Отчёт по лабораторной работе №3 по фундаментальным концепциям искусственного интеллекта на тему: “ Оптимизация гиперпараметров”

Выполнил студент группы М80-114СВ-24 Сипкин Владислав.

1. Выполнение оптимизации гиперпараметров нелинейной модели MLPClassifier для датасета make_blobs с семплером TPESampler и прунером HyperbandPruner

а) Этот код включает:

Листинг 1 – Программа реализации оптимизации гиперпараметров нелинейной модели MLPClassifier для датасета make_blobs с семплером TPESampler и прунером HyperbandPruner

```
import logging
import sys

import optuna
from optuna.visualization import plot_contour
from optuna.visualization import plot_intermediate_values
from optuna.visualization import plot_optimization_history
from optuna.visualization import plot_parallel_coordinate
from optuna.visualization import plot_param_importances
from optuna.visualization import plot_slice
from optuna.visualization import plot_edf
from optuna.visualization import plot_rank
from optuna.visualization import plot_timeline

from sklearn.neural_network import MLPClassifier
import sklearn.datasets
import sklearn.model_selection

def objective_classification(trial):
    # генерация данных из датасета
    breast_cancer = sklearn.datasets.load_breast_cancer()
    classes = list(set(breast_cancer.target))
    train_x, valid_x, train_y, valid_y =
sklearn.model_selection.train_test_split(
        breast_cancer.data, breast_cancer.target, test_size=0.25, random_state=42
    )

    # настройка гиперпараметров
    hidden_layer_sizes = trial.suggest_int('hidden_layer_sizes', 50, 200,
log=True)
    solver = trial.suggest_categorical("solver", ["adam", "sgd"])
    alpha = trial.suggest_float("alpha", 1e-5, 1e-1, log=True)

    clf = MLPClassifier(hidden_layer_sizes=(hidden_layer_sizes,), solver=solver,
alpha=alpha, max_iter=1000)
```

```
    for step in range(100):
        clf.partial_fit(train_x, train_y, classes=classes)

        # очет о промежуточном объективном значении
        intermediate_value = 1.0 - clf.score(valid_x, valid_y)
        trial.report(intermediate_value, step)

        if trial.should_prune():
            raise optuna.TrialPruned()

    return 1.0 - clf.score(valid_x, valid_y)

# настройка логирования
optuna.logging.get_logger("optuna").addHandler(logging.StreamHandler(sys.stdout))

# подключение к PostgreSQL для хранения результатов
storage_url = "postgresql://postgres:5555@localhost:5432/optimization_results"

# создание исследования (study)
study_name = "blobs_classification1"
sampler_name = optuna.samplers.TPESampler(seed=42)       # семплер TPESampler
pruner_name = optuna.pruners.HyperbandPruner()  # прунер HyperbandPruner

study = optuna.create_study(
    sampler=sampler_name,
    pruner=pruner_name,
    study_name=study_name,
    storage=storage_url,
    direction="maximize",
    load_if_exists=False
)

# запуск оптимизации
study.optimize(objective_classification, n_trials=50)

# вывод лучших параметров и значения
print("Best params:", study.best_params)
print("Best value:", study.best_value)

# визуализация результатов оптимизации гиперпараметров
plot_optimization_history(study).show()
plot_intermediate_values(study).show()
plot_parallel_coordinate(study, params=["hidden_layer_sizes", "solver",
"alpha"]).show()
plot_contour(study, params=["hidden_layer_sizes", "alpha"]).show()
plot_slice(study, params=["hidden_layer_sizes", "alpha"]).show()
plot_param_importances(study, target=lambda t: t.value,
target_name="error").show()
plot_edf(study).show()
plot_rank(study).show()
```

```
plot_timeline(study).show()
```

б) Вывод результатов работы программы в консоле:

```
Trial 37 pruned.
[I 2024-12-14 21:11:58,155] Trial 38 pruned.
Trial 38 pruned.
[I 2024-12-14 21:11:58,272] Trial 39 pruned.
Trial 39 pruned.
[I 2024-12-14 21:11:58,381] Trial 40 pruned.
Trial 40 pruned.
[I 2024-12-14 21:11:58,741] Trial 41 pruned.
Trial 41 pruned.
[I 2024-12-14 21:11:59,095] Trial 42 pruned.
Trial 42 pruned.
[I 2024-12-14 21:11:59,190] Trial 43 pruned.
Trial 43 pruned.
[I 2024-12-14 21:11:59,300] Trial 44 pruned.
Trial 44 pruned.
[I 2024-12-14 21:11:59,467] Trial 45 pruned.
Trial 45 pruned.
[I 2024-12-14 21:11:59,558] Trial 46 pruned.
Trial 46 pruned.
[I 2024-12-14 21:11:59,718] Trial 47 pruned.
Trial 47 pruned.
[I 2024-12-14 21:11:59,829] Trial 48 pruned.
Trial 48 pruned.
[I 2024-12-14 21:11:59,923] Trial 49 pruned.
Trial 49 pruned.
Best params: {'hidden_layer_sizes': 113, 'solver': 'sgd', 'alpha': 4.809461967501571e-05}
Best value: 0.3776223776223776
```

