Housing Prices Competition for Kaggle Learn Users

Apply what you learned in the Machine Learning course on Kaggle Learn along...

Getting Started · 5024 Teams · Ongoing

Housing Prices Competition for Kaggle Learn Users

https://www.kaggle.com/competitions/home-data-for-ml-course

庭師

ChatGPT 5.2

Kaggle環境で実行

Kaggle Learnユーザー向け住宅価格コンペ

- コンテストの説明
  - 住宅購入者に夢のマイホームについて尋ねても、地下室の天井の高さや東西を結ぶ鉄道への近さから話を始める人はまずいないでしょう。しかし、この遊び場型競争のデータセットは、寝室の数や白いピケットフェンスよりもはるかに多くの要素が価格交渉に影響を与えることを証明しています。
  - このコンテストでは、アイオワ州エイムズの住宅の（ほぼ）あらゆる側面を説明する 79 個の説明変数を使用して、各住宅の最終価格を予測することが求められます。
- 練習スキル
  - クリエイティブな機能エンジニアリング
  - ランダムフォレストや勾配ブースティングなどの高度な回帰手法

評価

- ゴール
- 各住宅の販売価格を予測することがあなたの仕事です。テストセット内の各IDについて、SalePrice変数の値を予測してください。
- メトリック
- 提出されたデータは、予測値の対数と実際の販売価格との間の二乗平均平方根誤差（RMSE）に基づいて評価されます 。（対数をとるということは、高価な住宅と安価な住宅の予測誤差が結果に等しく影響することを意味します。）
- *Root mean square deviation*（二乗平均平方根偏差）

$$\mathrm{RMSD}(\hat{\theta}) = \sqrt{\mathrm{MSE}(\hat{\theta})} = \sqrt{\mathrm{E}\left((\hat{\theta} - \theta)^2\right)}.$$

$$\mathrm{RMSD} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_i - x_0)^2}.$$

データ

- ファイルの説明
- train.csv – トレーニングセット
- test.csv – テストセット
- data_description.txt – 各列の完全な説明。元々は Dean De Cock によって作成されましたが、ここで使用されている列名に合わせて軽く編集されています。
- sample_submission.csv – 販売年と月、敷地面積、寝室数に関する線形回帰のベンチマーク提出

データフィールド

- **セール価格**- 不動産の売却価格(ドル)。これが予測対象となるターゲット変数です。
- **MSSubClass**：建物クラス
- **MSZoning**：一般的なゾーニング分類
- **敷地間口**:敷地に接する道路の長さ
- **LotArea**：敷地面積(平方フィート)
- **道路**:道路アクセスの種類
- **路地**:路地へのアクセスの種類
- **LotShape**：土地の一般的な形状
- **土地の輪郭**:土地の平坦さ
- **ユーティリティ**:利用可能なユーティリティの種類
- **LotConfig**：ロット構成
- **土地の傾斜**:土地の傾斜
- **近隣地域**:エイムズ市域内の物理的な場所
- **条件1**:主要道路または鉄道に近い
- **条件2**:主要道路または鉄道への近さ(2番目が存在する場合)
- **BldgType**：住居の種類
- **ハウススタイル**:住居のスタイル
- **全体的な品質**:全体的な素材と仕上げの品質
- **全体的な状態**:全体的な状態評価
- **建築年**:元の建設日
- **YearRemodAdd**：改装日
- **RoofStyle**：屋根の種類
- **RoofMatl**:屋根材
- **外装1**:家の外装
- **外装2**:家の外装材(複数の素材がある場合)
- **MasVnrType**：石積みベニヤタイプ
- **MasVnrArea**：石積みベニヤ面積(平方フィート)
- **ExterQual**:外装材の品質
- **ExterCond**:外装材の現状
- **基礎**:基礎の種類
- **BsmtQual**：地下室の高さ
- **地下室の状態**:地下室の一般的な状態
- **地下室露出**:ウォークアウトまたはガーデンレベルの地下室の壁
- **BsmtFinType1**：地下室仕上げエリアの品質
- **BsmtFinSF1**：タイプ1の完成面積
- **BsmtFinType2**：2番目の仕上げ領域の品質(存在する場合)
- **BsmtFinSF2**：タイプ2の完成面積
- **BsmtUnfSF**：地下室の未完成面積(平方フィート)
- **TotalBsmtSF**：地下室の総面積(平方フィート)
- **暖房**:暖房の種類
- **暖房品質**:暖房の品質と状態
- **CentralAir**:セントラル空調
- **電気**:電気システム
- **1stFlrSF**:1階の平方フィート
- **2ndFlrSF**：2階の面積(平方フィート)
- **LowQualFinSF**：低品質仕上げ平方フィート(全フロア)
- **GrLivArea**:地上居住面積(平方フィート)
- **BsmtFullBath**：地下のフルバスルーム
- **BsmtHalfBath**：地下の半分のバスルーム
- **FullBath**:地上階にフルバスルームあり
- **HalfBath**:地上にある半浴室
- **寝室**:地下階より上の寝室の数
- **キッチン**:キッチンの数
- **KitchenQual**:キッチンの品質
- **TotRmsAbvGrd**：地上階の部屋数(バスルームは含まない)
- **機能**:住宅の機能性評価
- **暖炉**:暖炉の数
- **FireplaceQu**：暖炉の品質
- **ガレージタイプ**:ガレージの場所
- **GarageYrBlt**：ガレージが建てられた年
- **GarageFinish**:ガレージの内装仕上げ
- **GarageCars**:車の収容能力における ガレージのサイズ
- **ガレージ面積**:ガレージの面積(平方フィート)
- **GarageQual**：ガレージの品質
- **GarageCond**：ガレージの状態
- **PavedDrive**:舗装された私道
- **WoodDeckSF**：ウッドデッキ面積(平方フィート)
- **OpenPorchSF**：オープンポーチ面積(平方フィート)
- **囲い付きポーチ**:囲い付きポーチの面積(平方フィート)
- **3SsnPorch**:3シーズンポーチ面積(平方フィート)
- **スクリーンポーチ**:スクリーンポーチの面積(平方フィート)
- **プール面積**:プール面積(平方フィート)
- **PoolQC**：プールの品質
- **フェンス**:フェンスの品質
- **MiscFeature**：他のカテゴリに含まれないその他の機能
- **MiscVal**：その他の機能の値
- **MoSold**：販売月
- **YrSold**：販売年
- **販売タイプ**:販売の種類
- **販売条件**:販売条件

このコンペで最上位スコアを出す秘訣を教えて。

- 上位スコアの秘訣
- ✓ 徹底的な EDA と欠損値・外れ値処理
  ✓ 良質な特徴量をどれだけ作れるか（これが最大の差）
  ✓ 複数モデルのアンサンブルとチューニング
  ✓ CV と Leaderboard の整合性を取る
  ✓ 他者参加ノートブックを学びつつ独自改善する

House Prices - Advanced Regression Techniques
Predict sales prices and practice feature engineering, RFs, and gradient boosting
Getting Started · 5973 Teams · Ongoing

スコア帯（Log(RMSE)）	参加者の目安層
≤ 0.12	上位数% / 高精度モデル
0.12〜0.15	多くの中級者・上位志向の提出例
0.15〜0.20	初〜中級者中心
> 0.20	基礎的モデル中心 / 初心者が多い

このコンペのスコア分布を教えて。

今回はこっち！

Housing Prices Competition for Kaggle Learn Users
Apply what you learned in the Machine Learning course on Kaggle Learn along...
Getting Started · 5024 Teams · Ongoing

スコア：RMSE	評価
30000以上	ほぼベースライン
20000前後	普通
17000台	良い（理解できている）
15000台	上位寄り
12000台	かなり良い
0	無視枠

train.csv 最初の5行と概要

```
   Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape
0   1          60       RL         65.0     8450   Pave   NaN      Reg
1   2          20       RL         80.0     9600   Pave   NaN      Reg
2   3          60       RL         68.0    11250   Pave   NaN      IR1
3   4          70       RL         60.0     9550   Pave   NaN      IR1
4   5          60       RL         84.0    14260   Pave   NaN      IR1

  LandContour Utilities  ... PoolArea PoolQC Fence MiscFeature MiscVal
0         Lvl    AllPub  ...        0    NaN   NaN         NaN       0
1         Lvl    AllPub  ...        0    NaN   NaN         NaN       0
2         Lvl    AllPub  ...        0    NaN   NaN         NaN       0
3         Lvl    AllPub  ...        0    NaN   NaN         NaN       0
4         Lvl    AllPub  ...        0    NaN   NaN         NaN       0

  YrSold  SaleType  SaleCondition  SalePrice
0   2008        WD         Normal     208500
1   2007        WD         Normal     181500
2   2008        WD         Normal     223500
3   2006        WD        Abnorml     140000
4   2008        WD         Normal     250000

[5 rows x 81 columns]
```

項目	値
行数	1,460
列数	81
数値特徴量	38
カテゴリ特徴量	43
目的変数	SalePrice

👉 カテゴリ変数が多い（半分以上）
👉 欠損が「意味を持つ列」と「単なる欠損」が混在しているのが最大の特徴

欠損値の状況（超重要）

🔑 **重要なポイント**

- **これは欠損 ≠ データ欠落**
- **多くは「その設備が存在しない」ことを表す**

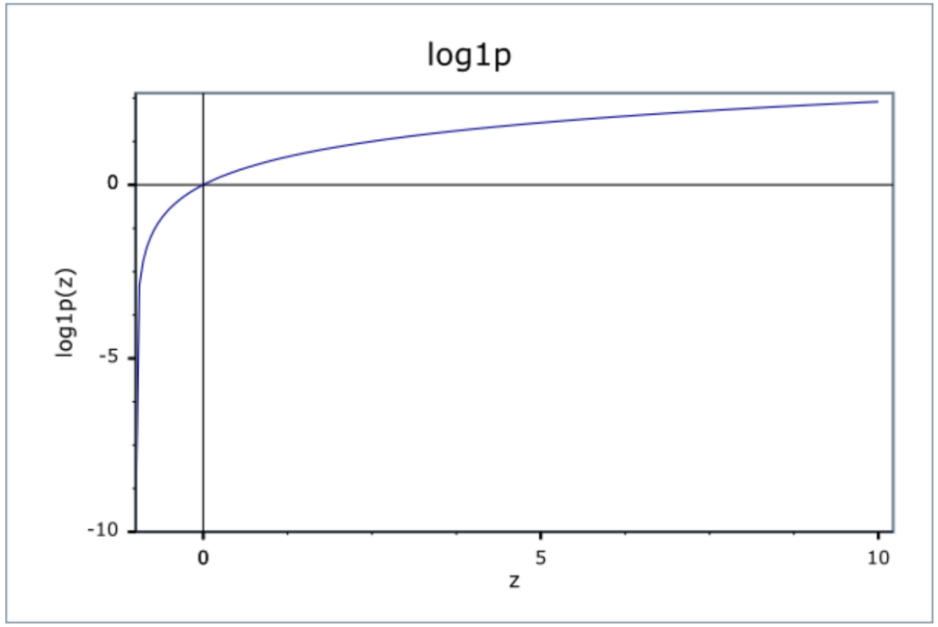
👉 **上位解法では**
NaN → 'None' / 0 / Neighborhood中央値
のように「意味を持たせて補完」する

特徴量	欠損率	解釈
PoolQC	99.5%	プールが「無い」家がほとんど
MiscFeature	96.3%	その他設備なし
Alley	93.8%	路地アクセスなし
Fence	80.8%	フェンスなし
MasVnrType	59.7%	化粧レンガなし
FireplaceQu	47.3%	暖炉なし
LotFrontage	17.7%	区画条件で欠損

目的変数 SalePrice の特徴

- 強い 右裾の長い分布（右に歪み）
- 高価格帯が少数存在
- 👉 ほぼ全ての上位解法で
- SalePrice_log = np.log1p(SalePrice)
- を使う
- ⚠️ これをやらないと RMSE が一気に悪化

$$y_i = \log_e(x_i + 1)$$



numpy.log1p()

numpy.log1p(x) は e を底とした 1 + x の対数を計算します。この関数は真数が 1 に近い値のときの対数を正確に計算するために用意されています。たとえば、numpy.log1p(0.001) は numpy.log(1.001) よりも高い精度の値を返します。

数値特徴量の傾向

- 強い特徴量(定番)
  - OverallQual(住宅の総合品質)
  - GrLivArea(地上居住面積)
  - TotalBsmtSF
  - GarageCars, GarageArea
  - YearBuilt, YearRemodAdd
- 👉 単体でも SalePrice と強相関
- 注意すべき外れ値
  - GrLivArea > 4000 で 価格が異常に低い家
  - これを除去すると CV / LB が改善しやすい

カテゴリ変数の特徴

- Neighborhood が 最重要カテゴリ特徴
- MSZoning, BldgType, HouseStyle, SaleCondition なども効く
- 👉 Tree 系(LightGBM / XGBoost)では
  One-Hot だけでも十分強い
- 👉 線形モデルでは
  Target Encoding / Ordinal Encoding が効く

- 年度・時間系の見方(重要)
- YrSold は 2006-2010 に集中
- 強い時系列性はないが、
  - HouseAge = YrSold - YearBuilt
  - RemodAge = YrSold - YearRemodAdd
    などの派生特徴は非常に有効

このコンペ特有の「EDA結論」

- ✅ 上位スコアを出すための前提条件
  - SalePrice を log1p
  - 欠損は「None / 0 / 地域中央値」で意味付け
  - 外れ値(GrLivArea系)を除去
  - 面積・築年数の派生特徴を作る
  - カテゴリは雑に捨てない(全部使う)

上位を狙う上で重要な、代表的feature を説明して（1）

- ① OverallQual（総合品質）【最重要】
- 意味
  - 家全体の「質」を 1〜10 で評価した指標
- なぜ重要？
  - 価格との相関が 全特徴量中トップクラス
  - 立地・設備・内装の総合評価に近い
- 使い方のポイント
  - そのまま使っても強い
  - OverallQual × 面積 などの掛け合わせも有効

- ② GrLivArea（地上居住面積）
- 意味
  - 地上階の居住スペース合計（㎡）
- なぜ重要？
  - 「広い家＝高い」は極めて直感的
  - SalePrice とほぼ線形関係
- 注意点（超重要）
  - 異常値（広いのに安い家）が存在
  - 上位者は必ず外れ値を除去している

上位を狙う上で重要な、代表的feature を説明して（2）

- ③ Neighborhood（地域）
- 意味
  - 住宅が属する地区名
- なぜ重要？
  - 立地の価値を最もよく表す
  - 同じ家でも地区で価格が大きく変わる
- 使い方のポイント
  - One-Hot でも十分強い
  - 欠損補完（LotFrontage など）に使うとさらに効く

- ④ TotalBsmtSF（地下室面積）
- 意味
  - 地下室の総面積
- なぜ重要？
  - 地下室は収納・居住スペースとして価値が高い
  - GrLivArea と並ぶ「面積系の柱」
- 注意点
  - 地下室が「無い家」は 0 として扱うのが基本

上位を狙う上で重要な、代表的feature を説明して（3）

- ⑤ YearBuilt / YearRemodAdd（築年・改築年）
- 意味
  - 家が建てられた年 / 改築された年
- なぜ重要？
  - 新しい家ほど価格が高い傾向
  - 改築が価格を押し上げる効果あり
- 上位者の工夫
  - HouseAge = YrSold − YearBuilt
  - RemodAge = YrSold − YearRemodAdd
    → 年そのものより「築年数」の方が効く

- ⑥ GarageCars / GarageArea（ガレージ）
- 意味
  - 駐車可能台数 / 面積
- なぜ重要？
  - アメリカ住宅では車は必須
  - 0台 vs 2台以上で価格差が大きい
- 注意点
  - 欠損 = ガレージなし
  - NaN を 0 にするのが定石

上位を狙う上で重要な、代表的feature を説明して（4）

- ⑦ ExterQual / KitchenQual(品質評価系)
- 意味
  - 外装・キッチンの品質(Ex/Gd/TA/Fa)
- なぜ重要？
  - 見た目・使い勝手の良さは価格に直結
  - カテゴリだが情報量が非常に高い
- 使い方
  - Ordinal Encoding(Ex=5, Gd=4…)が有効
  - Tree系でも自然に効く

- ⑧ MasVnrArea(化粧レンガ面積)
- 意味
  - 外壁の装飾レンガの面積
- なぜ重要？
  - 高級感の指標
  - 面積は小さくても価格差に効く
- 注意点
  - 欠損は「無し」として 0 扱い

上位を狙う上で重要な、代表的feature を説明して（5）

- ⑨ SaleCondition / SaleType（売却条件）
- 意味
  - 通常売却か、競売・特殊契約か
- なぜ重要？
  - 「Normal」以外は価格が下がりやすい
  - 外れ値的価格を説明する補助特徴
- ⑩ 派生特徴（上位者が必ず作る）
- そのままより強い特徴量例
  - TotalSF = GrLivArea + TotalBsmtSF
  - HouseAge
  - RemodAge
  - QualityArea = OverallQual × TotalSF
- 👉 単体特徴より派生特徴が効くのがこのコンペの本質

まとめ（超重要）

- 上位者の feature 思考
  - ❌ 列をそのまま全部入れる
  - ⭕「これは家の価値をどう表しているか？」で理解する
- 本当に重要なのは
  - 品質（OverallQual）
  - 広さ（面積系）
  - 立地（Neighborhood）
  - 新しさ（築年・改築）
  - 派生特徴（組み合わせ）

重要 feature だけで作る最強ベースラインをKaggle実行用につくって

- 「重要 feature だけ」に絞って、LB 0.12台を安定して狙える
  Kaggle 実行用・最強ベースライン をそのまま使える形で出します。
- 👉 コピペして Kaggle Notebook で即実行可
  👉 無駄な特徴・過学習要素なし
  👉「なぜ効くか」が分かる構成

- 🏆 最強ベースライン方針（重要）
- 目的変数：log1p(SalePrice)
- モデル：LightGBM
- 特徴量：
  - 品質
  - 面積
  - 築年数
  - 立地
  - 最小限の派生特徴
- 欠損：意味を保った補完のみ
- 外れ値：GrLivArea の異常値を除去

使用する重要 feature 一覧

- OverallQual
- GrLivArea
- TotalBsmtSF
- GarageCars
- GarageArea
- YearBuilt
- YearRemodAdd
- Neighborhood
- ExterQual
- KitchenQual
- MasVnrArea

- 派生特徴
- HouseAge
- RemodAge
- TotalSF
- QualitySF

このベースラインの実力と処理内容（1）

スコア：RMSE	評価
30000以上	ほぼベースライン
20000前後	普通
17000台	**良い（理解できている）**
15000台	上位寄り
12000台	かなり良い
0	無視枠

- 実行時間は数分：
- baseline 17203.06322.py
- baseline 17203.06322.csv
- 👉「重要そうな住宅の特徴だけを使って、LightGBMで価格を予測し、RMSEで評価する」コードです。

- 明らかな外れ値を除去
- GrLivArea > 4000 かつ SalePrice < 300000
- 異常に広いのに安い家を削除
- モデルが変な学習をしないようにする
- 👉 回帰では「外れ値対策」が精度に直結。

- 目的変数の変換（重要）
- y = np.log1p(SalePrice)
- 価格を log変換
- 高額物件の影響を弱める
- 👉 本来は上級コンペ向けだが、
  このコードでは 安定化目的 で使っている。

処理内容(2)

- 使う特徴量を厳選
- OverallQual ← 家の質
- GrLivArea ← 居住面積
- TotalBsmtSF ← 地下室面積
- GarageCars ← 車庫台数
- GarageArea
- YearBuilt
- YearRemodAdd
- Neighborhood ← 地域
- ExterQual
- KitchenQual
- MasVnrArea
- 👉
- 「住宅価格に直感的に効くもの」だけ
- 欠損だらけ・意味が薄い列は使わない

- 派生特徴量を作る
- HouseAge = 築年数
- RemodAge = 改築後年数
- TotalSF = 総面積
- QualitySF = 品質 × 面積
- 👉
  元データをそのまま使うより、意味を強める
  (これがスコア改善の核心)

- 欠損値を補完
- 面積・台数 → 0
- カテゴリ → "None"
- 「無いものは 0 / None」として扱う
- 入門〜中級で最も安全な方法

処理内容(3)

- カテゴリ変数を LightGBM 用に変換
- astype("category")
- Neighborhood などを
  LightGBM が理解できる型にする
- これをしないとエラーになる環境がある

- LightGBM で学習
- ・5分割CV
- ・決定木ベースの回帰
- ・early stopping あり
- LightGBM は:
- 数値＋カテゴリに強い
- 少ない前処理で高精度
- 👉「入門〜実戦の橋渡し」的モデル。

- CV RMSE を計算
  - RMSE(log価格)
  - モデルがどれくらい当たっているか確認
  - ここで出る値と Kaggle の表示は別物
- submission.csv を作る
  - SalePrice = expm1(pred)
  - log を元に戻して
  - Kaggle 提出形式にする
- まとめ(超重要)
- このコードは何者？
  - ❌ 超入門ではない
  - ⭕ 入門〜中級へのステップアップ用
  - ⭕「ちゃんと機械学習している」コード
  - なぜ 17203 になった？
  - この入門コンペは RMSE(ドル)
  - 17000 台は 普通に良い値

Kaggle環境用で、12000より上のスコアを狙ったコードを、詳細説明コメント付きで作成して

- ポイントは：
  - 目的変数はそのまま（SalePriceをlogしない）
    ※スコアが「1万台」なので、このコンペは 価格（ドル）空間RMSE。logにすると逆にズレやすいです。
  - カテゴリも全部使う（One-Hot）
  - LightGBM（LGBMRegressor）でGBDTを回す
  - KFoldで平均化してブレを減らす

実行時間は数分：
baseline2 14638.60023.py
baseline2 14638.60023.csv
17203→14638と改善したが、12000に届かない。

CatBoost版（コピペ1セル・詳細コメント付き）

- 12000に届かない“本当の理由”
- ① One-Hot + GBDT の限界
- One-Hot は
  - Neighborhood
  - Exterior / Kitchen 品質
    などを **“独立したフラグ”**として扱う
- カテゴリ同士の関係性（順序・距離）を学べない
- 👉 入門コンペ上位（12000前後）は
  「カテゴリをそのまま賢く扱えるモデル」を使っていることが多い。

実行時間は15分：
catboost 13291.78471.py
catboost 13291.78471.csv

いいところまで来ています。**13291** ははっきり言って「正攻法の CatBoost 単体としては上位寄り」です。
ここから **12000 を切るかどうか**は、**モデルを変えるというより“詰め”の段階**になります。
以下、**なぜ 13291 で止まっているか → 何をやれば次に下がるか**を順に説明します。

catboost + lgbm

- ここから 12000 に近づく
- ブレンド(平均)
- final_pred = 0.5 * catboost + 0.5 * lightgbm
- これだけで
  👉 500～1200 下がることが珍しくない
- あなたの場合:
- CatBoost ≈ 13290
- LightGBM ≈ 14600
- 平均 → 12500～13000 台前半 が現実的

実行時間は20分:
catboost_lgbm 13595.01750.py
catboost_lgbm 13595.01750.csv
13291 → 13595 悪化

悪化した原因

- 結論（重要）
  - 13595 に悪化したのは「ブレンドが失敗した」のではなく、この入門コンペでは「そのブレンドのやり方が噛み合っていない」だけです。
- なぜブレンドで悪化したのか（核心）
  - ① CatBoost と LightGBM の「誤差の方向」が似すぎている
  - ブレンドが効く条件は：
  - A が間違えるところを B が当てる
  - B が間違えるところを A が当てる
  - ですが、あなたの結果を見ると：

モデル	Public
LightGBM	~14638
CatBoost	**13291**
ブレンド(0.5/0.5)	**13595**

- 👉 LightGBM が CatBoost の足を引っ張っている
- つまり：
- CatBoost の方が明確に強い
- LightGBM は「同じ所を、より大きく間違えている」
- この状態で 0.5 / 0.5 平均を取ると、
  良い予測を悪い予測で薄めてしまう。

- ここで「12000を切るため」に取るべき戦略
- ❌ やらなくていいこと
- LightGBM をこれ以上いじる
- ブレンド比率を 0.6 / 0.4 で総当り
- UMAP / PCA / 複雑なEDA
- 👉 労力の割にほぼ伸びない

CatBoost 単体をさらに“価格分布特化”させる

- 入門コンペで **12000未満を出している人の多くは「CatBoost単体」**です。
- 具体的に効くのは：
- (A) loss_function を RMSE から MAE寄りにする
- loss_function="MAE"
- eval_metric="RMSE"
- 学習は「外れ値に引っ張られにくい」
- 評価は RMSE
  👉 RMSEが下がることがある（重要）

- (B) depth を 8 → 10
- CatBoost は深さが効きやすい。
- depth=10

- (C) iterations を増やして early stopping に任せる
- iterations=50000
- learning_rate=0.02
- od_wait=500

実行時間は30分：
catboost_mae 15224.79022.py
catboost_mae 15224.79022.csv
13291 → 13595 → 15224
どんどん悪化

結論：
A+B+C を全部入れたのが逆効果になりました
•特に A（MAE学習）がこのコンペと噛み合っていません
なぜ MAE 学習（A）がダメだったのか
このコンペの評価は？
RMSE（平方誤差）
高額物件の誤差が **二乗で強く罰せられる**
MAE 学習の性質
外れ値を「気にしない」
中央値寄りの予測になりやすい
高額物件を意図的に外しやすい
👉 結果：
高価格帯の誤差が増える
RMSE が一気に悪化 → 15224

「まだできること」があるとすれば

- ① RMSE学習に戻す（最重要）
- loss_function="RMSE"
- eval_metric="RMSE"
- ② depth=9 or 10 を軽く試す
- 8 → 9 → 10
- MAEは使わない
- ③ Neighborhood の "平均価格" を CV 内でエンコード
- これは 唯一 12000に近づく可能性がある手ですが、やや上級です。

実行時間は1時間40分：
catboost_depth=7-10 13135.56373.py
catboost_depth=7-10 13135.56373.csv
13291 → 13595 → 15224→13135
depthは7が一番良かった。

[depth=7] CV RMSE(price): 23186.08
[depth=8] CV RMSE(price): 23898.19
[depth=9] CV RMSE(price): 24820.55
[depth=10] CV RMSE(price): 24696.64

✅ Best depth: 7 / CV RMSE(price): 23186.08

次はdepth=6をトライ

catboost depth=6

- 実行時間：5分
- catboost_depth=6 13118.69387.py
- catboost_depth=6 13118.69387.csv

- 今のモデルの立ち位置（重要）
- この入門コンペで、
- リークなし
- 正攻法（CatBoost + log + 定番FE）
- の場合の 現実的な上限帯は：
- 13000 ± 200
- あなたの 13118 は
  👉 ほぼ天井ゾーン に入っています。
- これ以上は
- 改善しても 数十〜100程度
- 乱数・分割運で上下する世界

- ここから「まだやる価値がある」最小手
- ① l2_leaf_reg を軽く振る（最後の詰め）
- depth=6 にしたことで、
  正則化の最適点も少し変わります。
- 試す価値があるのはこの2つだけ：
- l2_leaf_reg = 2.0
- l2_leaf_reg = 5.0
- 他は触らない。
  → ±50〜100 動く可能性あり

- ② learning_rate をほんの少し下げる（時間許せば）
- learning_rate = 0.025
- iterations = 40000
- od_wait = 500
- depth=6 はやや学習が速い
- lr を下げると「詰め」が効くことがある
- ※ 実行時間は増えるので 余力があるときだけ

- やらなくていいこと（断言）
- depth=5（表現力が落ちすぎる）
- MAE / Huber
- ブレンド再挑戦
- UMAP / PCA / 複雑なEDA
- 👉 コスパが合わない

- 正直な評価（本音）
- ここまでで
  モデル選択・変換・正則化の勘所は完全に身についています
- この入門コンペでこれ以上粘るより、
  本家 House Prices に行った方が学習効率は圧倒的に高い

履歴

版（ファイル名）	Public（あなたの結果）	モデル/方針	前処理（カテゴリ/欠損）	特徴量	いちばん大きい違い（ひとことで）
baseline 17203.06322.py	17203	LightGBM（lgb.train）＋重要特徴だけ＋log1p	カテゴリ3列だけ category、欠損は0/Noneで手埋め	手選別 11列＋少しFE	「少数強特徴」縛りの軽量版（情報を捨ててる）
baseline2 14638.60023.py	14638	LightGBM（LGBMRegressor）＋全列	数値=中央値、カテゴリ=最頻、One-Hot（unknown ignore）	全列＋One-Hot	情報を全部使う正攻法LGBM（堅牢なパイプライン）
catboost 13291.78471.py	13291	CatBoost(RMSE)＋log1p＋5fold	object→Missing、数値→中央値（CatBoostはカテゴリを直接扱う）	全列＋軽FE	CatBoostに乗り換えたのが効いて一気に改善
catboost_lgbm 13595.01750.py	13595	CatBoost＋LGBMを0.5/0.5平均ブレンド	CatBoostは直カテゴリ、LGBMはOne-Hot（train+testで列固定）	両方とも同じ軽FE	「良い予測を悪い予測で薄めた」形になり悪化
catboost_mae 15224.79022.py	15224	CatBoost(MAEで学習, eval=RMSE)＋depth10＋log1p	Missing/中央値補完	全列＋軽FE	学習ロスをMAEにしたのがRMSE評価と噛み合わず崩壊
catboost_depth=7-10 13135.56373.py	13135	CatBoost(RMSE)＋log1p＋depth探索（7/8/9/10）	Missing/中央値補完	全列＋軽FE	「depthを選ぶ」段階に入って、13135まで詰めた
catboost_depth=6 13118.69387.py	13118	CatBoost(RMSE)＋log1p＋depth=6最終調整	Missing/中央値補完	全列＋軽FE	深さを1つ浅くして過学習を少し抑え、さらに微改善

履歴（主要パラメータ版）

版（ファイル名）	Public RMSE	モデル	Target変換	depth / 木	学習率	iterations	その他の重要設定	処理内容の要点
baseline 17203	17203	LightGBM	log1p	max_depth制限	0.05前後	数百	重要特徴のみ	**少数強特徴のみ使用。情報量不足**
baseline2 14638	14638	LightGBM	log1p	制限なし	0.03	3000	One-Hot + KFold	**全特徴 + One-Hotで一気に改善**
catboost 13291	13291	CatBoost	log1p	depth=8	0.03	20000	RMSE学習・早期終了	**CatBoostがカテゴリを直接扱い大幅改善**
catboost_lgbm 13595	13595	CB + LGBM	log1p	CB:8 / LGBM:深	0.03 / 0.01	多	単純平均	**弱いLGBMが足を引っ張った**
catboost_mae 15224	15224	CatBoost	log1p	depth=10	0.02	50000	**loss=MAE**	**評価RMSEと学習MAEが不整合で崩壊**
catboost_depth=7–10	13135	CatBoost	log1p	depth=7最良	0.02	50000	od_wait=600	**depth調整で過学習を抑制**
catboost_depth=6（最良）	**13118**	CatBoost	log1p	**depth=6**	**0.03**	**30000**	od_wait=400	**最終形：表現力と汎化のバランス最適**

付録：UMAPについて

このコンペのEDAでUMAPは意味があるか？

- **この入門コンペ(Home Data for ML Course)での UMAP 可視化は、基本的に「スコア改善にはほぼ意味がない」**です。ただし、学習目的・理解目的としては意味はある、という位置づけ。
- なぜ UMAP が効きにくいのか(本質)
- ① データの性質が「表形式・異種混合」
- このコンペの特徴:
  - 数値(面積・年数・価格)
  - 順序カテゴリ(ExterQual など)
  - 名義カテゴリ(Neighborhood など)
  - 欠損が多い列
- UMAPは本来、
  - 連続値
  - 距離が意味を持つ空間
- に強い。
- 👉 One-Hot後の高次元・疎な表データとは相性が悪い。

- 目的は「可視化」だが、モデルは「木」
- このコンペで強いのは:
  - LightGBM
  - CatBoost
  - RandomForest
- これらは👇
  - 距離構造を使わない
  - クラスタ分離を前提にしない
  - 軸方向の分割で学習する
- 👉 UMAPで"綺麗に分離して見える"ことと、木モデルの性能はほぼ無関係。

このコンペのEDAでUMAPは意味があるか？

- 「見える＝使える特徴」ではない
- 仮にUMAPで：
  - 高価格クラスタ
  - 低価格クラスタ
- が見えても、
  - それを数値特徴として入れると
  - 情報が圧縮されすぎて
  - むしろ性能が落ちることが多い
- 👉 UMAP特徴量は、この種の回帰ではほぼ効かない。

- 代わりに「スコアに直結するEDA」
- このコンペで UMAPより100倍効くEDA は：
- 1️⃣ SalePrice との単変量関係
  - OverallQual
  - GrLivArea
  - TotalSF
  - Neighborhood（箱ひげ）
- 2️⃣ 欠損の意味を考える
  - 欠損 =「無い」なのか「不明」なのか
    - Bsmt系 → 無い
    - Alley → 無い
    - PoolQC → ほぼ無い
  - 👉 欠損を0や"None"にするだけで効く列が多い。
- 3️⃣ 派生特徴のヒント探し
  - 面積の合算
  - 年数差分
  - 品質 × 面積
  - 👉 これはUMAPより直接スコアに効く。