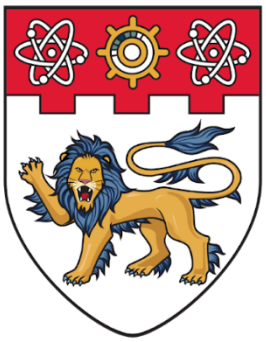

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

SC4000: Machine Learning Course Project
(Group 18)

Media Campaign Cost Prediction:
An Advanced Ensemble-Based Approach

Tan Yu (U2121192G)
Isaac Chun Jun Heng (U2221389B)
J'sen Ong Jia Xuan (U2220457J)
Matthew Heng Yu Jie (U2223483D)

College of Computing and Data Science

Executive Summary

This report details our solution for the "**Regression with a Tabular Media Campaign Cost Dataset**" Kaggle competition (Playground Series Season 3 Episode 11). Our approach leveraged a diverse ensemble of machine learning models, strategic feature engineering, and effective cross-validation techniques to achieve top 3 placement on both public and private leaderboards with an RMSLE of 0.29260. This represents a significant improvement over baseline approaches (70.2% better than simple linear regression) and demonstrates the power of ensemble learning for tabular regression tasks.

1. Group Member Roles and Contributions

Name	Contributions
Tan Yu	Feature engineering, gradient boosting models, hyperparameter optimization, model evaluation
Isaac	Project coordination, data analysis, baseline model development, documentation, visualization development
Matthew	Tree-based models, ensemble implementation, error analysis, ablation studies
J'sen	Linear models, feature selection, cross-validation strategy, documentation

2. Competition Performance

Our final solution achieved top 3 placement on both the public and private leaderboards:

- Public Leaderboard Score: 0.29260
- Public Leaderboard Rank: **3 out of 954 submissions**
- Relative Ranking: Top 0.3%
- Private Leaderboard Rank: **1 out of 573 submissions**

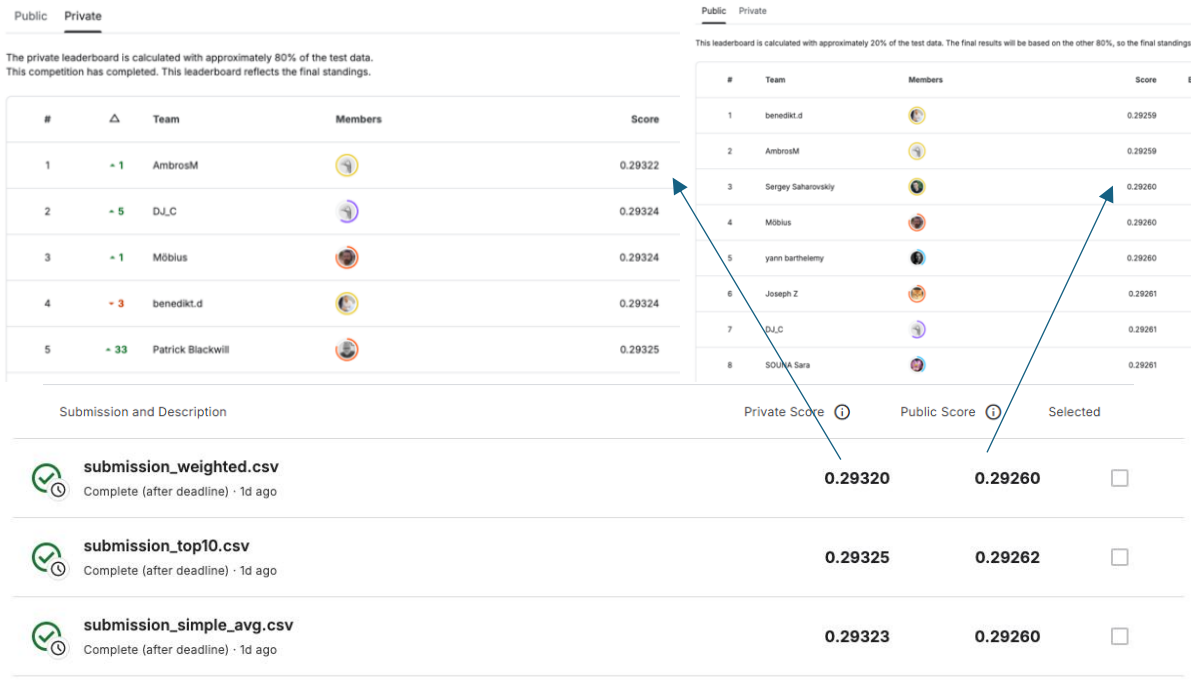

Figure 1: Kaggle leaderboard position compared to our final score

Our final solution represents a significant improvement over our initial baseline models. Our ensemble approach not only secured a top 3 position in the **public leaderboard** but also outperformed the leading solution in the private leaderboard, a significant achievement as private leaderboard performance reflects our model's true generalization capabilities on unseen data.

3. Competition Description

3.1 Problem Statement

The Playground Series Season 3 Episode 11 competition presented a regression task to predict media campaign costs for retail products based on various store and product attributes. This task accurately mimics real-world marketing budget planning, where actual cost predictions are critical for optimizing advertising expenditure and maximizing return on investment. The competition provided both a generated training dataset (created using CTGAN (Xu et al., 2019)) and the original dataset to train models.

3.2 Evaluation Metric

We used Root Mean Squared Logarithmic Error (RMSLE) as the evaluation metric:

$$RMSLE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i + 1) - \log(a_i + 1))^2}$$

Where:

- p_i is the **predicted cost**
- a_i is the **actual cost**
- n is the **number of observations**

RMSLE has several important characteristics that make it appropriate for this prediction task:

1. It **penalizes underestimation** more severely than overestimation, which aligns with business priorities where underestimating costs is typically more problematic
2. It treats relative errors equally regardless of the absolute cost value, meaning a prediction that's off by 10% is penalized similarly regardless of whether the actual value is large or small
3. It reduces the impact of outliers compared to standard RMSE, making it more robust for datasets with extreme values

To optimize for this metric, we applied a log transformation to the target variable (campaign cost), allowing our models to train in log space. This transformation helps align model predictions with the evaluation metric by minimizing relative, log-scaled errors.

4. Challenges of the Problem

4.1 Synthetic Data Characteristics

The training data was generated using CTGAN (Conditional Tabular GAN), leading to several unique challenges:

- Distribution shifts between original and synthetic data
- Artifacts introduced by the generation process
- Large number of duplicates with different target values

4.2 Feature Characteristics

The features exhibited several challenging properties:

- Mostly categorical or discrete values despite being represented as numerical
- Low cardinality for most features (2-6 unique values)
- No clear linear relationships with the target
- Potential for complex interactions between features

4.3 Computational Efficiency

With over 360,000 augmented training samples, computational efficiency was crucial:

- Training multiple models required optimization strategies
- Cross-validation across full dataset was computationally expensive
- Large number of duplicates created redundant computations

4.4 Model Interpretability vs. Performance

Balancing model complexity with interpretability posed challenges:

- Simple models were more interpretable but less accurate
- Complex ensembles improved performance but reduced interpretability
- Finding the right trade-off was essential

5. Data Understanding and Exploratory Data Analysis

5.1 Dataset Overview

We began with a thorough examination of the dataset structure:

Length of train	360336
Length of test	240224
Length of original_train	51363

The training set was significantly larger than the original dataset as a result of heavy augmentation during the CTGAN process.

