

SC4000 Project Report

Regression with a Tabular Media Campaign Cost Dataset

Group Number 26

HASAN ADIL

SAHANA

BAKLOUTI ALI

Table of contents

Table of Contents.....	2
1. Background: Choice of Competition.....	3
2. Exploratory Data Analysis.....	3
2.1 What additional data is available to us?.....	3
2.2 Understanding our input and target variables.....	3
2.3 How are our numerical variables distributed?.....	4
2.4 Are our input variables conditionally independent?.....	5
2.5 Summary of Recommendations Resulting from EDA.....	6
3. Feature Selection.....	7
4. Deduplication of repeated rows.....	8
5. Model selection.....	9
5.1 Random Forest vs LightGBM, chosen hyperparameter values.....	9
5.2 Feature Set A vs Feature Set B vs no feature selection.....	10
5.2 Summary of model selection results.....	10
6. Artificial features.....	11
7. Automatic selection of hyperparameters.....	12
8. Test-set ablation of the best model.....	13
9. Final score.....	14
10. Conclusion.....	15
References.....	16

1. Background: Choice of Competition

The course project requires us to develop a solution to one of the several candidate Kaggle competitions given to us. Thus, our first task was to select a competition from the provided list. In order to do so, we considered several factors.

Firstly, the selected competition should not require deep learning (an artificial neural network) to obtain high accuracy. Deep learning carries with it the following issues: the difficulty of reproducing results with even minor alterations to the learning system, the lack of interpretability (Alzubaidi et al., 2021), and a long turn-around time on solution development, as trialling minor tweaks requires hours of training on a GPU before it is possible to reevaluate the accuracy of the system. Thus, we ruled out any computer vision competitions.

Secondly, we decided that competitions featuring time series data (i.e. AMEX Default Prediction, Elo Merchant Category Prediction and the Google Smartphone Decimeter Challenge), as well as competitions benefiting from specialised scientific knowledge (such as Nomad2018), would not be considered due to a lack of relevant experience among team members. Consequently, only the Regression with a Tabular Media Campaign Cost Dataset task was left eligible for us to attempt.

2. Exploratory Data Analysis

2.1 What additional data is available to us?

The first step of any statistical or machine-learning process is the informal examination of the dataset from which we seek to make predictions. With our specific project, a detail which immediately jumped out was this quote from the Kaggle overview: “the dataset for this competition (both train and test) was generated from a deep learning model trained on the [...] [real] dataset.” We have a synthetic dataset of ~360,000 rows generated from a publicly available “original” dataset of ~50,000 rows. This original dataset was not part of the competition but can be [easily obtained from this link](#) (Dutta, 2023) and used in conjunction with the synthetic data for model training.

2.2 Understanding our input and target variables:

What does our synthetic dataset look like? We have 360,336 rows of data and 15 input features; at first glance, **recyclable_package**, **low_fat**, **coffee_bar**, **video_store**, **salad_bar**, **prepared_food**, and **florist** take binary one-or-zero values, whereas **store_sales**, **unit_sales**, **total_children**, **num_children_at_home**, **avg_cars_at_home**, **gross_weight**, **units_per_case**, and **store_sqft** are numerical (as opposed to categorical) data. Furthermore, we have a numerical target variable: **cost**. However, a closer look reveals that, except **store_sales**, **gross_weight** and **cost**, our features take on only a handful of unique values (Table 1.)

Feature name	# unique values in synthetic dataset.	Feature name	# unique values in synthetic dataset.
store_sales	1044	num_children_at_home	6
cost	328	unit_sales	6
gross_weight	384	total_children	6
units_per_case	36	avg_cars_at_home	5
store_sqft	20		

Table 1: Number of unique values per feature in synthetic dataset

Furthermore, with the exception of **store_sales**, **gross_weight** and **cost**, our synthetic datasets’ unique values for these features are identical (with no additional exclusions or inclusions) to those of our 51,363-row non-synthetic dataset and our 240,224-row test set (on our predictions for which we will be evaluated by Kaggle).

Even for **store_sales**, **gross_weight**, and our output variable, **cost**, our number of unique values ranges in the hundreds rather than the hundred-thousands, and 99.9% of the values taken are shared with the non-synthetic dataset and (for input variables) the test dataset.

