Подготовка данных

```
import pandas as pd
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.ensemble import StackingRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_absolute_error, r2_score
import gmdh

# Загрузка данных
california_housing = fetch_california_housing()
X = california_housing.data
y = california_housing.target

# Преобразование в DataFrame для удобства работы
df = pd.DataFrame(data=X, columns=california_housing.feature_names)
df['target'] = y

# Удаление или заполнение пропусков
imp = SimpleImputer(strategy="mean")
X_imputed = imp.fit_transform(X)

# Разделение на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X_imputed, y,
test_size=0.3, random_state=42)

# Стандартизация признаков
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Обучение моделей

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

# Инициализация StackingRegressor
stacking = StackingRegressor(
    estimators=[
        ('lr', LinearRegression()),
        ('dt', DecisionTreeRegressor())
    ],
    final_estimator=LinearRegression()
)
```

```
# Обучение модели
stacking.fit(X_train_scaled, y_train)
```

```
StackingRegressor(estimators=[('lr', LinearRegression()),
                              ('dt', DecisionTreeRegressor())],
                  final_estimator=LinearRegression())
```

```
# Инициализация MLPRegressor
mlp = MLPRegressor(max_iter=500, random_state=42)

# Обучение модели
mlp.fit(X_train_scaled, y_train)
```

```
MLPRegressor(max_iter=500, random_state=42)
```

```
combi_model = gmdh.Combi()
combi_model.fit(X_train_scaled, y_train, verbose=1, n_jobs=-1,
test_size=0.24, limit=0,

criterion=gmdh.Criterion(gmdh.CriterionType.REGULARITY))
```

```
LEVEL 1  [>                          ] 0% [00m:00s] (8 combinations)

LEVEL 1  [==========================] 100% [00m:00s] (8 combinations)
error=2453.56326
LEVEL 2  [==========================] 100% [00m:00s] (28 combinations)
error=2295.770403
LEVEL 3  [==========================] 100% [00m:00s] (56 combinations)
error=1973.872907
LEVEL 4  [==========================] 100% [00m:00s] (70 combinations)
error=1932.572114
LEVEL 5  [==========================] 100% [00m:00s] (56 combinations)
error=1892.182281
LEVEL 6  [==========================] 100% [00m:00s] (28 combinations)
error=1870.162867
LEVEL 7  [==========================] 100% [00m:00s] (8 combinations)
error=1871.662831

<gmdh.gmdh.Combi at 0x21a4d76d0f0>
```

```
mia_model = gmdh.Mia()
mia_model.fit(X_train_scaled, y_train, verbose=1, n_jobs=-1,
test_size=0.45, limit=0, k_best=9,

criterion=gmdh.Criterion(gmdh.CriterionType.SYM_REGULARITY),
                    polynomial_type=gmdh.PolynomialType.LINEAR)
```

```
LEVEL 1  [==========================] 100% [00m:00s] (28 combinations)
error=9500.862384
LEVEL 2  [==========================] 100% [00m:00s] (36 combinations)
error=7942.429512
LEVEL 3  [==========================] 100% [00m:00s] (36 combinations)
```

```
error=7782.312763
LEVEL 4  [========================] 100% [00m:00s] (36 combinations)
error=7768.706041
LEVEL 5  [========================] 100% [00m:00s] (36 combinations)
error=7751.514395
LEVEL 6  [========================] 100% [00m:00s] (36 combinations)
error=7747.500749
LEVEL 7  [========================] 100% [00m:00s] (36 combinations)
error=7744.418602
LEVEL 8  [========================] 100% [00m:00s] (36 combinations)
error=7743.133109
LEVEL 9  [========================] 100% [00m:00s] (36 combinations)
error=7742.478564
LEVEL 10 [========================] 100% [00m:00s] (36 combinations)
error=7741.742821
LEVEL 11 [========================] 100% [00m:00s] (36 combinations)
error=7741.659116
LEVEL 12 [========================] 100% [00m:00s] (36 combinations)
error=7741.738835

<gmdh.gmdh.Mia at 0x21a4d76ee60>
```

Оценка моделей

```
# Предсказания
stacking_preds = stacking.predict(X_test_scaled)
mlp_preds = mlp.predict(X_test_scaled)
combi = combi_model.predict(X_test_scaled)
mia = mia_model.predict(X_test_scaled)

# Метрики оценки
print("Stacking MAE:", mean_absolute_error(y_test, stacking_preds))
print("Stacking R^2:", r2_score(y_test, stacking_preds))

print("Combi MAE:", mean_absolute_error(y_test, combi))
print("Combi R^2:", r2_score(y_test, stacking_preds))

print("Mia MAE:", mean_absolute_error(y_test, mia))
print("Mia R^2:", r2_score(y_test, stacking_preds))

print("MLP MAE:", mean_absolute_error(y_test, mlp_preds))
print("MLP R^2:", r2_score(y_test, mlp_preds))


Stacking MAE: 0.43292327852795115
Stacking R^2: 0.7055286469526402
Combi MAE: 0.528009891504426
Combi R^2: 0.7055286469526402
Mia MAE: 0.5307090263377373
Mia R^2: 0.7055286469526402
```

```
MLP MAE: 0.3732695682052936
MLP R^2: 0.7792734258554929
```

Вывод

MLP (Multilayer Perceptron) является лучшей моделью среди представленных по обоим метрикам. Она демонстрирует наименьшее среднее количество ошибок (MAE) и наибольшую объяснительную способность (R^2).