Mercari Price Suggestion Challenge

Hyun woo Kim | Jin woo Na | Eun-guk Cho | **Team 11**

28th May, 2026

Index

1. Problem Definition
2. Exploratory Data Analysis (EDA)
3. Baseline Model Architecture
4. Advanced Strategies & Enhancements
5. Experimental Results
6. Summary & Key Takeaways

Problem Definition

Mercari Price Suggestion Challenge



- Build a regression model that automatically **suggests prices for items listed on Mercari**, given product information such as item name, brand name, and condition, etc.
- Mercari is **Japan's largest C2C marketplace**, serving as a vast platform where millions of individual sellers list and trade used goods.
- Accurate price prediction is crucial because individual sellers lack **professional valuation skills**; dynamic pricing guides them to set competitive rates, which directly accelerates transaction speeds and increases overall platform satisfication.

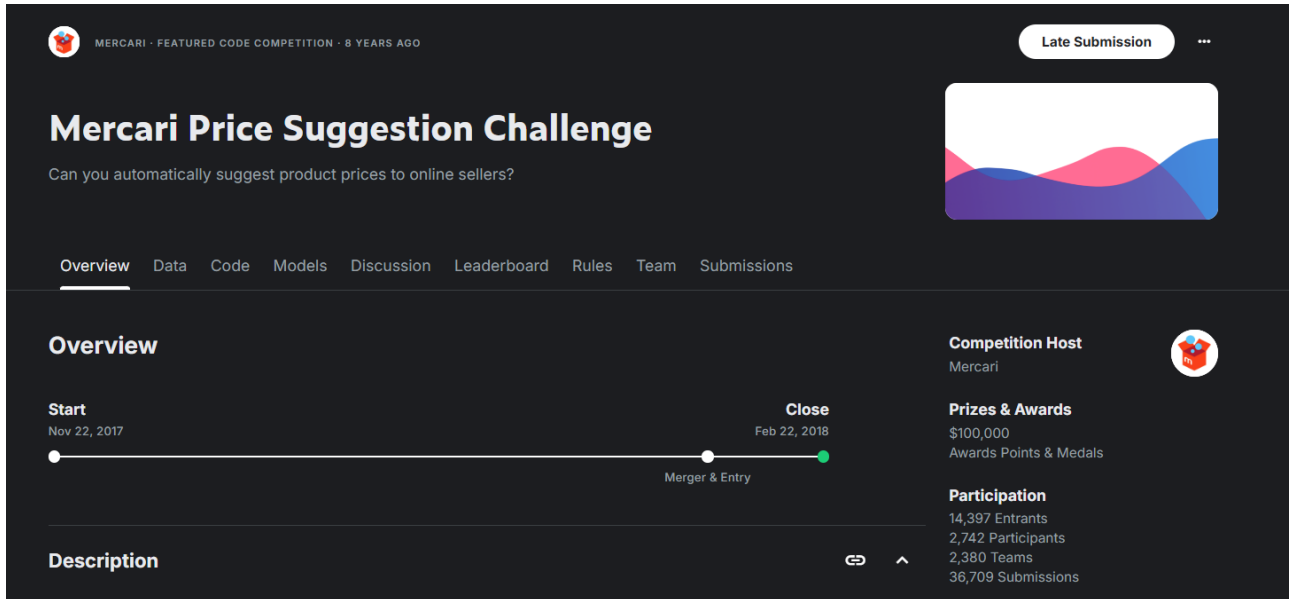


Problem Definition

Mercari Price Suggestion Challenge



- Build a regression model that automatically **suggests prices for items listed on Mercari**
- Held in 2018, and 2381 teams participated (leaderboard)
- **Kernels-only environment**: 60-min time limit, 16GB RAM, and 4-Core CPU
- Evaluate prediction performance using **Root Mean Squared Logarithmic Error (RMSLE)**, leveraging its asymmetric penalty structure that strictly penalizes underestimation.



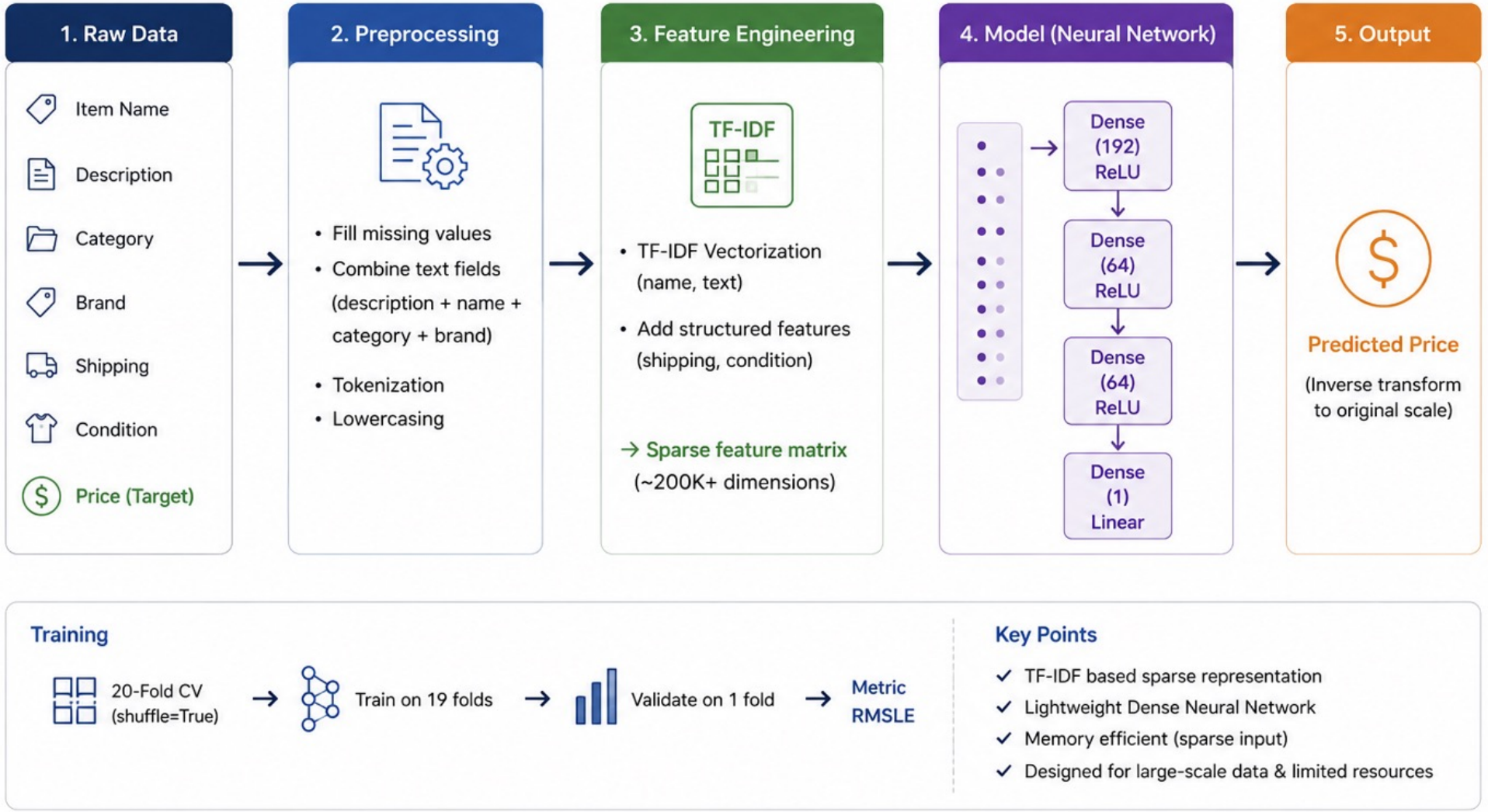
$$\varepsilon = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\log(p_i+1)-\log(a_i+1)\right)^2}$$

Where:

ε	is the RMSLE value (score)
n	is the total number of observations in the (public/private) data set,
p_i	is your prediction of price, and
a_i	is the actual sale price for i.
$\log(x)$	is the natural logarithm of x

Exploratory Data Analysis (EDA)

- Overall design from raw data to processed input and output



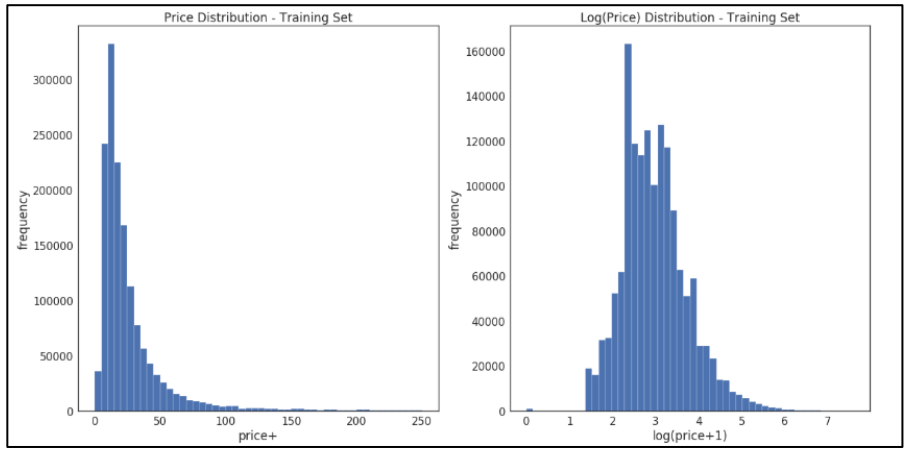
Exploratory Data Analysis (EDA)

	train_id	name	item_condition_id	category_name	brand_name	price	shipping	item_description
0	0	MLB Cincinnati Reds T Shirt Size XL	3	Men/Tops/T-shirts	NaN	10.0	1	No description yet
1	1	Razer BlackWidow Chroma Keyboard	3	Electronics/Computers & Tablets/Components & P...	Razer	52.0	0	This keyboard is in great condition and works ...
2	2	AVA-VIV Blouse	1	Women/Tops & Blouses/Blouse	Target	10.0	1	Adorable top with a hint of lace and a key hol...
3	3	Leather Horse Statues	1	Home/Home Décor/Home Décor Accents	NaN	35.0	1	New with tags. Leather horses. Retail for [rm]...
4	4	24K GOLD plated rose	1	Women/Jewelry/Necklaces	NaN	44.0	0	Complete with certificate of authenticity

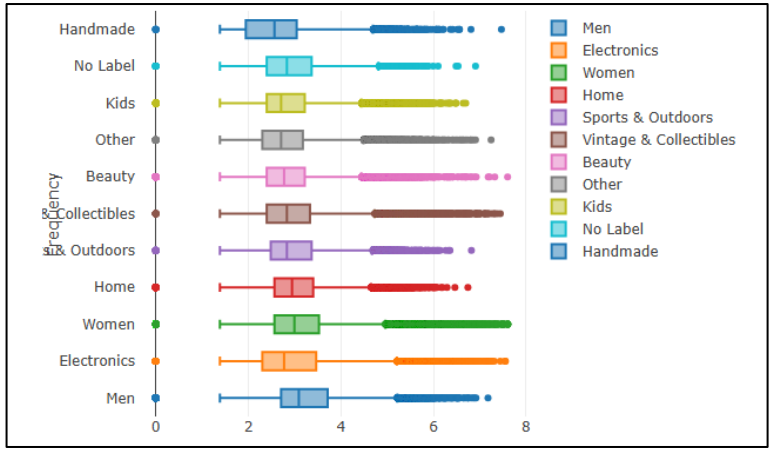
< Data Structure >

- 4 text input (name, category, brand, description)
- 2 categorical input (condition, shipping)
- output (price)

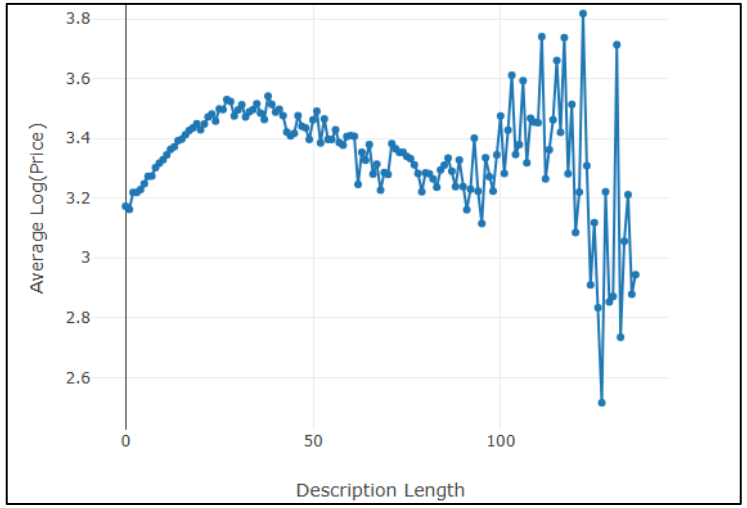
1,482,535 train / 3,460,725 test rows



< Price Distribution>



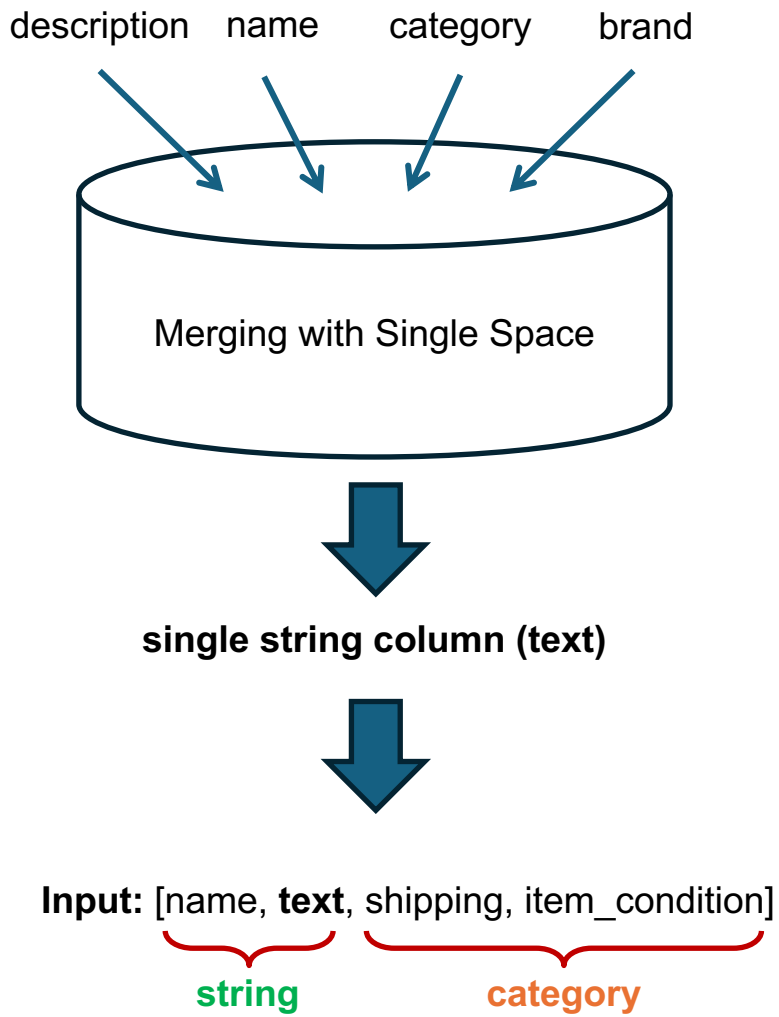
< Price Distribution by Category>



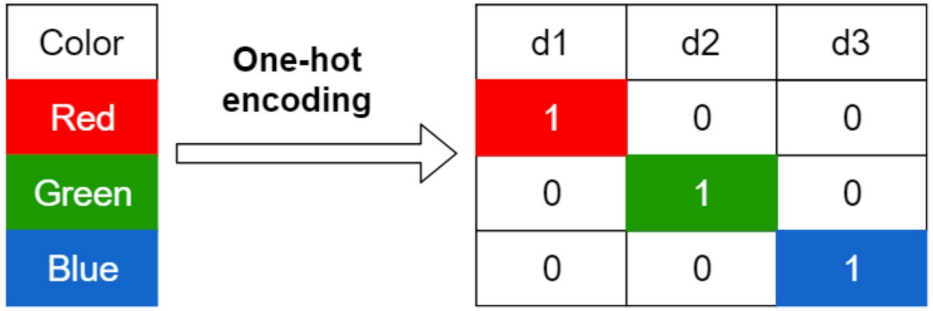
< Average Log(price) by Description Length>

Baseline Model Architecture

Preprocessing & Feature Engineering



- Replaced **missing values with empty strings** (‘ ‘) with Pandas.
- Text Concatenation: Following the consensus in Kaggle discussions that complex feature engineering yields minimal performance gains, **all text fields were merged with a single space.**
- TF-IDF (Python Package): Adjusts word weights based on term frequency and importance
- Applied **One-Hot Encoding to categorical data** (shipping, item_condition) to save memory and prevent numerical bias.



< One-Hot Encoding>

Baseline Model Architecture

Why we have selected sparse MLP

- Three lightweight models were initially tested, leading to the **selection of the Multilayer Perceptron (MLP).**
- Since LightGBM violated the inference time constraints on the large-scale text dataset, heavier architectures (e.g., CNN, LSTM) were excluded from experimentation.

Model Architecture	RMSLE (Public)	Time Feasibility
Ridge Regression	0.44228	O
LightGBM	0.45480	X
Sparse MLP	**0.39842**	**O**

[Base Parameter]

Ridge Regression

- solver: sag
- alpha: 3.0

LightGBM

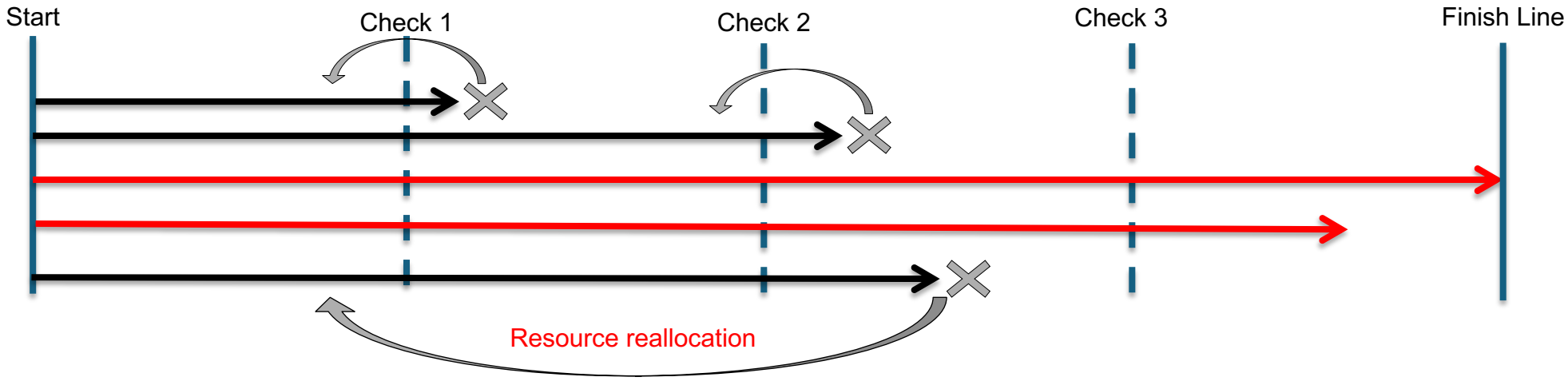
- max_bin: 63
- colsample_bytree: 0.4
- num_leaves: 127

Sparse MLP

- Hidden Layers: 128 * 64
- Loss: MSE
- Batch Size: 2048

Advanced Strategies & Enhancements

Hyperparameter Optimization with Optuna



- **Hyperparameter optimization framework:** highly scalable, tailored for Python
- **Define-by-Run API**: Dynamically constructs the search space at runtime for flexible parameter
- **Asynchronous Pruning**: Instantly early-stops unpromising trials to **reallocate resources**

Independent parameter search (4 models)

- Hidden layer dimensions (128, 192, 256)
- Learning rate, Batch size, and Epochs
- MSE vs. Huber Loss (for outlier handling)

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019, July). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 2623-2631).

Advanced Strategies & Enhancements

Ensemble Construction with Quadratic Programming

loss: $\underbrace{\|Pw - y\|^2}_{\text{loss}} + \underbrace{\lambda\|w\|_2^2}_{\text{regularization}} = w^T P^T P w - 2y^T P w + y^T y + \lambda w^T w$