Thus, only **store_sales**, **gross_weight** and **cost** can be reasonably considered as numerical data; all others should be considered as categorical data by whichever machine-learning model is ultimately used. In fact, our non-binary features are ordinal, as there is a natural notion of “order” to i.e. the number of children or the number of cars we have; in practice, however, common machine learning algorithms implemented in libraries may not have a notion of an ordinal variable as distinct from a categorical variable.

Although not shown here for brevity, with regard to uniqueness of values, our original dataset is similar to the synthetic dataset and produces a similar verdict.

2.3 How are our numerical variables distributed?

Some machine learning models assume that numerical variables are distributed according to a normal distribution; a notable example is linear regression. Thus, prior to training models, we should check whether **store_sales**, **gross_weight** and **cost** are distributed according to a Gaussian. We can do this by simply plotting histograms of each of these features. We use the synthetic dataset for this diagram; the Kaggle overview notes that the synthetic dataset features are distributed similarly to their counterparts on the non-synthetic dataset.

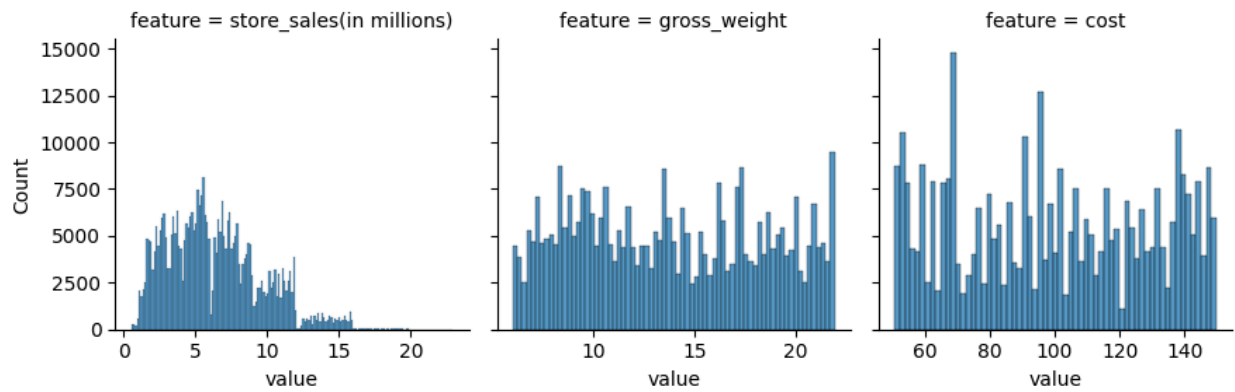


Figure 1: Histograms of numeric features (automatic binning)

Judging from Figure 1, it appears that out of our numerical variables, only **store_sales** follows a Gaussian distribution.

2.4 Are our input variables conditionally independent?

Some models expect independent input variables. One way to easily check the independence hypothesis is to plot a heatmap showing the absolute value of Pearson's correlation between our features; this metric being high implies non-independence. For our categorical data, such as **total_children**, it does still make sense to take Pearson's correlation as a clear ordering exists and the mean rank may be calculated.