Рис. 1. Результаты оптимизации гиперпараметров нелинейной модели MLPClassifier для датасета make_blobs с семплером TPESampler и прунером HyperbandPruner

в) Инструкция по запуску реляционной базы данных, хранящая результаты выше приведённого исследования:

1. Запустить PostgreSQL;

2. Подключиться к PostgreSQL с помощью команды:

psql -U postgres

3. Выйти из psql:

\q

3. Создать базу данных с помощью команды:

CREATE DATABASE optimization_results

4. Настроить строку подключения в коде:

storage_url = "postgresql://postgres:5555@localhost:5432/optimization_results"

Изменить в этой строке пароль (5555) и порт (5432) на те, что указаны в вашем PostgreSQL;

5. Создать Python-скрипт с кодом, представленным в Листинг 1, к примеру, в VS Code и запустить его;

6. Затем подключиться к нашей базе данных:

psql -U postgres -d optimization_results

7. Посмотреть созданные таблицы:

\dt

8. Проверить записи:

SELECT * FROM studies;

г) Визуализация графиков:

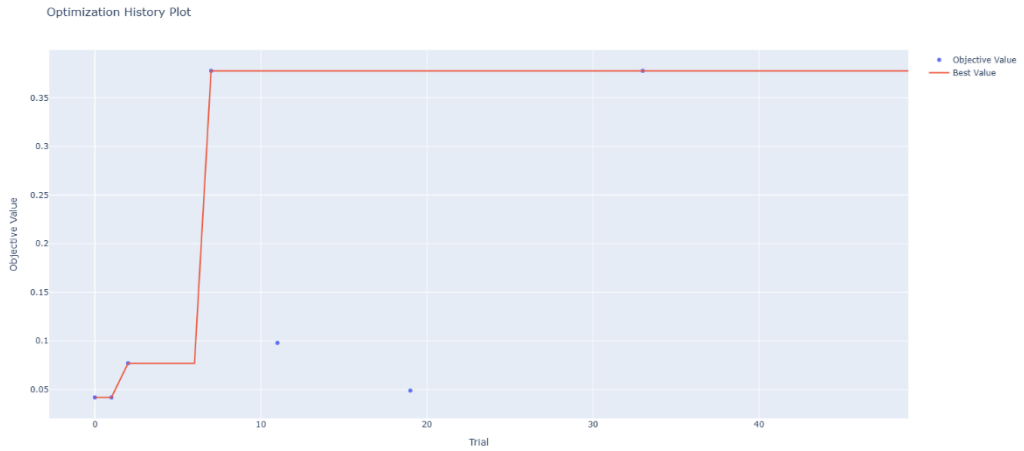

Рис. 2. Визуализация истории оптимизации при семплере TPESampler и прунере HyperbandPruner

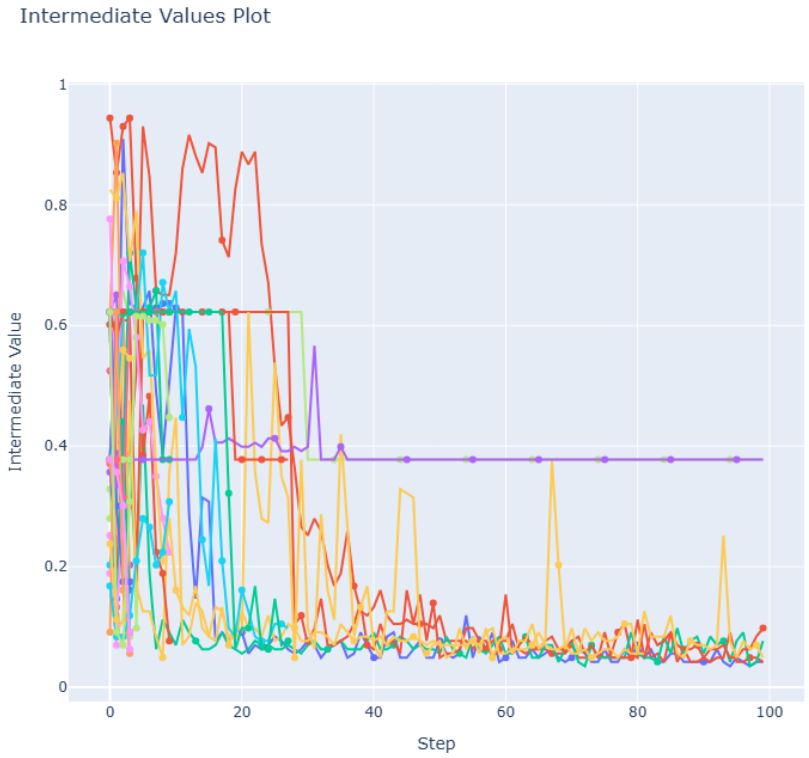

Рис. 3. Визуализация кривых обучения в ходе испытаний при семплере TPESampler и прунере HyperbandPruner

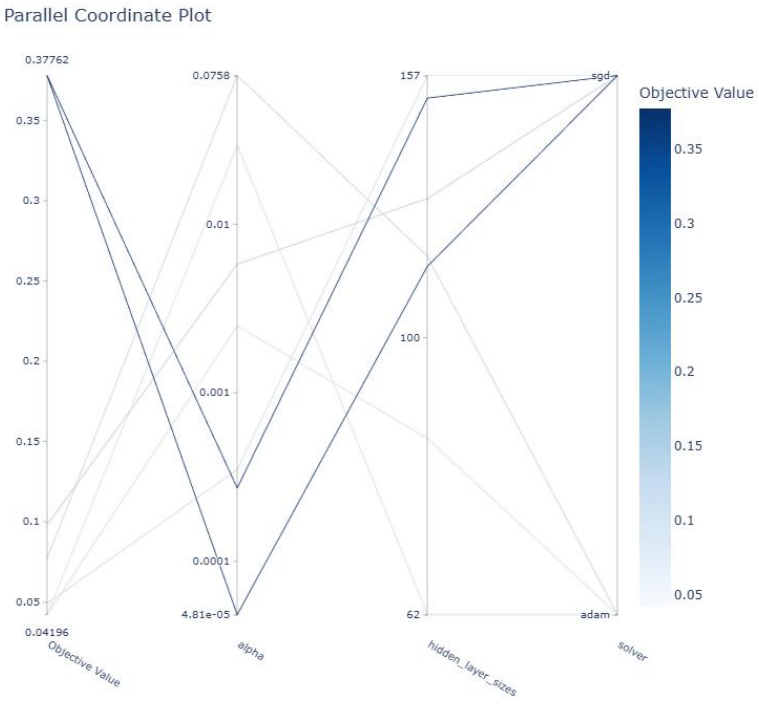

Рис. 4. Визуализация многомерных отношений параметров alpha, hidden_layer_sizes и solver при семплере TPESampler и прунере HyperbandPruner

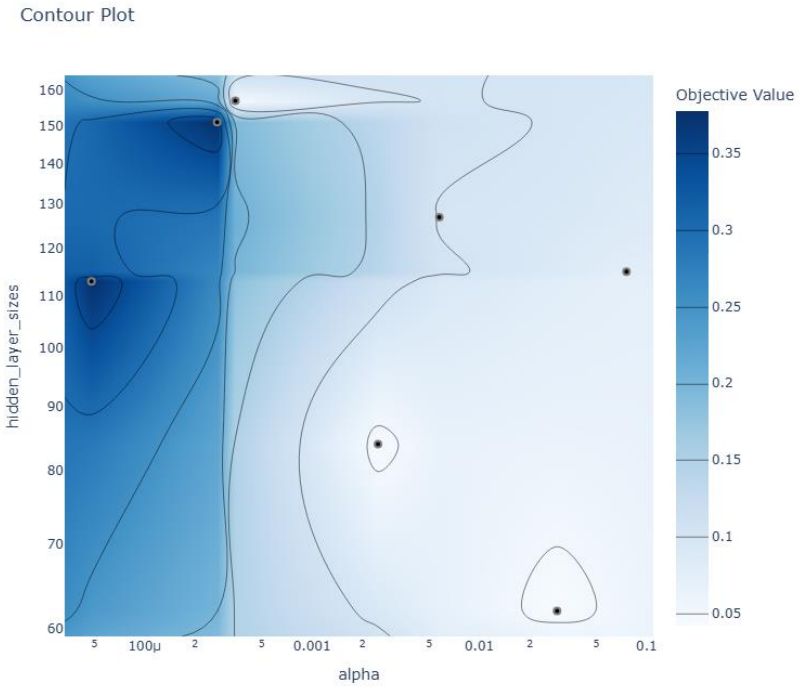

Рис. 5. Визуализация гиперпараметрических зависимостей alpha и hidden_layer_sizes и при семплере TPESampler и прунере HyperbandPruner

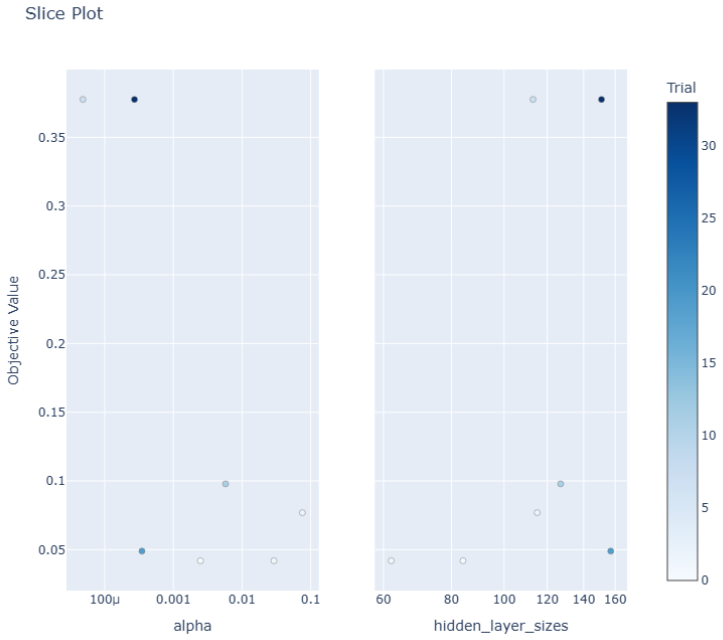

Рис. 6. Визуализация alpha и hidden_layer_sizes в виде графика срезов при семплере TPESampler и прунере HyperbandPruner

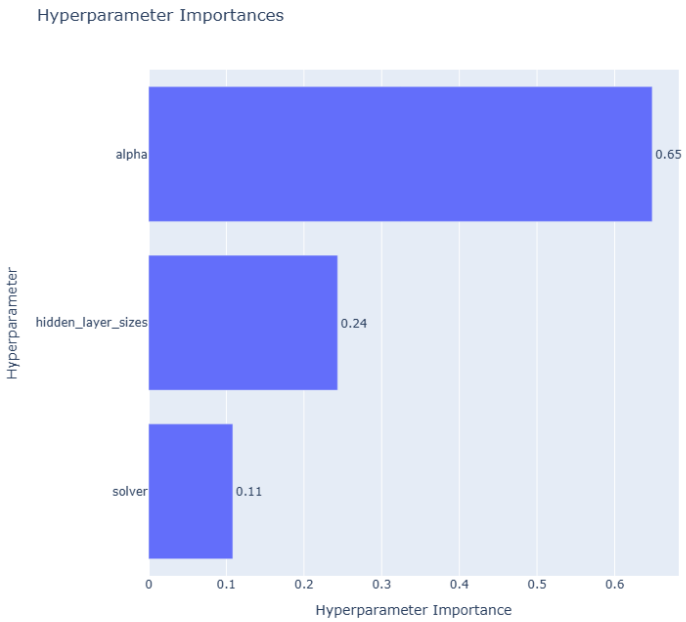

Рис. 7. Визуализация влияния alpha, hidden_layer_sizes и solver на продолжительность исследования при семплере TPESampler и прунере HyperbandPruner

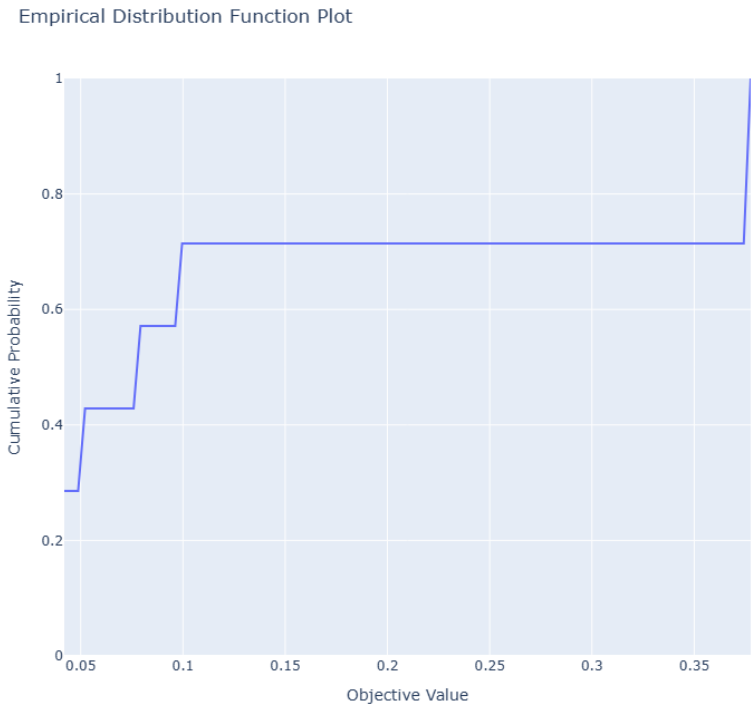

Рис. 8. Визуализация эмпирической функции распределения при семплере TPESampler и прунере HyperbandPruner

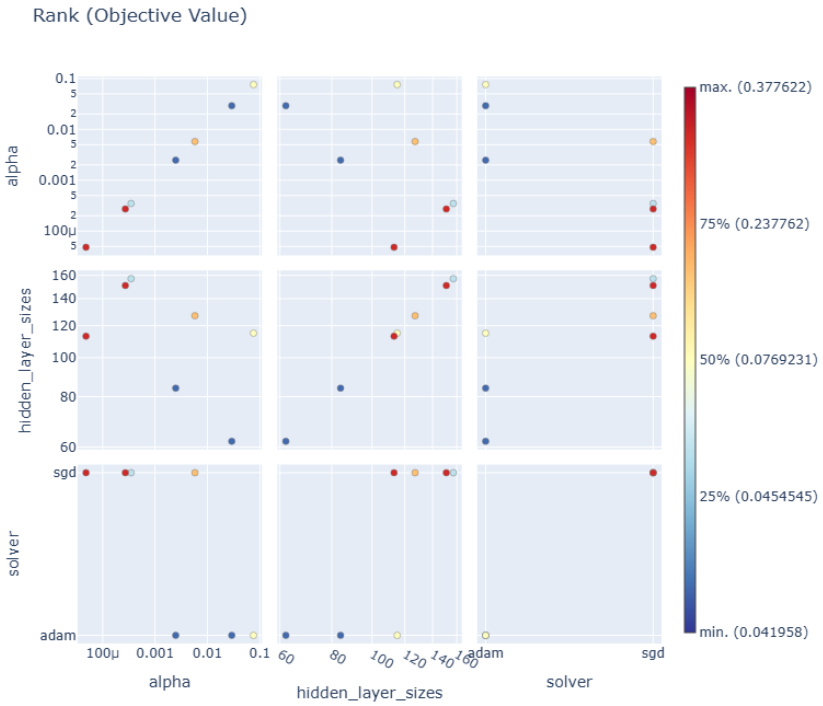

Рис. 9. Визуализация взаимосвязи параметров с помощью диаграмм рассеяния, окрашенных в соответствии с объективными значениями при семплере TPESampler и прунере HyperbandPruner

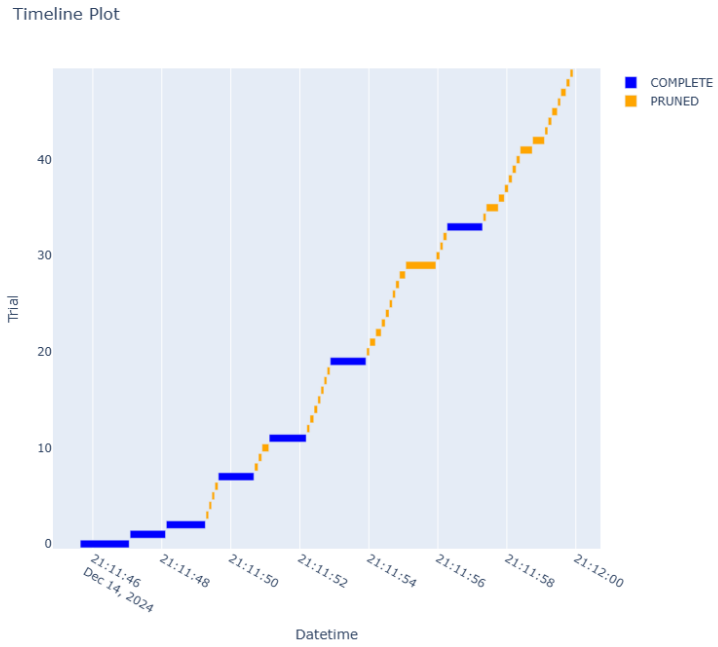

Рис. 10. Визуализация временной шкалы оптимизации выполненных испытаний при семплере TPESampler и прунере HyperbandPruner

2. Выполнение оптимизации гиперпараметров нелинейной модели MLPClassifier для датасета make_blobs с семплером RandomSampler и прунером MedianPruner

а) Этот код включает:

Листинг 2 – Программа реализации оптимизации гиперпараметров нелинейной модели MLPClassifier для датасета make_blobs с семплером RandomSampler и прунером MedianPruner

```
import logging
import sys

import optuna
from optuna.visualization import plot_contour
from optuna.visualization import plot_intermediate_values
from optuna.visualization import plot_optimization_history
from optuna.visualization import plot_parallel_coordinate
from optuna.visualization import plot_param_importances
from optuna.visualization import plot_slice
from optuna.visualization import plot_edf
from optuna.visualization import plot_rank
from optuna.visualization import plot_timeline

from sklearn.neural_network import MLPClassifier
import sklearn.datasets
import sklearn.model_selection

def objective_classification(trial):
    # генерация данных из датасета
    breast_cancer = sklearn.datasets.load_breast_cancer()
    classes = list(set(breast_cancer.target))
    train_x, valid_x, train_y, valid_y =
sklearn.model_selection.train_test_split(
        breast_cancer.data, breast_cancer.target, test_size=0.25, random_state=42
    )

    # настройка гиперпараметров
    hidden_layer_sizes = trial.suggest_int('hidden_layer_sizes', 50, 200,
log=True)
    solver = trial.suggest_categorical("solver", ["adam", "sgd"])
    alpha = trial.suggest_float("alpha", 1e-5, 1e-1, log=True)

    clf = MLPClassifier(hidden_layer_sizes=(hidden_layer_sizes,), solver=solver,
alpha=alpha, max_iter=1000)

    for step in range(100):
        clf.partial_fit(train_x, train_y, classes=classes)

        # очет о промежуточном объективном значении
```

```
        intermediate_value = 1.0 - clf.score(valid_x, valid_y)
        trial.report(intermediate_value, step)

        if trial.should_prune():
            raise optuna.TrialPruned()

    return 1.0 - clf.score(valid_x, valid_y)

# настройка логирования
optuna.logging.get_logger("optuna").addHandler(logging.StreamHandler(sys.stdout))

# подключение к PostgreSQL для хранения результатов
storage_url = "postgresql://postgres:5555@localhost:5432/optimization_results"

# создание исследования (study)
study_name = "blobs_classification2"
sampler_name = optuna.samplers.RandomSampler(seed=42)       # семплер
RandomSampler
pruner_name = optuna.pruners.MedianPruner()      # прунер MedianPruner

study = optuna.create_study(
    sampler=sampler_name,
    pruner=pruner_name,
    study_name=study_name,
    storage=storage_url,
    direction="maximize",
    load_if_exists=False
)

# запуск оптимизации
study.optimize(objective_classification, n_trials=50)

# вывод лучших параметров и значения
print("Best params:", study.best_params)
print("Best value:", study.best_value)

# визуализация результатов оптимизации гиперпараметров
plot_optimization_history(study).show()
plot_intermediate_values(study).show()
plot_parallel_coordinate(study, params=["hidden_layer_sizes", "solver",
"alpha"]).show()
plot_contour(study, params=["hidden_layer_sizes", "alpha"]).show()
plot_slice(study, params=["hidden_layer_sizes", "alpha"]).show()
plot_param_importances(study, target=lambda t: t.value,
target_name="error").show()
plot_edf(study).show()
plot_rank(study).show()
plot_timeline(study).show()
```

б) Вывод результатов работы программы в консоле:

```
[I 2024-12-14 21:55:09,789] Trial 39 finished with value: 0.0699300699300699 and parameters: {'hidden_layer_sizes': 151, 'solver': 'adam', 'alpha': 0.008182111518618418}. Best is trial 7
 with value: 0.4475524475524476.
Trial 39 finished with value: 0.0699300699300699 and parameters: {'hidden_layer_sizes': 151, 'solver': 'adam', 'alpha': 0.008182111518618418}. Best is trial 7 with value: 0.4475524475524
476.
[I 2024-12-14 21:55:09,840] Trial 40 pruned.
Trial 40 pruned.
[I 2024-12-14 21:55:11,167] Trial 41 finished with value: 0.04195804195804198 and parameters: {'hidden_layer_sizes': 63, 'solver': 'adam', 'alpha': 5.572807500029961e-05}. Best is trial
7 with value: 0.4475524475524476.
Trial 41 finished with value: 0.04195804195804198 and parameters: {'hidden_layer_sizes': 63, 'solver': 'adam', 'alpha': 5.572807500029961e-05}. Best is trial 7 with value: 0.447552447552
4476.
[I 2024-12-14 21:55:11,227] Trial 42 pruned.
Trial 42 pruned.
[I 2024-12-14 21:55:11,284] Trial 43 pruned.
Trial 43 pruned.
[I 2024-12-14 21:55:11,341] Trial 44 pruned.
Trial 44 pruned.
[I 2024-12-14 21:55:11,393] Trial 45 pruned.
Trial 45 pruned.
[I 2024-12-14 21:55:11,446] Trial 46 pruned.
Trial 46 pruned.
[I 2024-12-14 21:55:11,500] Trial 47 pruned.
Trial 47 pruned.
[I 2024-12-14 21:55:11,557] Trial 48 pruned.
Trial 48 pruned.
[I 2024-12-14 21:55:11,611] Trial 49 pruned.
Trial 49 pruned.
Best params: {'hidden_layer_sizes': 139, 'solver': 'sgd', 'alpha': 4.809461967501571e-05}
Best value: 0.4475524475524476
```

Рис. 11. Результаты оптимизации гиперпараметров нелинейной модели MLPClassifier для датасета make_blobs с семплером RandomSampler и прунером MedianPruner

в) Визуализация графиков:

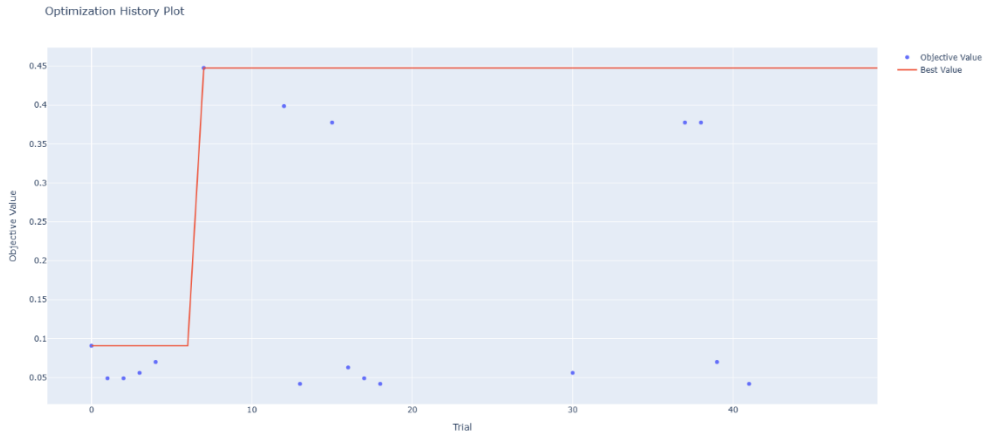

Рис. 12. Визуализация истории оптимизации при семплере с семплером RandomSampler и прунером MedianPruner

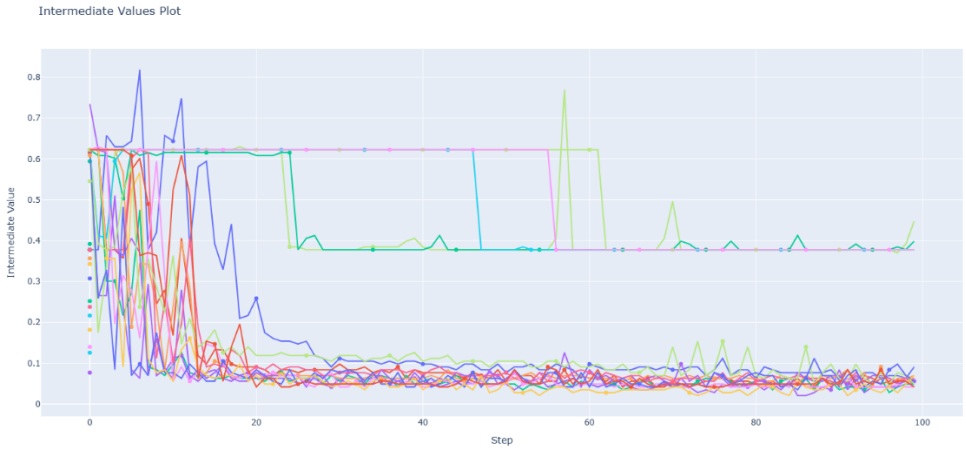

Рис. 13. Визуализация кривых обучения в ходе испытаний при семплере с семплером RandomSampler и прунером MedianPruner

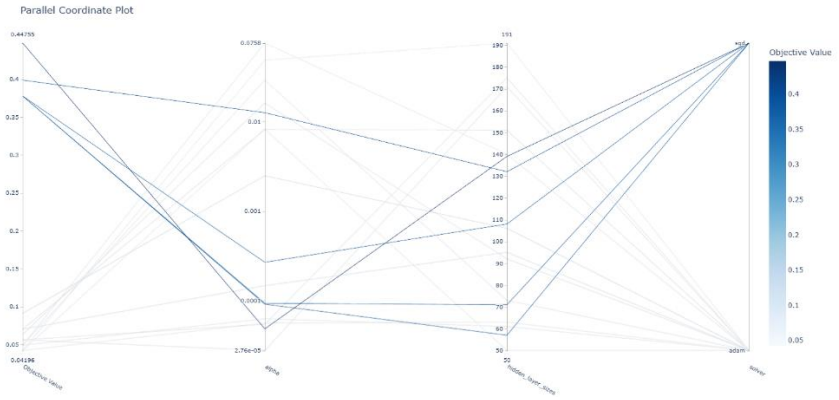

Рис. 14. Визуализация многомерных отношений параметров alpha, hidden_layer_sizes и solver при семплере с семплером RandomSampler и прунером MedianPruner

