the identifier in the order dataset
   - `date_time`: the date and time of the defect report
   - `comment`: a free-text description of the issue
   - `reported_by`: the name of the client employee who reported the defect
5. **Negotiation Dataset**; a table containing one row for every attempt the customer made to negotiate the price of an order
   - `order_id`: the unique identifier of the order being negotiated; this will match the identifier in the order dataset
   - `date_time`: the date and time of the defect report
   - `length_of_conversation`: the duration of negotiation (in minutes)
   - `customer_rep_name`: the name of the company representative
   - `outcome`: whether the negotiation was successful or not
   - `amount_negotiated`: the dollar value of the discount negotiated in the conversation.

As described in the body of the case, the data contains inconsistent and ambiguous naming conventions. For example, "The Coca-Cola Company," "CocaCola," and "Coke PR" may appear as distinct customers. You are challenged to develop a strategy—manual or algorithmic—to consolidate and clean these records.

Appendix B—The Levenshtein Distance

The Levenshtein distance measures the minimum number of single-character edits (insertions, deletions, or substitutions) required to change one string into another.

Example: Comparing "Powerade" and "PowerAde," the Levenshtein distance between the two strings is 1—the uppercase "A" needs to be changed to a lowercase "a" to turn one string into another.

Here's a more complicated example—consider "David" vs. "Daniel." The Levenshtein distance between these two strings is 3; the "v" needs to be replaced by an "n"; the "d" needs to be replaced by an "l"; and an "e" needs to be inserted before the last letter.

The Levenshtein distance does a good job at identifying the distance between strings. Unfortunately, it has some shortcomings in this particular context. For example, strings like "The Coca-Cola Company" and "Coca Cola Inc." clearly refer to the same entity, but they have a large distance from each other.

Supplementary Handout: Blend360's LLM-Based Solution for Client Name Rationalization and Churn Prediction

Introduction

In early 2024, the Blend360 team took part in an internal generative AI hackathon. At the same time, they were working with a packaging and labeling client facing severe data quality issues: Customer names were inconsistent across systems, and brand fields were missing or unstandardized. These inconsistencies made it nearly impossible to identify customers at risk of churn, let alone model it. The hackathon provided a timely opportunity to try something new—could large language models be used to resolve the messy data?

This document outlines the approach they ultimately deployed: how they used LLMs to clean and rationalize the data, how they validated the approach, and how it enabled the development of a scalable churn prediction model.

The Challenge Revisited

To build a reliable churn model, the Blend360 team first needed to unify order data drawn from multiple systems. The problem was not just technical but conceptual: Each system had developed independently, and customer records had never been standardized. The same client might appear as "Coca-Cola Inc." in one dataset, "Coke Puerto Rico" in another, and "CocaCola" elsewhere. Brand fields were missing or inconsistently filled, and product descriptions—often stored as free text—were dense with abbreviations, packaging codes, and internal jargon. Initial attempts to resolve these inconsistencies using traditional tools like string similarity metrics or natural language processing (NLP) libraries fell short. In many cases, there simply wasn't enough structure in the data to extract consistent information.

The team's early attempts—based on string similarity metrics and classical NLP libraries—were insufficient, especially for messy abbreviations and inconsistently formatted brand names.

Using LLMs to Clean the Data

It was around this time that the team began exploring the use of large language models. The Blend360 hackathon provided the perfect opportunity to test a hypothesis: Could LLMs infer structure from disorder and help rationalize the dataset?

They focused on two related problems. First, they needed to infer missing brand names from inconsistent product descriptions. Second, they needed to cluster variant customer names to build a unified customer profile. Initial attempts to use LLMs at the row level—feeding one item description at a time—were disappointing. The model often hallucinated brands or failed to interpret the shorthand in isolation. The breakthrough came when the team changed tactics: Instead of feeding in each name in isolation, they provided the model with grouped context.

For example, they aggregated all item descriptions associated with a single customer into one input. With this additional context, the model was able to identify patterns and return more consistent results.