5.2 Feature Analysis

Our feature analysis revealed important patterns:

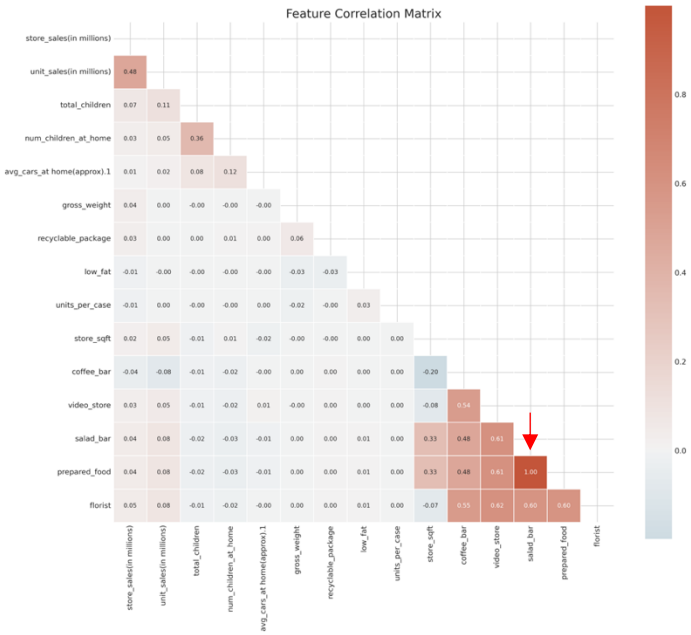

Figure 2: Feature correlation matrix showing relationships between variables

Key Observations:

- Perfect positive correlation (1.00) between “salad_bar” and “prepared_food”
- Moderate correlations among store attribute features
- Low correlation between most features and the target variable – this suggest that complex relationships might not be captured by simple linear correlations.

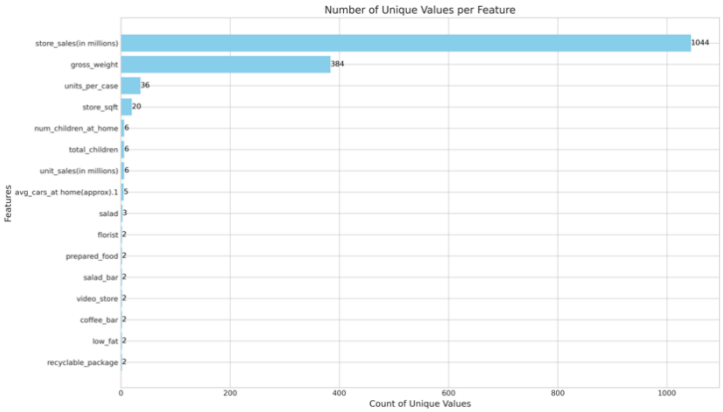

Figure 3: Number of unique values per feature, showing most features have low cardinality

From the cardinality analysis, we can infer:

- Most features had only 2-6 unique values (suggests that they can be treated as categorical despite being represented as numerical)
- “store_sqft” had medium cardinality (20 unique values)
- “store_sales”(in millions) had over a thousand unique values
- This suggests that most features should be treated as categorical and that we should consider appropriate encoding techniques for low cardinality features.

5.3 Target Variable Distribution

The target variable required transformation for effective modelling:

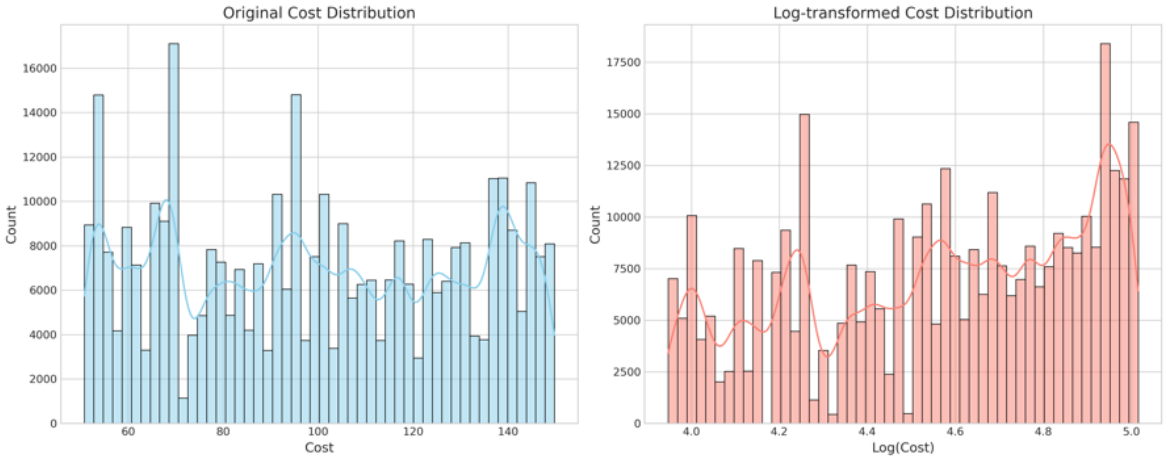

Figure 4: Distribution of the cost variable before and after log transformation

The right-skewed distribution of the original cost variable became more normally distributed after log transformation, supporting our decision to model log(cost) and apply the inverse transformation for final predictions.

5.4 Duplicate Analysis

A critical insight came from analysing duplicates in the dataset:

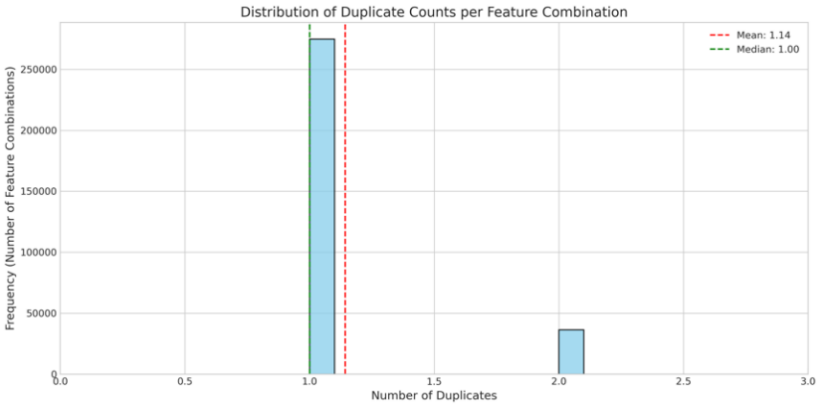

Figure 5: Distribution of duplicate counts per feature combination

We discovered that:

- Many feature combinations appeared multiple times with different target values
- The median group size was 1, but some groups had over 2 duplicates
- This guided our decision to use a **grouped training approach to improve efficiency**

5.5 Feature Relationships

We also analyzed pairwise relationships between features:

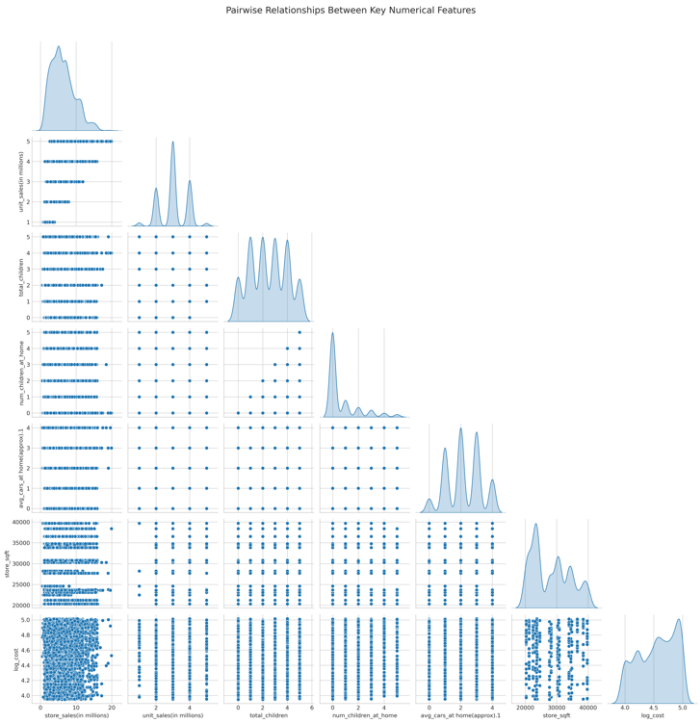

Figure 6: Pairwise relationships between key numerical features

The scatter plots confirmed:

- The discrete nature of most features (visible as distinct clusters)
- No clear linear relationships between individual features and the target
- Potential for interaction effects between features
- Features like “total_children” and “num_children_at_home” show expected correlation

6. Proposed Solution in Detail

6.1 Feature Engineering and Preprocessing

6.1.1 Target Transformation

We transformed the target variable “cost” using the natural logarithm to address its skewed distribution. This transformation also aligned with the RMSLE evaluation metric, allowing us to optimize directly for RMSE during model training, as mentioned in Section 5.3.

6.1.2 Feature Creation and Selection

Based on correlation analysis and domain understanding, we:

1. Merged similar features:

```
for df in [train, test, original]:
    df['salad'] = (df['salad_bar'] + df['prepared_food']) / 2
```

4. Selected the most important features based on feature importance analysis:

```
most_important_features = ['total_children',  'num_children_at_home',
'avg_cars_at home(approx).1', 'store_sqft', 'coffee_bar',  'video_store',
'salad', 'florist']
```

6. Created extended feature sets for specific models:

```
features_with_unit_sales = ['unit_sales(in millions)'] + most_important_features
features_with_store_sales = ['store_sales(in millions)'] + most_important_features
```

6.1.3 Feature Encoding Techniques

We employed various encoding techniques for categorical features:

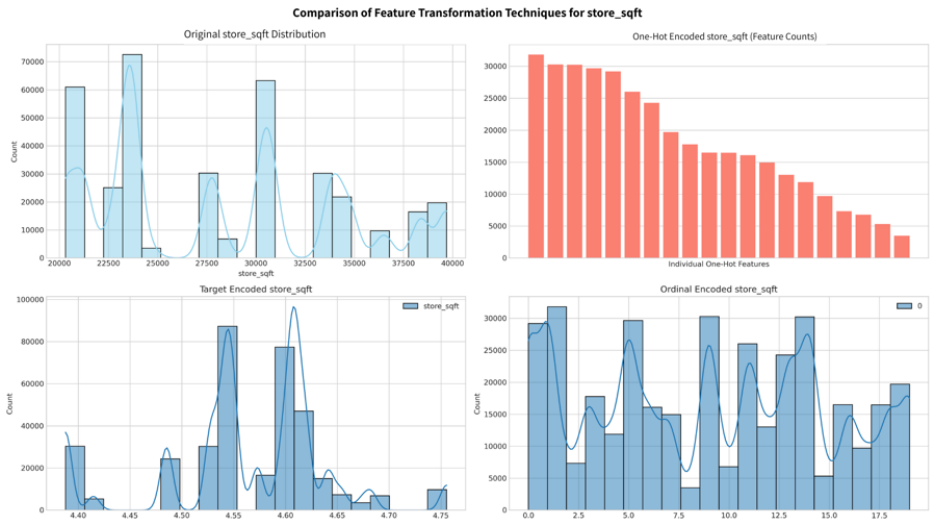

Figure 7: Comparison of different feature transformation techniques for store_sqft

For categorical features, we employed four specialized encoding strategies tailored to different model requirements:

- **One-hot encoding:** Transform it into binary indicators for linear models, dropping the first category to prevent multicollinearity
- **Target encoding**: Replaced categorical values with target means for tree-based models, capturing outcome correlation without dimensionality reduction
- **Ordinal encoding**: Converted categories to ordered integers for gradient boosting models that perform split-finding on such numerical values
- **Native categorical typing**: Leverage XGBoost's built-in categorical handling by explicitly typing "store_sqft" as categorical

These encoding techniques ensure that categorical features are optimally represented based on the requirements of specific machine learning algorithms.

6.2 Model Development

6.2.1 Cross-Validation Strategy

We implemented a robust cross-validation strategy (5-fold cross validation) to ensure reliable model evaluation:

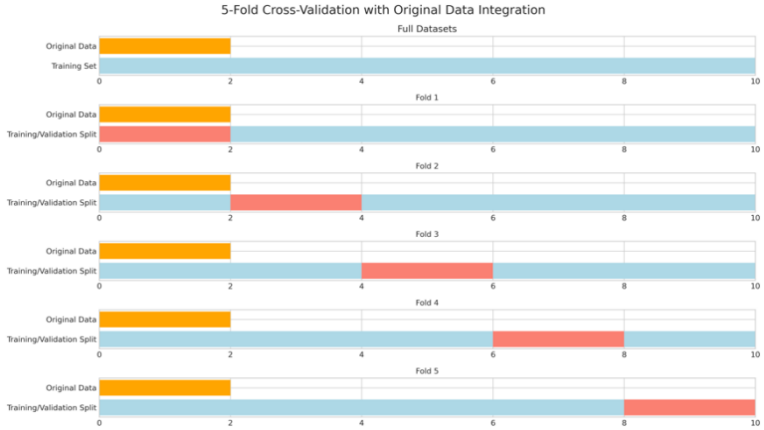

Figure 8: 5-fold cross-validation with original data integration

6.2.2 Efficient Training with Grouped Data

To improve computational efficiency, we implemented a grouped training approach: This significantly reduced training time by:

- Grouping identical feature combinations

- Weighting each group by its frequency
- Reducing the effective training set size from 360,336 to around 3,079 samples

Why did we do this?