where,

N: *Total number of models to be ensemble*

P: *Matrixes built from predictions of individual models*

w: *The model − specific weight vector*

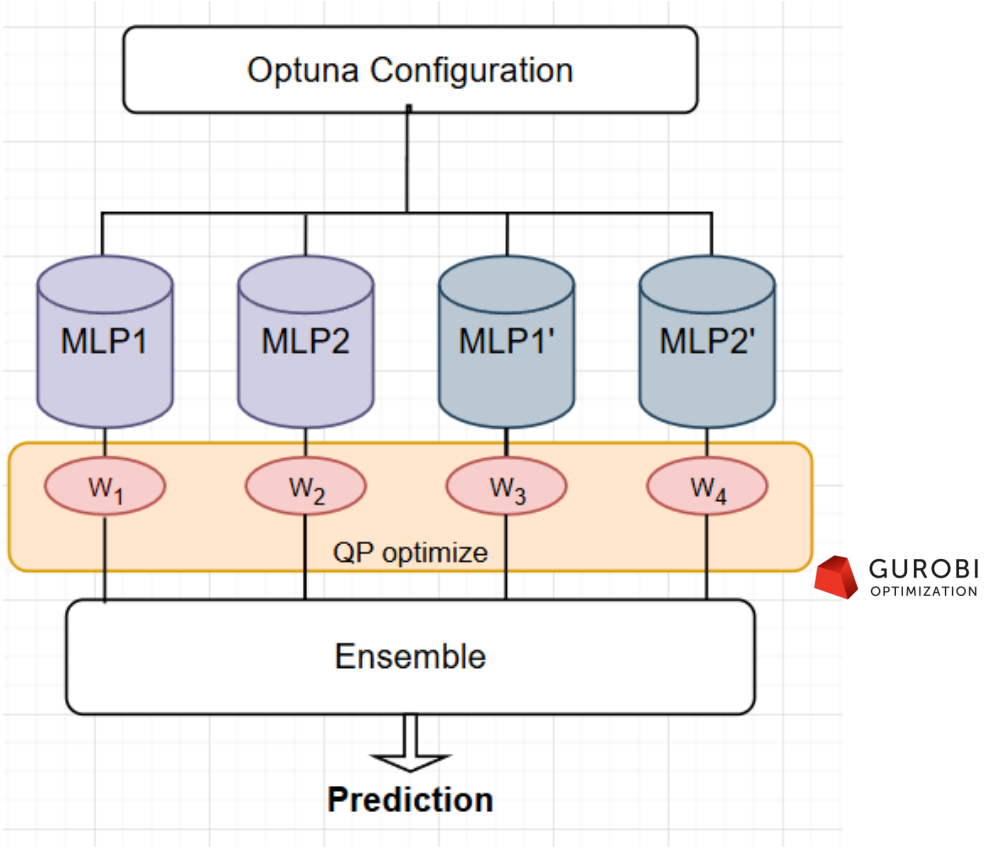
y: *Actual target vector*

λ: *L2 regularization hyperparameter*

$$\min_w \; w^T(P^T P + \lambda I)w - 2(P^T y)^T w$$

$$s.t. \sum_{i=1}^{N} w_i = 1$$

$$0 \le w_i \le 1, \quad \forall i \in \{1, .., N\}$$



- **OR Domain-Driven Optimization**: Determines optimal ensemble weights via Quadratic Programming (QP) with Gurobi.
- **Overfitting Mitigation**: Incorporates an L2 regularization penalty to prevent weight concentration and enhance robustness.

Experimental Results

[Final Configuration]

Model1 (MSE)

lr = 3e-3 / batch=2048 / layer=192*64*64 / epochs=3
Ensemble Weight = [0.174, 0.207]

Model2 (MSE)

lr = 3e-3 / batch=2048 / layer=192*64*64 / epochs=3
Ensemble Weight = [0.235, 0.228]

Model3 (Huber)

lr = 2.132 e-3 / batch=2048 / layer=192*64*64 / epochs=2
Ensemble Weight = [0.236, 0.254]

Model4 (Huber)

lr = 7.424 e-3 / batch=1024 / layer=256*64*64 / epochs=3
Ensemble Weight = [0.355, 0.310]