For customer name resolution, they used a similar strategy, processing names in small, overlapping batches and prompting the model to group them. To preserve data privacy, the entire pipeline was run from within the client's Snowflake environment using Cortex AI, calling Anthropic Claude 3 models without allowing data to leave the platform.

How It Worked

The implementation required some careful experimentation. For brand extraction, the team grouped all item descriptions associated with a given customer and fed them to the LLM as a single, contextualized prompt. This broader context helped the model detect patterns and infer missing brand names more reliably than when examining records one by one.

A similar approach worked for customer name resolution. Instead of feeding the model each name in isolation, they created moving windows of 10 to 20 names and asked the LLM to group those referring to the same entity. Prompt design and chunk size proved critical to minimizing hallucination and ensuring stability across runs.

Validating the Approach

The Blend360 team knew that any AI-generated results needed validation. They conducted several rounds of random sampling—reviewing approximately 100 entries per round—to spot-check the model's outputs. Particular attention was given to high-revenue customers, where accuracy was especially important.

Overall, the results were encouraging. In roughly 96% to 98% of cases, the model provided a valid brand or rationalized name. While the best possible label was returned in about 80% to 90% of cases, the team also found that the aggregated order values aligned well with business expectations after the cleaning process. For ambiguous or high-risk entries, they retained a human-in-the-loop process to manually override the model's suggestions.

Building the Churn Model

With a clean and rationalized dataset in hand, the Blend360 team turned to the core task: building a churn prediction model. Their goal was to anticipate which customers were at risk of reducing or reallocating order volume to competing vendors—before it actually happened.

The team began by engineering features that captured meaningful shifts in customer behavior over time. They created more than 300 variables that reflected patterns like order volume trends, changes in delivery timeliness, frequency of support contacts, and details from recent negotiations. For example, they compared recent three-month order volumes to the prior six-month baseline, calculated the share of late orders, tracked the number of complaints, and

summarized negotiation outcomes and duration. This multidimensional view helped the model detect subtle behavioral changes that could precede a customer's decision to churn.

For modeling, the team experimented with several machine learning algorithms, including Random Forest and XGBoost. In the end, XGBoost proved the most effective, offering strong performance even in the face of noisy, sparse features. One nuance the team had to account for was the heterogeneity of the customer base. Some clients were primary customers who ordered regularly; others were secondary or tertiary clients who only placed occasional orders. This variability meant that a sharp drop in volume didn't always signal churn—sometimes it was just business as usual. The model had to be trained carefully to avoid flagging such customers as false positives.

Impact and Results

The results were compelling. The final model successfully flagged accounts representing nearly $45 million in potential churn. Even if the client managed to retain just 10% of that amount, the model could be credited with preserving $4.5 million in revenue. Beyond the direct financial impact, the project offered substantial operational efficiencies. The LLM-powered data cleaning pipeline replaced what would have otherwise been months of tedious manual work. Because the entire approach was modular and reusable, Blend360 has since deployed variations of the same solution for other clients.

Importantly, the output of the model wasn't just a churn score. It also surfaced the likely drivers of churn for each customer—such as a string of late shipments or unresolved complaints—giving sales teams a clear rationale for outreach and a way to tailor their conversations. Sales teams received actionable alerts with both a churn score and top contributing factors (e.g., repeated late shipments, recent unresolved complaints).

Lessons Learned

Reflecting on the project, the team walked away with several lessons. First, while LLMs were effective at resolving messy entity data, they were not turnkey. Success depended heavily on prompt tuning and thoughtful batching. Larger, context-rich prompts outperformed smaller ones and helped the model generalize more accurately.

Second, human review remained essential. Especially for high-revenue accounts or ambiguous entries, it was important to retain a human-in-the-loop. This added oversight improved trust and ensured data quality.

Third, by deploying the entire solution within the client's secure Snowflake environment, Blend360 avoided many of the usual friction points around data security and model deployment. The client could use modern AI tools without compromising control over sensitive information.

Finally, the hybrid architecture—using generative AI to prepare the data and classical ML to predict churn—proved especially powerful. Rather than rely on a single technology, the team used each for what it did best.