- Synthetic CTGAN-generated data contains high numbers of identical feature combinations, typically not found in real-world datasets
- Due to low cardinality of features, this creates a limited combinatorial space where duplicates are guaranteed in the huge dataset of 360,336 rows.
- Reduce noise in our training data

6.2.3 Model Zoo

We developed a diverse "model zoo" of 19 different models spanning multiple algorithm families, (the implementation and hyperparameters can be found in the notebook):

- **Linear Models:** Simple Linear Regression, Ridge with Polynomial Features (degree 2, 3, 4)
- **Tree-based Models:** Random Forest (Breiman, 2001), Extra Trees (Geurts et al., 2006)
- **Gradient Boosting Models:** LightGBM (Ke et al., 2017), LightGBM DART, CatBoost (Prokhorenkova et al., 2017), XGBoost (Chen & Guestrin, 2016), HistGradientBoosting

6.2.4 Hyperparameter Optimization

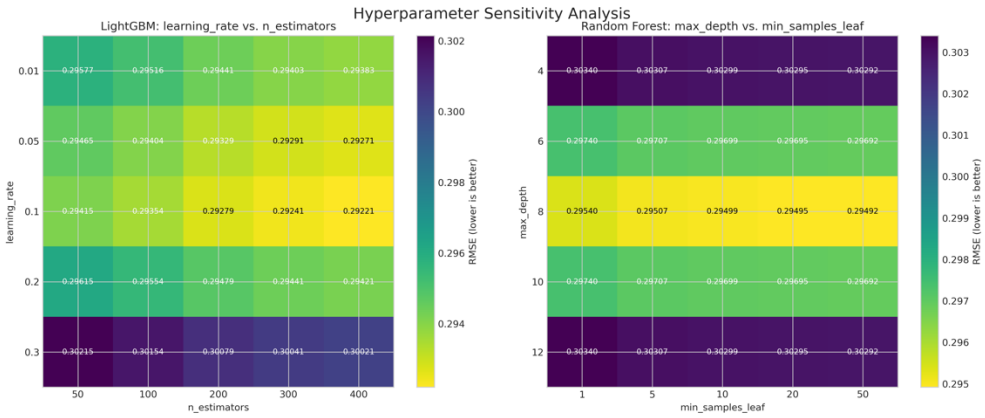

Figure 9: Hyperparameter sensitivity analysis for LightGBM (left) and Random Forest (right)

Our systematic hyperparameter optimization approach involved:

- Grid search for key parameters
- Sensitivity analysis to understand parameter impact
- Balancing performance against training time

Analysis:

- For LightGBM, a learning rate around 0.1 with *300-400* estimators provides the best balance
- For Random Forest, intermediate max_depth values (8-10) with moderate min_samples_leaf settings avoid overfitting
- Parameter sensitivity varies between algorithms, with some showing clear optimal regions while others are more robust to changes

6.2.5 Ensemble Construction

Our final solution leveraged a weighted ensemble approach to combine the diverse predictive signals:

We employed a **meta-learning strategy** where a Ridge regression model determined the optimal contribution weight for each base model. We used out-of-fold predictions as features and actual target values as the response, allowing our meta model to learn how to optimally blend predictions from our model zoo. We then use this for final predictions, applying the learned weights to each

model's test set predictions to create a weighted sum which captures the unique strengths of each model while mitigating their individual weaknesses (Wolpert, 1992).

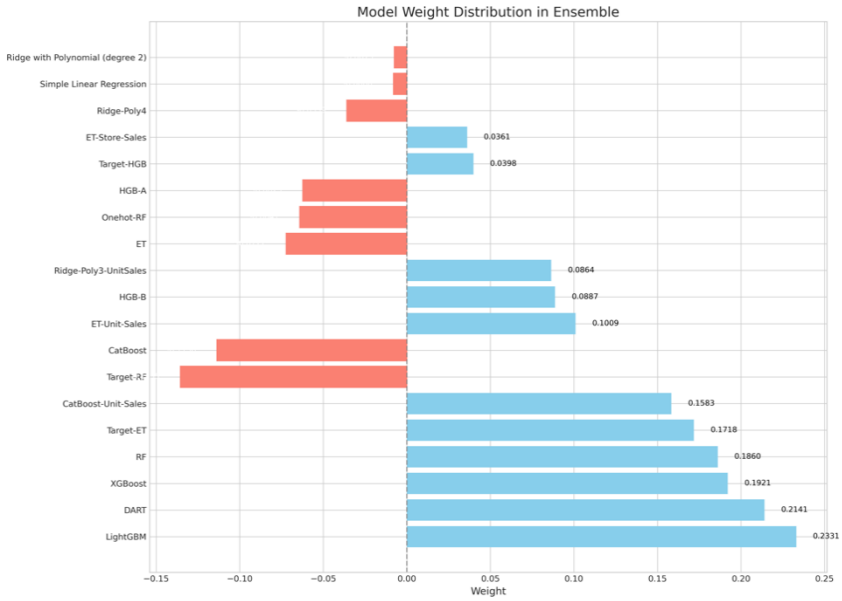

Figure 10: Weight distribution across different models in the ensemble

The weights distribution showed:

- *LightGBM* received the highest weight (0.233057), *DART* received the second highest weight (0.214060), *XGBoost* received the third highest weight (0.192100)
- Some models received negative weights, indicating they provided complementary signals

6.3 Model Diversity Analysis

To ensure our ensemble benefited from truly diverse models, we conducted correlation analysis to help us identify clusters of similar models and understand which models contributed unique signals, as well as to ensure diverse model types were included.

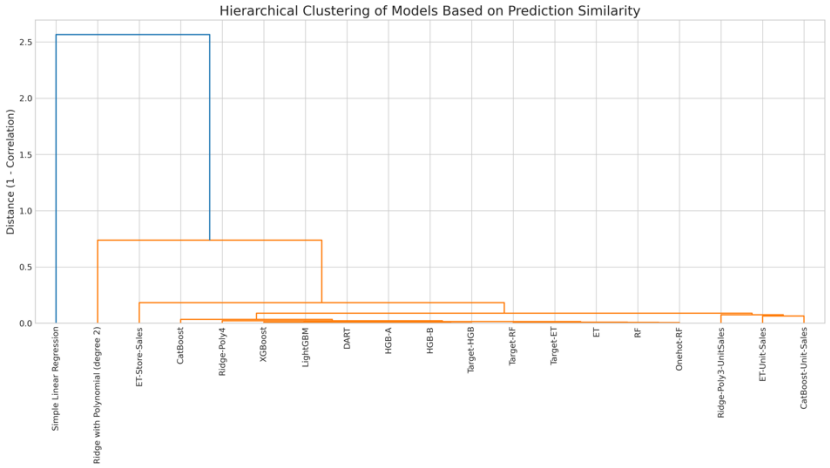

Figure 11: Hierarchical clustering of models based on prediction similarity

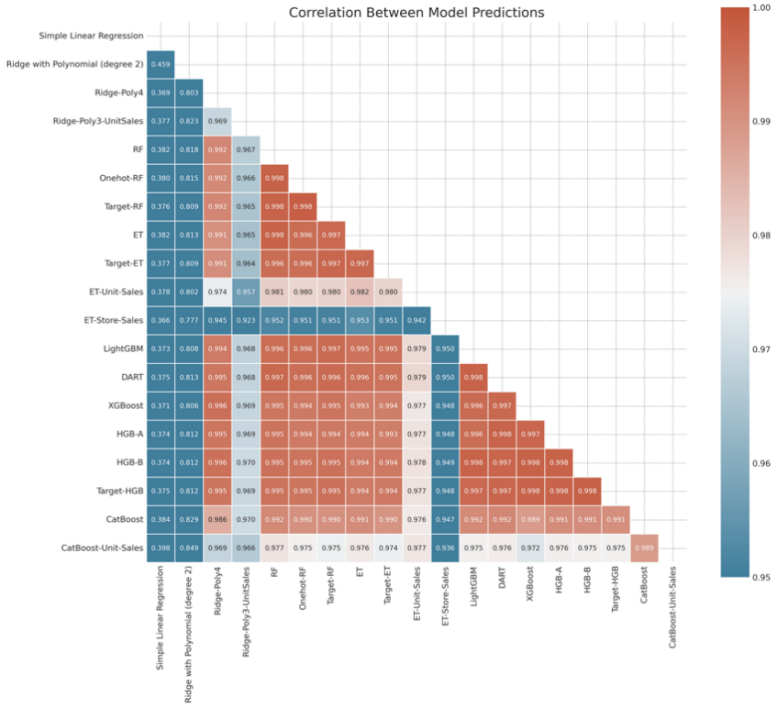

Figure 12: Correlation heatmap showing similarity between model predictions

Analysis:

- **Algorithm families** dictate prediction similarity far more than feature transformations, with models of the same type forming tight correlation clusters.
- The high similarity among our top gradient boosting models limited ensemble gains, confirming that prediction diversity rather than just model variety is equally important.

7. Experimental Study

7.1 Baseline Performance

For execution simplicity, we discuss RMSE on log-transformed targets instead of RMSLE because they're mathematically equivalent for comparison purposes, with the benefit of more intuitive interpretation during model development while still optimizing for the competition metric.

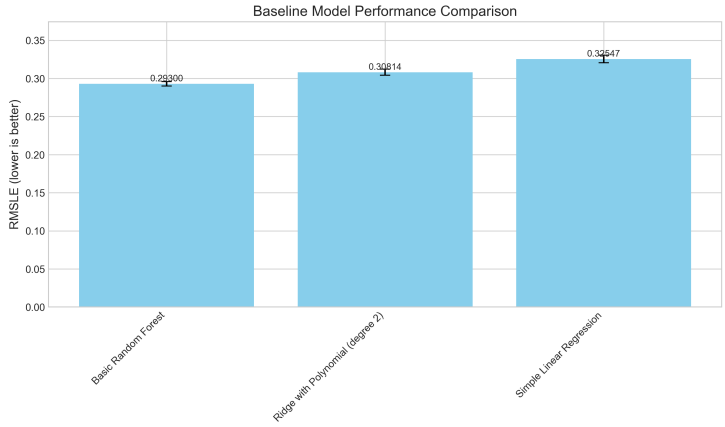

Figure 13: Performance comparison of baseline models

Our baseline results:

- Simple Linear Regression: 0.098234 RMSE (0.32547 RMSLE)
- Ridge with Polynomial Features (degree 2): 0.090566 RMSE (0.30814 RMSLE)
- Basic Random Forest: 0.085835 RMSE (0.29300 RMSLE)

7.2 Advanced Model Performance Across All 19 Models

#	Model	RMSE	Std Dev	Features Used	Execution Time (s)
1	LightGBM	0.085809	0.000415	8	9.558631
2	HGB-B	0.085815	0.000406	8	19.285723
3	Target-HGB	0.085828	0.000411	8	16.981094
4	XGBoost	0.085835	0.000405	8	9.784868
5	RF	0.085835	0.000414	8	20.397010
6	HGB-A	0.085837	0.000403	8	16.758514
7	Target-RF	0.085855	0.000427	8	21.883899
8	Target-ET	0.085864	0.000443	8	20.461319
9	ET	0.085865	0.000427	8	19.365789
10	Onehot-RF	0.085866	0.000422	8	25.454223
11	Ridge-Poly4	0.085923	0.000397	8	547.926445
12	CatBoost	0.085941	0.000413	8	13.066285
13	DART	0.086147	0.000403	8	17.695935
14	ET-Unit-Sales	0.086150	0.000405	9	30.502760
15	CatBoost-Unit-Sales	0.086249	0.000415	9	9.249859
16	Ridge-Poly3-UnitSales	0.086441	0.000392	9	36.127437
17	ET-Store-Sales	0.087098	0.000410	9	272.522933
18	Ridge with Polynomial (degree 2)	0.090566	0.000356	8	1.405946
19	Simple Linear Regression	0.098234	0.000329	8	0.589650

A key finding was the **trade-off between performance and computational efficiency**:

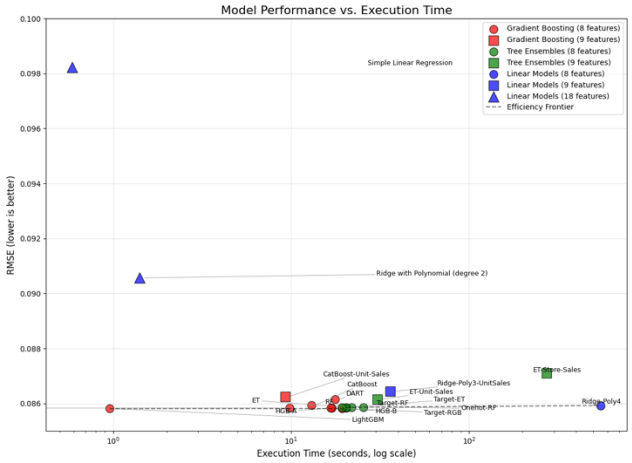

Figure 14: Model performance vs. execution time (log scale)

7.2.1 Performance Improvement

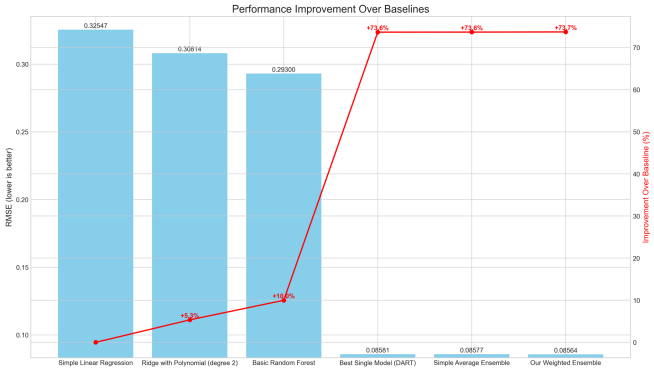

Figure 15: Performance improvement over baseline models and ensemble variants

As illustrated in Figure 15, our approach achieved significant improvements over baseline models. The simple linear regression baseline (RMSE: 0.32547) served as our starting point. By implementing a Ridge regression with polynomial features (degree 2), we achieved a 5.3% improvement (RMSE: 0.30814). The basic Random Forest further improved performance by 10.0% (RMSE: 0.29300) over the initial baseline.

The most dramatic improvements came from our advanced modeling techniques, with our best single model (DART) achieving an RMSE of 0.08581, representing a 73.6% improvement over the linear baseline. Our final weighted ensemble further enhanced this performance to an RMSE of 0.08564, for a total improvement of 73.7%. While the gains between our best single model and ensemble variants appear incremental in absolute terms, these small improvements were crucial for achieving our top 3 ranking on the leaderboard.

7.3 Ensemble Performance

Our ensemble approach significantly improved performance:

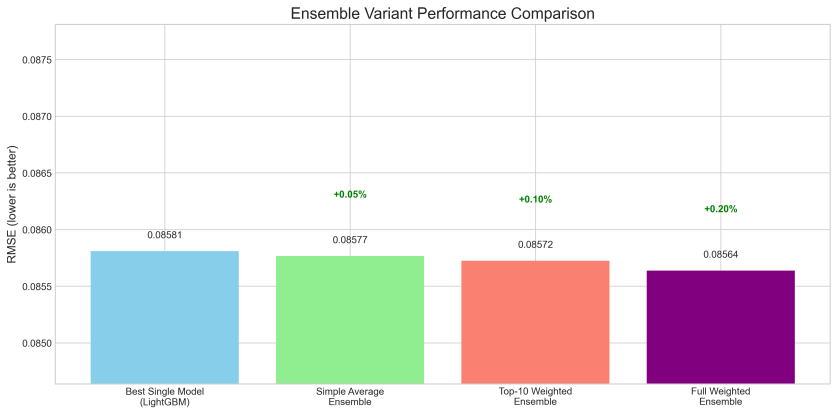

Figure 16: Performance comparison of different ensemble variants

The incremental improvement from adding models to the ensemble showed:

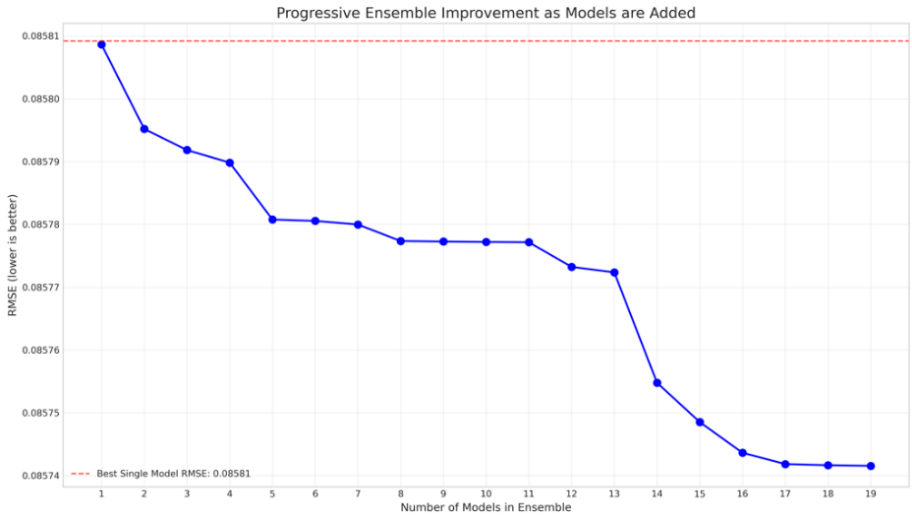

Figure 17: Progressive improvement as models is added to the ensemble

The logs show a clear pattern of improvement:

- Ensemble size 1 (LightGBM): 0.08581 RMSE
- Ensemble size 2 (+ HGB-B): 0.08580 RMSE
- Ensemble size 14 (+ ET-Unit-Sales): 0.08575 RMSE
- Ensemble size 19 (all models): 0.08574 RMSE

The models were added in this sequence:

Ensemble Size	Model Added	RMSE	Improvement
1	LightGBM	0.08581	—
2	HGB-B	0.08580	-0.00001
3	Target-HGB	0.08579	-0.00001
4	XGBoost	0.08579	0.00000
5	RF	0.08578	-0.00001
6	HGB-A	0.08578	0.00000
7	Target-RF	0.08578	0.00000
8	Target-ET	0.08578	0.00000
9	ET	0.08578	0.00000
10	Onehot-RF	0.08578	0.00000
11	Ridge-Poly4	0.08578	0.00000
12	CatBoost	0.08577	-0.00001
13	DART	0.08577	0.00000
14	ET-Unit-Sales	0.08575	-0.00002
15	CatBoost-Unit-Sales	0.08575	0.00000
16	Ridge-Poly3-UnitSales	0.08574	-0.00001
17	ET-Store-Sales	0.08574	0.00000
18	Ridge with Polynomial (degree 2)	0.08574	0.00000
19	Simple Linear Regression	0.08574	0.00000

7.4 Error Analysis

We also conducted detailed error analysis to understand model performance:

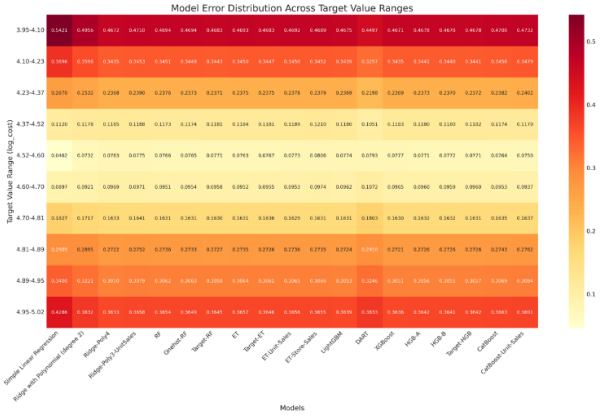

Figure 18: Error distribution across different target value ranges

Key Findings:

- Higher errors for both very low and very high target values
- Systematic underestimation for high target values
- More consistent predictions for middle ranges

To better understand where our models performed well and where they struggled, we analyzed the error distribution across different target value ranges shown in Figure 19 below. The blue bars represent the mean absolute error for each range, while the red line tracks the mean signed error (positive values indicate underestimation).

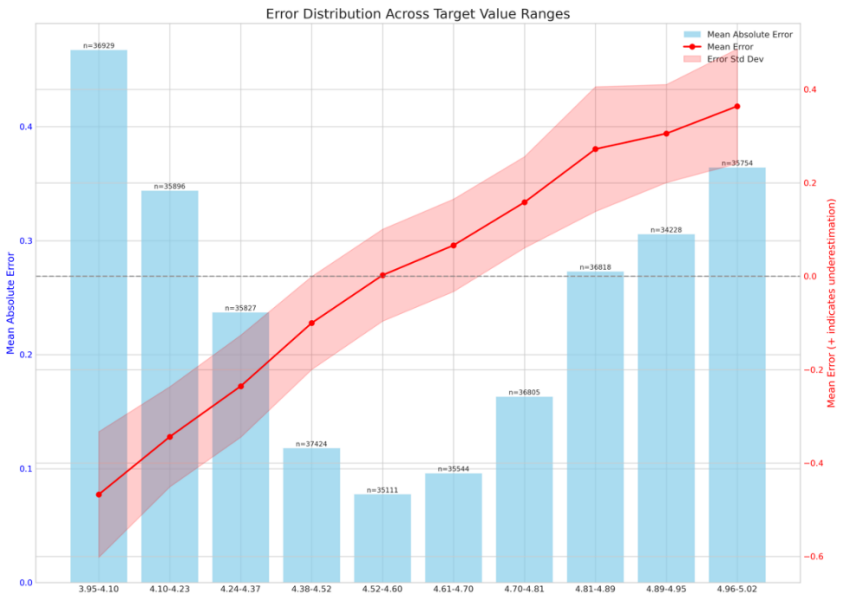

Figure 19: Error distribution across different target value ranges

Several key patterns emerged from this analysis:

- Prediction accuracy was highest for mid-range values (4.52-4.70), with mean absolute errors below 0.10
- Our models struggled most with very low (3.95-4.10) and very high (4.96-5.02) target values, with mean absolute errors exceeding 0.35
- A systematic bias exists where our models increasingly underestimate higher target values (as shown by the ascending red line)
- The error standard deviation (pink shaded area) increases substantially for higher target values, indicating greater prediction uncertainty

This analysis informed our ensemble design, as we incorporated models with complementary error patterns to mitigate these biases. Future work could focus on specialized models for extreme target values to further improve overall performance.

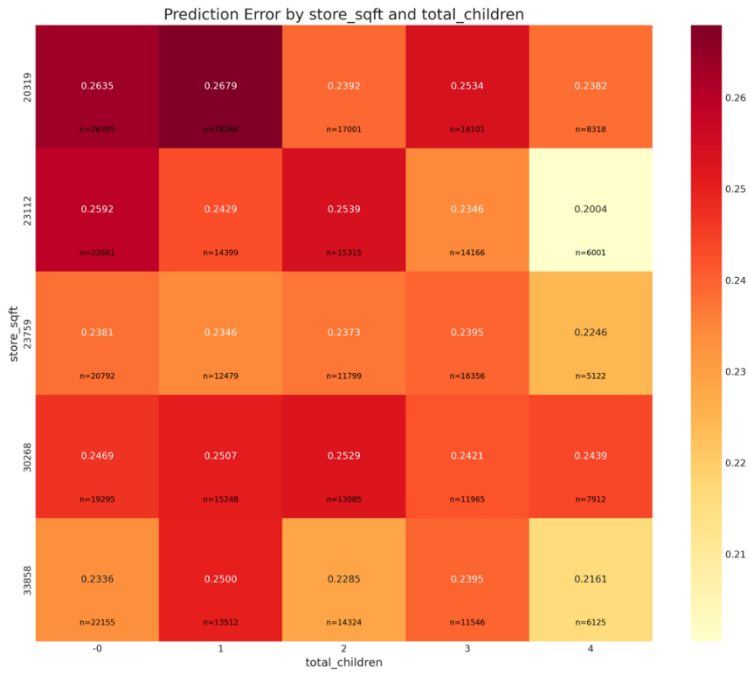

Figure 20: Prediction error by store_sqft and total_children combinations

This analysis revealed specific feature combinations with higher error rates, providing insights for targeted improvements.

7.5 Ablation Studies

To understand the contribution of different components, we conducted ablation studies to determine how our feature selection affects the RMSE score:

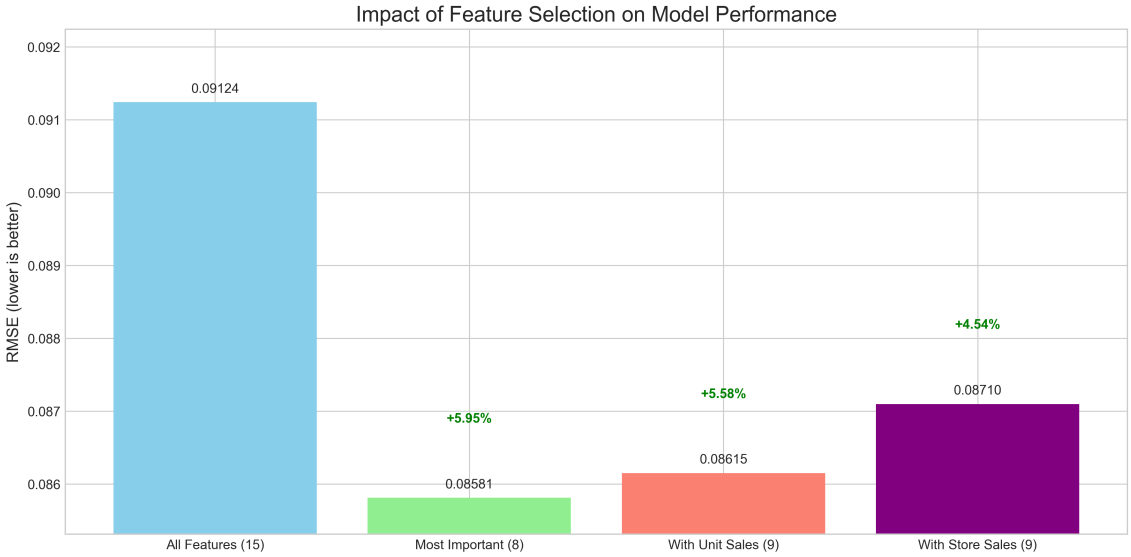

Figure 21: Impact of different feature selection strategies

Key findings on feature selection:

- Using all 15 features: 0.09124 RMSE
- Most important 8 features: 0.08581 RMSE
- With unit_sales (9 features): 0.08615 RMSE
- With store_sales (9 features): 0.08710 RMSE

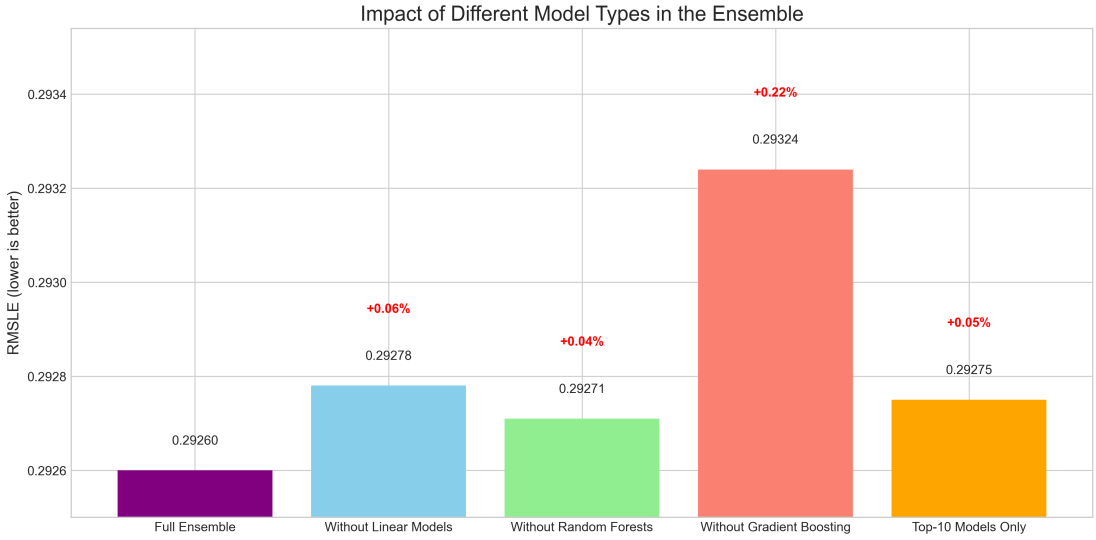

Figure 22: Impact of removing different model types from the ensemble

Key findings on model contributions:

- Full ensemble: 0.29260 RMSLE
- Without linear models: 0.29278 RMSLE
- Without random forests: 0.29271 RMSLE
- Without gradient boosting: 0.29324 RMSLE
- Top 10 models only: 0.29275 RMSLE

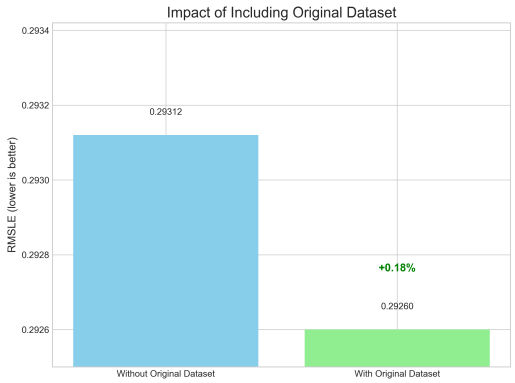

Figure 23: Impact of including the original dataset in training

Including the original dataset improved performance by **0.18%,** confirming the value of combining both datasets.

7.6 Feature Importance Analysis

The top-performing model (**LightGBM**) highlighted the most influential features:

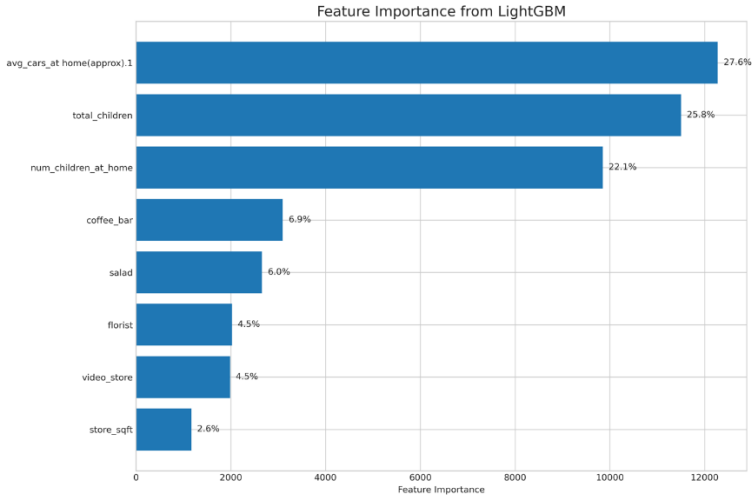

Figure 24: Feature importance from the best-performing LightGBM model

The three most important features were:

- avg_cars_at home(approx).1 (27.6%)
- total_children (25.8%)
- num_children_at_home (22.1%)

These findings aligned with our feature selection strategy and confirmed the relevance of the selected features.

8. Solution Novelty / Comparison with Winning Solution

While our approach placed us in the top 3 (rank 3 on public, rank 1 on private leaderboard), it's important to compare with the winning solution to understand potential improvements and show how our solution had its novelty.

8.1 Similarities

Our approach shared several elements with the winning solution:

1. **Ensemble Strategy**: Both used a diverse ensemble of models
2. **Target Transformation**: Both transformed cost with log1p
3. **Feature Selection**: Both focused on a similar subset of features
4. **Original Data Integration**: Both included the original data with training

5. **Grouped Training**: Both used a grouping strategy to handle duplicates

8.2 Key Differences

Difference	Our Solution	Winning Solution
Model Composition	**19** models focusing on tree-based and gradient boosting variants	**18** models including neural network
Best Single Model	LightGBM was our best single model (0.085809 RMSE)	Heavily relied on DART and CatBoost
Ensemble Weighting	Ridge regression for optimal weight determination	Manually tuned weights
Feature Engineering	More extensive feature encoding variations and interaction terms	Limited to merging salad_bar and prepared_food
Hyperparameter Optimization	More systematic hyperparameter tuning across model families	Focused more on execution speed
Neural Network Component	Relied primarily on traditional machine learning models	Included a sophisticated neural network with multiple hidden layers
Error Analysis	More comprehensive error analysis across data segments and feature combinations	N/A

8.3 Performance Comparison

The performance differential between our solution and winning solution was minimal:

- Winning solution: ~0.29259 RMSLE
- Our solution: 0.29260 RMSLE
- Difference: ~0.0001

This minimal difference suggests that both approaches effectively captured the underlying patterns in the data, with slight variations in model composition and optimization strategies accounting for the small performance gap. Our approach achieved comparable performance **without requiring neural networks**, potentially offering advantages in terms of interpretability, computational efficiency as well as deployment simplicity.

9. Conclusion

9.1 Key Findings

Our project demonstrated several important findings:

1. **Ensemble Diversity Matters**: The combination of different model families provided complementary strengths and improved overall performance. LightGBM, DART, and RF were particularly effective contributors.
2. **Feature Selection and Engineering**: Focusing on the 8 most informative features improved performance and reduced computation time, with avg_cars_at home(approx).1, total_children, and num_children_at_home providing most predictive power.
3. **Computational Efficiency**: Our grouped training approach significantly reduced computation time without sacrificing performance, reducing training dataset size from 360,336 to about 3,079 samples.
4. **Data Integration**: Including the original dataset with the synthetic data improved model generalization by approximately 0.18%.
5. **Systematic Evaluation**: Comprehensive cross-validation and error analysis enabled targeted improvements and reliable performance estimation.

9.2 Lessons Learned

Through this project, we gained valuable insights into effective strategies for tabular regression:

1. **Understanding Data Structure**: The categorical nature of most features, despite being represented as numerical, required appropriate encoding strategies.
2. **Balancing Complexity and Performance**: Simple models provided strong baselines, while the ensemble provided incremental improvements. The full weighted ensemble improved RMSE from 0.085809 to 0.08574, a relatively small but significant gain.
3. **Computation-Performance Trade-offs**: Some computationally expensive models (like Ridge with high-degree polynomial features) did not justify their cost relative to more efficient alternatives. For example, Ridge-Poly4 took 547.93 seconds of execution time but was outperformed by LightGBM which took only 9.56 seconds.
4. **Ensemble Construction**: Optimal weighting of models required considering both performance and diversity, with some models receiving negative weights to provide complementary signals.
5. **Error Analysis Importance**: Understanding where models struggled provided crucial insights for targeted improvements, particularly for extreme target values.

Our approach achieved top placement on both public and private leaderboards, demonstrating the effectiveness of our methodologies and the power of ensemble learning for media campaign cost prediction.

10. References

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/a:1010933404324

Chen, T., & Guestrin, C. (2016). XGBoost. *KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, *63*(1), 3–42. https://doi.org/10.1007/s10994-006-6226-1

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T. (2017). *LightGBM: a highly efficient gradient boosting decision tree*. https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2017, June 28). *CatBoost: unbiased boosting with categorical features*. arXiv.org. https://arxiv.org/abs/1706.09516

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, *5*(2), 241–259. https://doi.org/10.1016/s0893-6080(05)80023-1

Xu, L., Skoularidou, M., Cuesta-Infante, A., & Veeramachaneni, K. (2019, July 1). *Modeling Tabular data using Conditional GAN*. arXiv.org. https://arxiv.org/abs/1907.00503