

Interpolation Tables

`mlumina.interpolation_tables`

Пакет объединяет две независимые темы под одним именем (совпадение по расположению в дереве кода, не по смыслу):

1. `TableInterpolator` — 2D-интерполяция табличных данных.
2. `models_evaluation.py` — общие утилиты сравнения/подбора sklearn-совместимых моделей регрессии (не связаны с интерполяцией).

`TableInterpolator`

```
from mlumina.interpolation_tables import TableInterpolator

interp = TableInterpolator(df, x_col="x", y_col="y", z_col="z")

value = interp.get(4723, 937)
Xg, Yg, Zg = interp.get_grid(step_x=50, step_y=50)
interp.plot_surface(step_x=50, step_y=50)
```

`TableInterpolator(df, x_col, y_col, z_col)`:

- сортирует `df` по `x_col/y_col`, строит регулярную сетку через `df.pivot(index=x_col, columns=y_col, values=z_col)`;
- пропуски в сетке (`NaN`) заполняются нулями — если исходная таблица неполная/нерегулярная, это может незаметно исказить интерполяцию;
- под капотом — `scipy.interpolate.RegularGridInterpolator` с `method="cubic"`.

Методы:

- `get(x_val, y_val) -> float` — значение Z в точке.
- `get_grid(step_x=100, step_y=100) -> (Xg, Yg, Zg)` — регулярная сетка с заданным шагом (в единицах X/Y, а не количестве точек).
- `plot_surface(step_x=100, step_y=100)` — 3D-график (matplotlib): scatter исходных точек + поверхность.

models_evaluation.py (

Общие sklearn-совместимые (`.fit`/`.predict`) утилиты для регрессии, ничего специфичного для интерполяции.

```
from mlumina.interpolation_tables import (
    evaluate_model,
    grid_search_model,
    compare_multiple_models,
    compare_multiple_models_ray,
    hyperopt_search_model,
)

from mlumina.interpolation_tables.models_evaluation import compute_regression_metrics,
expand_param_grid
```

(`compute_regression_metrics`, `expand_param_grid` и `param_grid_to_hyperopt_space` не реэкспортируются в `mlumina.interpolation_tables.__init__` — импортировать напрямую из `mlumina.interpolation_tables.models_evaluation`.)

- `compute_regression_metrics(y_true, y_pred) -> dict` — `mse`, `rmse`, `mae`, `r2`.
- `evaluate_model(model, X_train, y_train, X_test, y_test, metrics=None) -> (model, metrics)` — обучает модель и считает метрики; `metrics` — `None` (все), строка (одно число) или список строк (словарь).
- `expand_param_grid(param_grid)` — превращает `{"param": (start, end, step)}` или `[...]` в явный список значений для перебора (используется внутри `grid_search_model`).
- `grid_search_model(model_class, param_grid, X_train, y_train, X_test, y_test, metrics=None, optimize_metric="mse") -> (best_model, best_params, best_metrics)` — полный перебор сетки для одного класса модели.
- `compare_multiple_models(model_list, X_train, y_train, X_test, y_test, metrics=None, optimize_metric="mse") -> (best_model, best_params, best_metrics, best_model_class)` — `model_list` это `[(model_class, param_grid), ...]`; для каждого класса выполняется `grid_search_model`, выбирается лучший по `optimize_metric` (меньше — лучше).
- `compare_multiple_models_ray(...)` — то же самое, но каждый `(model_class, param_grid)` обрабатывается в отдельной Ray-задаче (`ray.put` для данных + `@ray.remote` `_ray_grid_search_task`). Требуется `ray.init()` заранее, иначе `RuntimeError`.
- `param_grid_to_hyperopt_space(param_grid) / hyperopt_search_model(...)` — тот же перебор, но через `hyperopt` (TPE) вместо полного grid search.

```
import ray

from mlumina.interpolation_tables import compare_multiple_models_ray
```

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import Ridge

ray.init()

model, params, metrics, model_class = compare_multiple_models_ray(
    model_list=[
        (DecisionTreeRegressor, {"max_depth": (2, 10, 1)}),
        (Ridge, {"alpha": [0.1, 1.0, 10.0]}),
    ],
    X_train=X_train, y_train=y_train, X_test=X_test, y_test=y_test,
    optimize_metric="rmse",
)
```

В корневом README упоминается функция `evaluate_models_parallel` и класс `InterpolationTable` — таких имён в коде нет, см. [known-issues.md](#).

Revision #1

Created 2026-09-23 06:40:07 UTC by Маслов Алексей

Updated 2026-09-23 06:40:08 UTC by Маслов Алексей