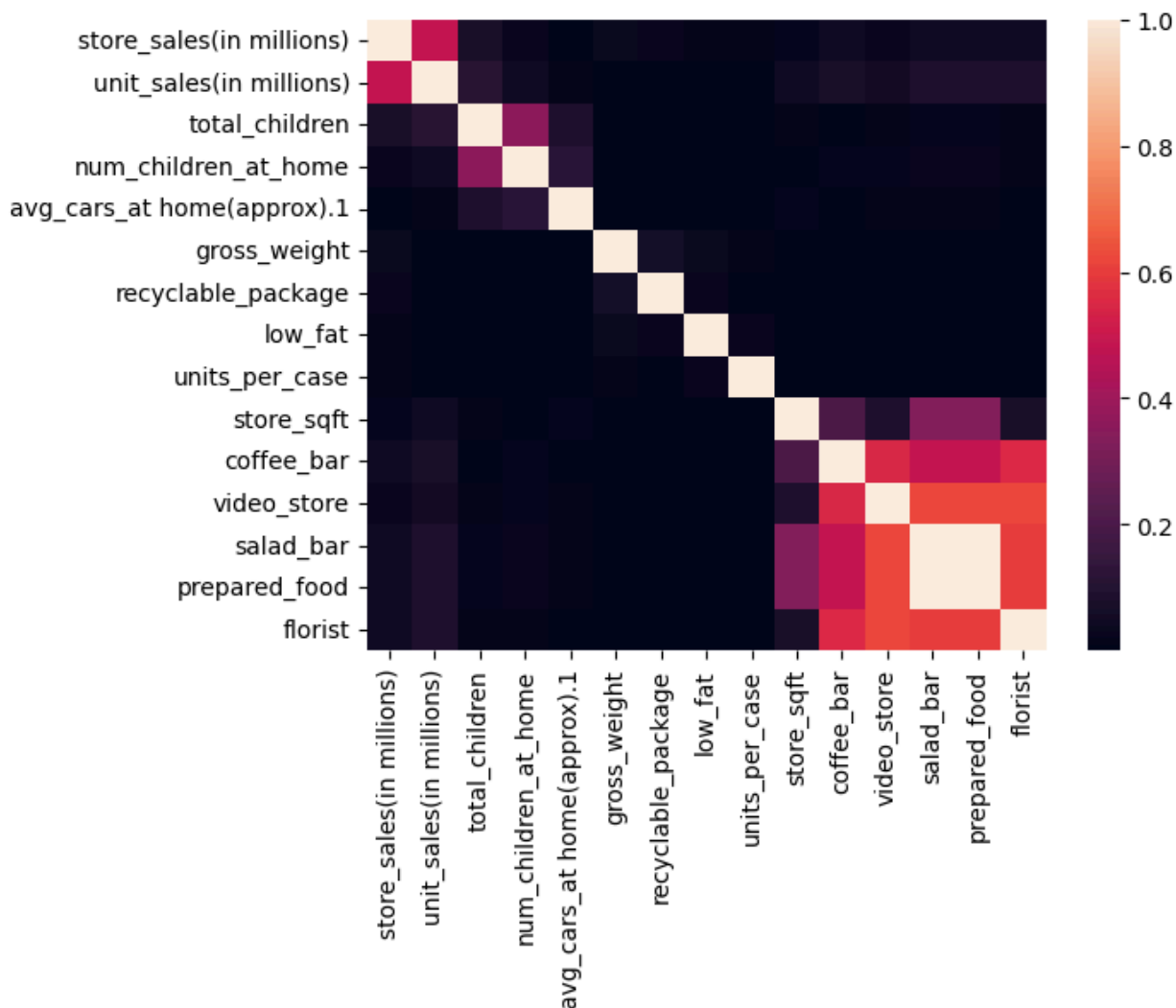


Figure 2: Heatmap of Pearson's correlation between our features

Figure 2 shows us that many input variables correlate linearly with each other. First off, the two binary variables **salad_bar** and **prepared_food** have almost perfect correlation, due to which we should merge them in preprocessing prior to running our model. Besides these, we can find many pairs with non-perfect but still high correlation, such as **total_children** and **num_children_at_home** (which makes intuitive sense, as one would expect having many children in total to often cause having many children in one's house.)

2.5 Summary of Recommendations Resulting from EDA

Due to the availability of the original dataset from which our synthetic data was generated, it may be beneficial to train on the concatenation of the original and synthetic datasets. Features besides **store_sales**, **gross_weight** and **cost** (our target) should be considered to be categorical, rather than numerical, and encoded as such prior to being provided to a

machine-learning model. Our numerical input features, **store_sales** and **gross_weight**, are not normally distributed and should not be used as-is with a model which assumes a normal distribution. Lastly, there is significant Pearson’s correlation between input features; highly redundant pairs (especially **salad_bar** and **prepared_food**) should be merged and additional pruning and feature selection is likely warranted.

3. Feature Selection

In the naive case, we may make use of all available input features for our final model. This implicitly assumes that all available input features are *useful* and *relevant* for making predictions; if instead some of them are mostly “random noise” which has little bearing on the target variable, then factoring in all of the variables into our model can lead to it “overfitting” by picking up statistical patterns where they are none (Venkatesh & Anuradha, 2019).

How may we formalise the notion of the “relevance” of an input feature to the target variable? One intuitive notion is that of the *mutual information* of two random variables. Mutual information quantifies how much knowing the value of one variable lets us “narrow down” or reduce uncertainty in the values the other variable can take. The mutual information of two variables X and Y can be thought of as reaching its minimum when they are completely independent of each other, so that $P(X=x|Y=y) = P(X=x) \forall y$, and reaching its maximum when one variable is entirely defined as a function of the other so that $P(X=f(y)|Y=y) = 1 \forall y$. Filtering input variables based on their calculated mutual information with the target variable turns out to be a common technique in the literature (Xiang Sean Zhou et al., 2003).

Prior to calculating our mutual information scores and in accordance with the EDA, we merged the nearly perfectly correlated **salad_bar** and **prepared_food** into the combined feature **salad_bar_and_prepared_food**, by simply adding the boolean one-or-zero values of the two features (thereby achieving an almost lossless transformation.) Figure 3 shows the resulting mutual information scores of each input feature with the target feature (**cost**), on the basis of which we selected **unit_sales**, **total_children**, **num_children_at_home**, **avg_cars_at_home**, **store_sqft**, **coffee_bar**, **video_store**, **florist**, and **salad_bar_and_prepared_food** as our nine features for further consideration. Out of these, **unit_sales** was considered a possible additional candidate for removal due to its high Pearson’s correlation with **store_sales**, which does not have a high mutual information score with the target variable. The result is two candidate feature sets of respectively nine and eight features (with and without **unit_sales**), which we term *Feature Set A* and *Feature Set B*.

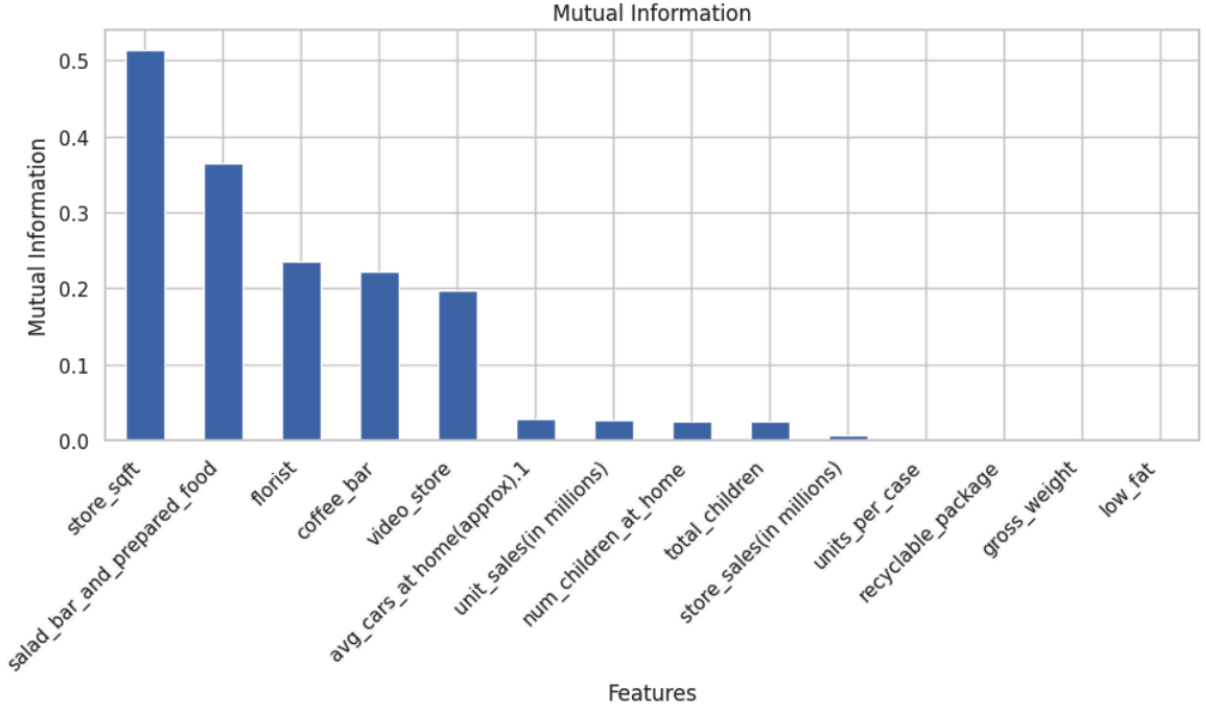


Figure 3: Bar plot of the mutual information scores between input features.

