In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import random
import seaborn as sns
import lightgbm as lgb

from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression
from pathlib import Path
from matplotlib.gridspec import GridSpec
from scipy.signal import savgol_filter
from sklearn.model_selection import KFold
```

In [12]:
```
train = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")

train["facility_rating"] = train["facility_rating"].astype("category")
test["facility_rating"]  = test["facility_rating"].astype("category")
```

In [13]:
```
TARGET = "exam_score"
ID_COL = "id"

X = train.drop(columns=[TARGET])
y = train[TARGET]

X_test = test.copy()
```

In [14]:
```
categorical_features = [
    "gender",
    "course",
    "internet_access",
    "sleep_quality",
    "study_method",
    "exam_difficulty"
]

categorical_features.append("facility_rating")
```

In [15]:
```
for col in categorical_features:
    X[col] = X[col].astype("category")
    X_test[col] = X_test[col].astype("category")
```

In [16]:
```
params = {
    "objective": "regression",
    "metric": "rmse",
    "learning_rate": 0.05,
    "num_leaves": 64,
    "max_depth": -1,
    "feature_fraction": 0.8,
    "bagging_fraction": 0.8,
    "bagging_freq": 5,
    "verbosity": -1,
    "seed": 42
}
```

In [18]:
```python
N_SPLITS = 5
kf = KFold(n_splits=N_SPLITS, shuffle=True, random_state=42)

oof_preds = np.zeros(len(X))
test_preds = np.zeros(len(X_test))

for fold, (train_idx, val_idx) in enumerate(kf.split(X)):
    print(f"\nFold {fold + 1}")

    X_train, X_val = X.iloc[train_idx], X.iloc[val_idx]
    y_train, y_val = y.iloc[train_idx], y.iloc[val_idx]

    train_set = lgb.Dataset(
        X_train,
        label=y_train,
        categorical_feature=categorical_features
    )

    val_set = lgb.Dataset(
        X_val,
        label=y_val,
        categorical_feature=categorical_features,
        reference=train_set
    )

    model = lgb.train(
        params,
        train_set,
        num_boost_round=5000,
        valid_sets=[train_set, val_set],
        valid_names=["train", "valid"],
        callbacks=[
            lgb.early_stopping(stopping_rounds=100),
            lgb.log_evaluation(period=200)  # prints every 200 rounds
        ]
    )

    oof_preds[val_idx] = model.predict(X_val, num_iteration=model.best_it
    test_preds += model.predict(X_test, num_iteration=model.best_iteratio
Fold 1
Training until validation scores don't improve for 100 rounds
[200]	train's rmse: 8.72141	valid's rmse: 8.77623
[400]	train's rmse: 8.64141	valid's rmse: 8.7619
[600]	train's rmse: 8.57422	valid's rmse: 8.75629
[800]	train's rmse: 8.51417	valid's rmse: 8.75387
Early stopping, best iteration is:
[879]	train's rmse: 8.49107	valid's rmse: 8.75361

Fold 2
Training until validation scores don't improve for 100 rounds
[200]	train's rmse: 8.71731	valid's rmse: 8.78286
[400]	train's rmse: 8.64292	valid's rmse: 8.77209
[600]	train's rmse: 8.5753	valid's rmse: 8.76624
[800]	train's rmse: 8.51584	valid's rmse: 8.76513
[1000]	train's rmse: 8.4569	valid's rmse: 8.76374
Early stopping, best iteration is:
[1045]	train's rmse: 8.44418	valid's rmse: 8.76351

Fold 3
Training until validation scores don't improve for 100 rounds
[200]	train's rmse: 8.72069	valid's rmse: 8.77869
[400]	train's rmse: 8.64165	valid's rmse: 8.76556
[600]	train's rmse: 8.57492	valid's rmse: 8.76181
[800]	train's rmse: 8.5145	valid's rmse: 8.75997
Early stopping, best iteration is:
[752]	train's rmse: 8.52853	valid's rmse: 8.75921

Fold 4
Training until validation scores don't improve for 100 rounds
[200]	train's rmse: 8.71679	valid's rmse: 8.79528
[400]	train's rmse: 8.64058	valid's rmse: 8.78272
[600]	train's rmse: 8.57318	valid's rmse: 8.77952
[800]	train's rmse: 8.5136	valid's rmse: 8.77754
[1000]	train's rmse: 8.45563	valid's rmse: 8.77573
Early stopping, best iteration is:
[934]	train's rmse: 8.47361	valid's rmse: 8.7752

Fold 5
Training until validation scores don't improve for 100 rounds
[200]	train's rmse: 8.70989	valid's rmse: 8.81125
[400]	train's rmse: 8.63354	valid's rmse: 8.79974
[600]	train's rmse: 8.56828	valid's rmse: 8.79569
[800]	train's rmse: 8.50975	valid's rmse: 8.79474
Early stopping, best iteration is:
[894]	train's rmse: 8.48245	valid's rmse: 8.79449
```

In [20]:
```python
rmse = mean_squared_error(y, oof_preds) ** 0.5
print(f"\nOverall CV RMSE: {rmse:.5f}")
```

```
Overall CV RMSE: 8.76922
```

In [21]:
```python
unknown_submission = pd.DataFrame({
    "id": test[ID_COL],
    "exam_score": test_preds
})

unknown_submission.to_csv("unknown_submission.csv", index=False)
unknown_submission.head
```

Out[21]:
```
<bound method NDFrame.head of             id  exam_score
0       630000   72.533020
1       630001   68.954896
2       630002   88.350431
3       630003   58.161802
4       630004   47.101893
...        ...         ...
269995  899995   60.953933
269996  899996   40.744867
269997  899997   89.962017
269998  899998   54.030336
269999  899999   67.196918

[270000 rows x 2 columns]>
```

In [ ]: