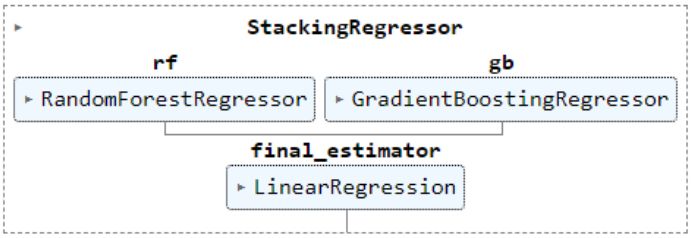
Средний доход HouseAge: Возраст дома AveRooms: Среднее количество комнат AveBedrms: Среднее количество спальных мест Population: Население AveOccup: Средняя занятость Latitude: Широта Longitude: Долгота

```
[ ] import pandas as pd
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import StandardScaler
    from sklearn.datasets import fetch_california_housing
    from sklearn.metrics import mean_squared_error

    # Загрузка датасета
    california = fetch_california_housing()
    X = pd.DataFrame(california.data, columns=california.feature_names)
    y = pd.Series(california.target, name='target')

    # Масштабирование признаков
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)
    # Разделение на обучающую и тестовую выборки
    X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
    print(X.columns)
    print(X)
```

```
Index(['MedInc', 'HouseAge', 'AveRooms', 'AveBedrms', 'Population', 'AveOccup',
       'Latitude', 'Longitude'],
      dtype='object')
       MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  Latitude  \
0      8.3252      41.0  6.984127   1.023810       322.0  2.555556     37.88
1      8.3014      21.0  6.238137   0.971880      2401.0  2.109842     37.86
2      7.2574      52.0  8.288136   1.073446       496.0  2.802260     37.85
3      5.6431      52.0  5.817352   1.073059       558.0  2.547945     37.85
4      3.8462      52.0  6.281853   1.081081       565.0  2.181467     37.85
...       ...       ...       ...        ...         ...       ...       ...
20635  1.5603      25.0  5.045455   1.133333       845.0  2.560606     39.48
20636  2.5568      18.0  6.114035   1.315789       356.0  3.122807     39.49
20637  1.7000      17.0  5.205543   1.120092      1007.0  2.325635     39.43
20638  1.8672      18.0  5.329513   1.171920       741.0  2.123209     39.43
20639  2.3886      16.0  5.254717   1.162264      1387.0  2.616981     39.37

       Longitude
0        -122.23
1        -122.22
2        -122.24
3        -122.25
4        -122.25
...          ...
20635    -121.09
20636    -121.21
20637    -121.22
20638    -121.32
20639    -121.24
```

```python
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor, StackingRegressor
from sklearn.linear_model import LinearRegression

# Создание базовых моделей для стекинга
estimators = [
    ('rf', RandomForestRegressor(n_estimators=100, random_state=42)),
    ('gb', GradientBoostingRegressor(n_estimators=100, random_state=42))
]

# Создание модели стекинга
stacking_model = StackingRegressor(
    estimators=estimators,
    final_estimator=LinearRegression()
)

# Обучение модели стекинга
stacking_model.fit(X_train, y_train)
```




```python
from sklearn.neural_network import MLPRegressor

# Создание и обучение модели многослойного персептрона
mlp_model = MLPRegressor(hidden_layer_sizes=(100,), max_iter=300, random_state=42)
mlp_model.fit(X_train, y_train)
```

```
MLPRegressor(max_iter=300, random_state=42)
```

```python
from gmdh import Combi,Mia

# Создание и обучение модели COMBI & MIA
combi_model = Combi()
mia_model = Mia()

combi_model.fit(X_train, y_train)
y_pred_combi = combi_model.predict(X_test)


mia_model.fit(X_train, y_train)
y_pred_mia = mia_model.predict(X_test)

print('y_pred_combi: ', y_pred_combi)
print('y_pred_mia: ', y_pred_mia)
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `
  and should_run_async(code)
y_pred_combi:  [0.71008084 1.75653192 2.72061734 ... 4.49580402 1.17270903 2.01798205]
y_pred_mia:  [1.06333843 1.47688664 2.29282534 ... 4.4127553  1.65918844 1.8076504 ]
```

```python
# Предсказания на тестовой выборке
y_pred_stacking = stacking_model.predict(X_test)
y_pred_mlp = mlp_model.predict(X_test)
y_pred_combi = combi_model.predict(X_test)
y_pred_mia = mia_model.predict(X_test)

# Оценка качества моделей с помощью метрики RMSE
rmse_stacking = mean_squared_error(y_test, y_pred_stacking, squared=False)
rmse_mlp = mean_squared_error(y_test, y_pred_mlp, squared=False)
rmse_combi = mean_squared_error(y_test, y_pred_combi, squared=False)
rmse_mia = mean_squared_error(y_test, y_pred_mia, squared=False)

# Вывод результатов
print(f'Staking RMSE: {rmse_stacking:.4f}')
print(f'MLP RMSE: {rmse_mlp:.4f}')
print(f'COMBI RMSE: {rmse_combi:.4f}')
print(f'MIA RMSE: {rmse_mia:.4f}')
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationW
  and should_run_async(code)
Staking RMSE: 0.5036
MLP RMSE: 0.5471
COMBI RMSE: 0.7461
MIA RMSE: 0.7977
```

Stacking RMSE: 0.5036 - Эта модель показывает наилучший результат среди всех, что указывает на высокую точность прогнозирования. Стекинг позволяет объединить преимущества нескольких моделей, улучшая таким образом общую производительность.

MLP RMSE: 0.5471 - Модель многослойного персептрона (MLP) демонстрирует хорошую производительность, хотя немного уступает модели стекинга. MLP хорошо подходит для решения задач классификации и регрессии, особенно когда есть сложные зависимости между входными и выходными переменными.

COMBI RMSE: 0.7461 - Метод COMBI (COMBination) в контексте генеративно-мутационного дифференциального хиллкластеризатора (GMDH) представляет собой линейный подход к комбинированию предсказаний. Значение RMSE говорит о том, что этот метод работает менее эффективно по сравнению с другими моделями, возможно, из-за своей простоты или недостаточной адаптивности к данным.

MIA RMSE: 0.7977 - Метод MIA (Multiple Input Aggregation) также является частью семейства методов GMDH, но он фокусируется на агрегации множественных входных данных. Уровень RMSE указывает на то, что этот метод имеет наибольшую ошибку среди всех рассмотренных, что может говорить о его меньшей способности к точному прогнозированию по сравнению с другими методами.