4. Deduplication of repeated rows

With both the nine-feature *Feature Set A* and the eighth-feature *Feature Set B*, we observed that many rows of the input datasets were exact duplicates with regard to the selected input features. *Feature Set A* produced only 8166 unique rows with regards to the selected input features out of our approximately 360,000-row synthetic dataset, or **2.26% unique rows**. *Feature set B* produced only 3075 unique rows or **0.854% unique rows**. Similar figures were produced for the original dataset.

Consequently, in order to accelerate model training, the decision was made to make use of the “sample weight” functionality provided by many machine learning models to deduplicate our training data; after selecting the input features and prior to training, all rows which formed an equivalence group with regards to the input features were merged into a single row, assigned the average of the target variable of the aggregated rows as its value for **cost** and the number of aggregated rows as the **sample weight**, so that machine learning algorithms would consider the data point with equivalent importance as to before aggregation. This proved especially helpful for conducting cross-validation, where many successive training fits must be computed.

5. Model selection

5.1 Random Forest vs LightGBM, chosen hyperparameter values

Considering the highly categorical nature of our remaining input features, of which **gross_weight** is the only numeric feature in either *Feature Set A* or *Feature Set B*, it was decided to make use of tree-based ensembles, as they are a) well-known to be suitable for direct usage with mixed numerical and categorical variables without much processing or encoding work, b) have wide support in software libraries for machine learning, and c) are covered in our course. For this, a decision had to be made between a standard Random Forest, and an alternative boosting-based model, LightGBM, which has proven popular in machine learning practice. To this end, we performed a 5-fold cross-validation of both LightGBM and Random Forest with our eight-input feature *Feature Set B*, with training data prepared as the concatenation of the synthetic and original datasets followed by deduplication. The choice of using both the synthetic dataset and the original dataset for training is validated in Section 8 of this report. Manual hyperparameter tuning was performed; for both models, `n_estimators=450` were used, with a learning rate of 0.1, `num_leaves=100`, `min_child_samples=1` and `min_child_weight=10` for our LightGBM. Our model is trained to minimise the RMSLE by way of providing the logarithm of the true target variable (**cost**) as the target provided to the machine-learning algorithm. The average cross-validated RMSLE across all five splits is reported in Table 2. Based on these scores, it was decided to go with the LightGBM as our model for the rest of this project.

Model used	Feature Set	Cross-validated RMSLE
Random Forest (manual tuning of hyperparameters)	B	0.293777
LightGBM (manual tuning of hyperparameters)	B	0.292931

Table 2: Comparison between Random Forest and LightGBM

5.2 Feature Set A vs Feature Set B vs no feature selection

Recall that the eight-input-feature Feature Set B is distinguished from the nine-input-feature Feature Set B via the exclusion of the **unit_sales** feature. Which one of these input feature sets performs better? We can see this by simply cross-validating both with the LightGBM, using the same procedure and hyperparameters as before. We also tested the base case of no feature selection (all provided features used).

Model used	Feature Set	Cross-validated RMSLE
LightGBM (manual tuning of hyperparameters)	No feature selection	0.296076
LightGBM (manual tuning of hyperparameters)	A	0.293383
LightGBM (manual tuning of hyperparameters)	B (no unit_sales)	0.292931

Table 3: Comparison between Feature Set A and Feature Set B

As shown in Table 3, Feature Set B performs best and no feature selection at all performs especially badly (showing the usefulness of our approach.)

5.2 Summary of model selection results

The best-performing model so far is LightGBM with Feature Set A (no unit sales), and manually-tuned hyperparameter values of `n_estimators=450`, a learning rate of 0.1, `num_leaves=100`, `min_child_samples=1` and `min_child_weight=10`.

6. Artificial features

Observing the high impact of feature selection on our results, we hypothesised that creating some additional artificial features of our own with which to supplement *Feature Set B* could leverage our domain expertise as humans and help our model perform better.

Note from our earlier Figure 1 that our binary features **coffee_bar**, **video_store**, **salad_bar**, **prepared_food** and **florist** all possess a high-magnitude Pearson's correlation against each other. It seems plausible that large stores will tend to have all or multiple of these, whereas smaller stores will not have enough space. Consequently, we can create our first artificial feature of **facility_score** as the mean value of these five binary features (we add all five and divide by the number of features added). With the hypothesised relationship of store size to the presence of

these facilities in mind (especially considering that **store_sqft** correlates with multiple of the mentioned binary features), we create our second artificial feature, **facility_score_ratio**, to normalise for this relationship by computing **store_sqft / facility_score**.

We can also note another correlated cluster in the form of **total_children**, **num_children_at_home** and **avg_cars_at_home**. It seems likely that being characteristics of the families living in an area, aggregating these into one additional feature could be useful. This motivates **family_score** as the mean of the three family-related features, which provides us with summarised information about the customers' households.

The last artificial feature we created was called **sales_per_unit**, which equals **store_sales / unit_sales**. This feature provides us with the average price at which a store sells items.

Table 4 performs an ablation test of some plausible combinations of these artificial features carried out with the same model, hyperparameters, and training and evaluation procedure as in Section 5. Based on this, we selected **facility_score** and **facility_score_ratio** as the only artificial features to augment *Feature Set B*. This achieved a cross-validated RMSLE of **0.292916**, our highest so far. Going forward, the augmentation of *Feature Set B* with **facility_score** and **facility_score_ratio** is termed *Feature Set C*.

Model used	Feature Set	Cross-validated RMSLE
LightGBM (as before)	<i>Feature Set B (baseline)</i>	0.292931
LightGBM (as before)	+ facility_score	0.292931
LightGBM (as before)	+ facility_score_ratio	0.292918
LightGBM (as before)	+ facility_score , facility_score_ratio	0.292916
LightGBM (as before)	+ family_score	0.292938 (-baseline)
LightGBM (as before)	+ sales_per_unit	0.293564 (-baseline)

Table 6: Comparison of different choices of artificial features

7. Automatic selection of hyperparameters

Previously in this report, we have made use of a single set of hyperparameter values for LightGBM which were chosen via manual trial-and-error. Ideally, we would have some kind of evidence motivating our choice of hyperparameter values. We decided to, using *Feature Set C* from Section 6 and keeping the rest of our hyperparameters the same as before, make use of an exhaustive grid search to examine our choice of $n_estimators$ and num_leaves for the LightGBM model. For $n_estimators$, we took 20 values in the interval $[50, 1000]$; for num_leaves , 20 values in the interval $[50, 200]$.

Plotting a 3D surface plot (Figure 4) with the maximum cross-validated RMSLE clamped to 0.294 subsequently shows us a clear “trough” in the value of the loss function corresponding to a certain range of hyperparameter choices. The lowest cross-validated RMSLE achieved is 0.292894, with $n_estimators=650$ and $num_leaves=57$. However, although this is even lower than the best cross-validated RMSLE of Section 6, we must consider that a hyperparameter search like this is essentially a way of actively optimising the cross-validation score, and thus the CV score cannot be used to predict the performance of the resulting model as it is akin to a training accuracy.

Instead, we submit test set predictions with the best model of section 6 versus the model with optimised hyperparameters; the resulting Kaggle RMSLE is 0.2926 for the former and 0.29824 for the latter, indicating that we have overfit to the cross-validation procedure. Our resultant recommendation is to not use grid search to find optimal hyperparameter values.