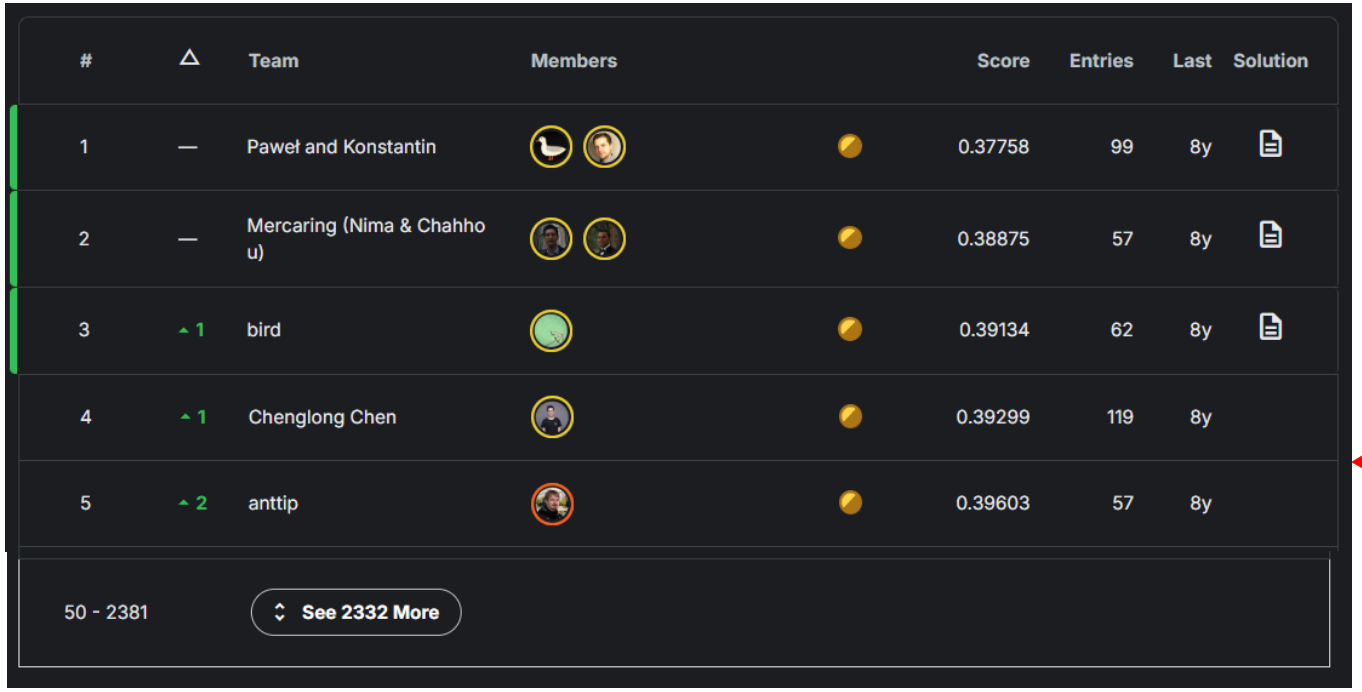
[Scores (private)]

Stage	Optimization Methodology	RMSLE (Private)	improvement rate (Δ)
#1	Base MLP Ensemble	0.39739	baseline
#2	Tuned Ensemble (MLP + Optuna)	0.39479	-0.00260
#3	QP Ensemble (MLP + Optuna + QP)	0.39395	- 0.00344
Final	Restricted QP Ensemble (MLP + Optuna + QP + $\lambda_{0.1}$)	0.39384	- 0.00355

- **Lambda Optimization**: Tested multiple lambda values (0, 0.05, 0.1, 0.5, 1, 5), identifying **0.1 as the optimal threshold**.
- **Structural Effectiveness**: Confirmed that the proposed method **captures data structures more effectively** than a standard MLP.

Experimental Results





#	△	Team	Members	Score	Entries	Last	Solution
1	—	Paweł and Konstantin		0.37758	99	8y	
2	—	Mercaring (Nima & Chahhou)		0.38875	57	8y	
3	▲ 1	bird		0.39134	62	8y	
4	▲ 1	Chenglong Chen		0.39299	119	8y	
5	▲ 2	anttip		0.39603	57	8y	

50 - 2381 See 2332 More

Our Model is here
(5th / 2381)

- We have **submitted 36 versions** and achieved a best competitive private score of 0.39384, placing our model in the **top tier (5th) among 2,381 participating teams.**

Summary & Key Takeaways

- **Summary**
  - Addressed the **Mercari Price Suggestion Challenge** by predicting optimal listing prices within a highly constrained infrastructure (60-min inference, 16GB RAM).
  - We tested various models and selected **a Sparse MLP**, followed by applying **Optuna for parameter tuning** and **Quadratic Programming for ensemble optimization**. Each method showed a distinct improvement rate, ultimately achieving a high performance rank.

- **Key Takeaways**
  - Selecting the **appropriate model** and implementing **effective preprocessing** are paramount.
  - Excluding the tuning phase, our model consists of only about 150 lines of code, whereas typical models range from 500 to 2,000 lines. This shows the **high efficiency of a simple model under limited hardware and time**. Instead of relying on a heavy model from the start, **starting with a simpler model** is a much better approach.
  - Tuning Approach – Various ideas from IME also can be implemented in the AI domain. (IME ➔ AI)
    (EX. IMEN242(Experimental Design), IMEN260(Operations Research), ...)

Thank you