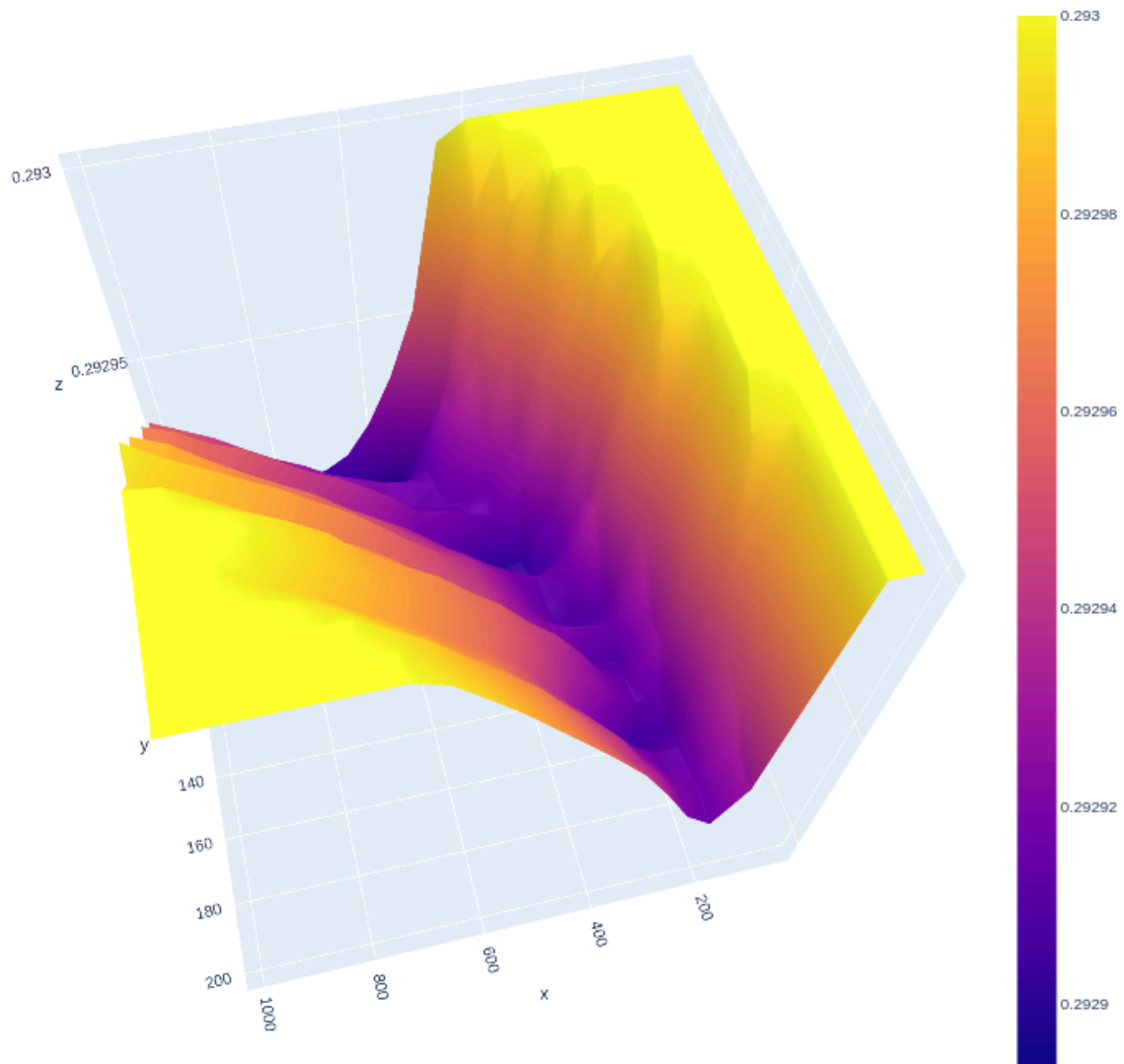


Figure 4: The RMSLE against $n_estimators$ on the x-axis and num_leaves on the y-axis.

8. Test-set ablation of the best model

Previously, we have made use of the cross-validated RMSLE to make various decisions about our machine-learning system. We now perform an ablation test (Table 7) with the best models so far, showing both the cross-validated RMSLE and the test-set RMSLE. The test-set RMSLE computation is performed by Kaggle, with us only submitting the predictions.


Model	Cross-validated RMSLE	Test-set RMSLE
LightGBM, Manually tuned, No feature selection, Only the synthetic dataset	0.296616	0.29609
LightGBM, Manually tuned, No feature selection, Synthetic+original dataset	0.296076	0.29568
LightGBM, Manually tuned, Feature Set B, Synthetic+original dataset	0.292931	0.29264
LightGBM, Manually tuned, Feature Set C (artificial features), Synthetic+original dataset	0.292916	0.29260
LightGBM, Automatically tuned (grid search), Feature Set C (artificial features), Synthetic+original dataset	0.298294	0.29284

With the exception of the model which was automatically tuned to maximise the cross-validated RMSLE by grid search, our cross-validated RMSLE tends to correspond to the true test-set RMSLE, validating our practice of making decisions based on cross-validated RMSLE. The best-performing model is ultimately the LightGBM with manually-tuned hyperparameters trained on *Feature Set C* and the concatenation of our synthetic and original datasets, developed in Sections 5 and 6.

9. Final score

Our best test-set RMSLE as of the ablation in Section 7 was **0.29260**, placing us at **position #6** on the public leaderboard. This corresponds to the **top 1%**.


× Submission Details




predict_lgbm_3.csv
Complete (after deadline) · 2mo ago

Score: 0.29260
Private score: 0.29327

UPLOADED FILES

 predict_lgbm_3.csv (5 MiB)



DESCRIPTION

Enter a description

0 / 500

SELECT FOR FINAL SCORE

☐ Select this submission to be scored for your final leaderboard score

Figure 5: Final RMSLE score obtained in the competition

Regression with a Tabular Media Campaign Cost Dataset

Public Private

This leaderboard is calculated with approximately 20% of the test data. The final results will be based on the other 80%, so the final standings may be different.

#	Team	Members	Score	Entries	Last	Solution
1	benedikt.d		0.29259	55	1y	
2	AmbrosM		0.29259	3	1y	
3	Sergey Saharovskiy		0.29260	71	1y	
4	Möbius		0.29260	51	1y	
5	yann barthelemy		0.29260	30	1y	
6	Joseph Z		0.29261	14	1y	
7	DJ_C		0.29261	41	1y	
8	SOUNA Sara		0.29261	5	1y	
9	TAUIL Abd Ellah		0.29261	5	1y	
10	parovka team		0.29261	16	1y	
11	Matt OP		0.29261	26	1y	
12	Iqbal Syah Akbar		0.29261	30	1y	
13	chaserendall		0.29261	22	1y	

Figure 6: Public leaderboard of the Kaggle competition

10. Conclusion

The recurring theme of this report is the importance of quantitative evaluations; the choices which ultimately produced our best-performing predictions were motivated entirely by the cross-validation scores reported under different choices.

Careful selection of training data proved vital, with the addition of the original dataset from which our synthetic data was generated, the selection of those features with the highest mutual information with the target variable, and the addition of certain artificial features all acting to increase our LightGBM's performance.

Although successfully reducing our cross-validated RMSLE to its minimum, the automatic selection of hyperparameters using a grid search ultimately illustrated the dangers of overfitting; although the resulting model learned to produce the lowest cross-validated RMSLE (the optimisation target of the grid search), it failed to produce a low test-set RMSLE.

References

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. (2021). Review of Deep Learning: Concepts, CNN Architectures, challenges, applications, Future Directions. *Journal of Big Data*, 8(1). <https://doi.org/10.1186/s40537-021-00444-8>

Dutta, G. (2023). Media Campaign Cost Prediction (Supplemental Dataset). <https://www.kaggle.com/datasets/gauravduttakiit/media-campaign-cost-prediction>

Venkatesh, B., & Anuradha, J. (2019). A review of feature selection and its methods. *Cybernetics and Information Technologies*, 19(1), 3–26. <https://doi.org/10.2478/cait-2019-0001>

Xiang Sean Zhou, Comaniciu, & Krishnan. (2003). Conditional feature sensitivity: A unifying view on active recognition and feature selection. *Proceedings Ninth IEEE International Conference on Computer Vision*. <https://doi.org/10.1109/iccv.2003.1238668>