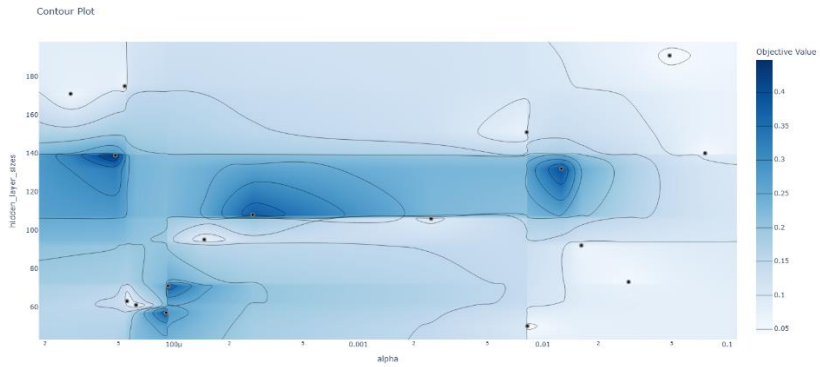

Рис. 15. Визуализация гиперпараметрических зависимостей alpha и hidden_layer_sizes и при семплере с семплером RandomSampler и прунером MedianPruner

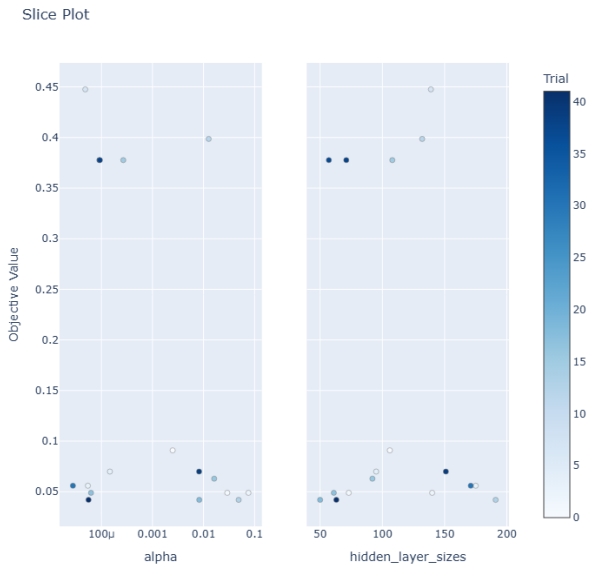

Рис. 16. Визуализация alpha и hidden_layer_sizes в виде графика срезов при семплере с семплером RandomSampler и прунером MedianPruner

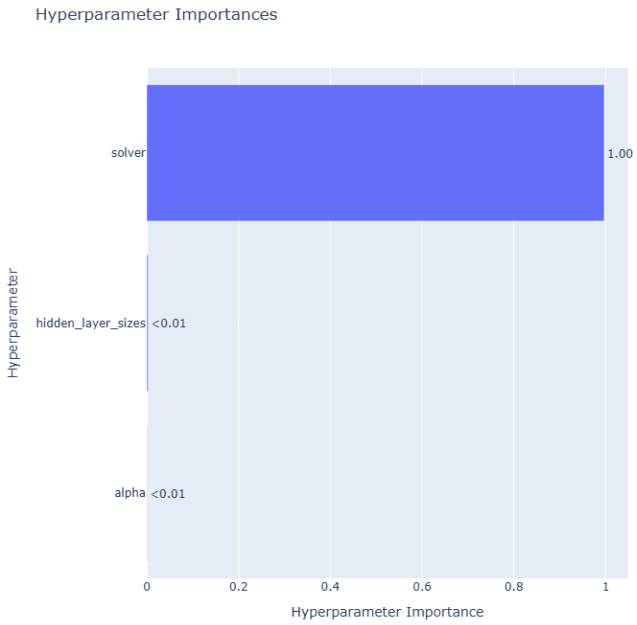

Рис. 17. Визуализация влияния alpha, hidden_layer_sizes и solver на продолжительность исследования при семплере с семплером RandomSampler и прунером MedianPruner

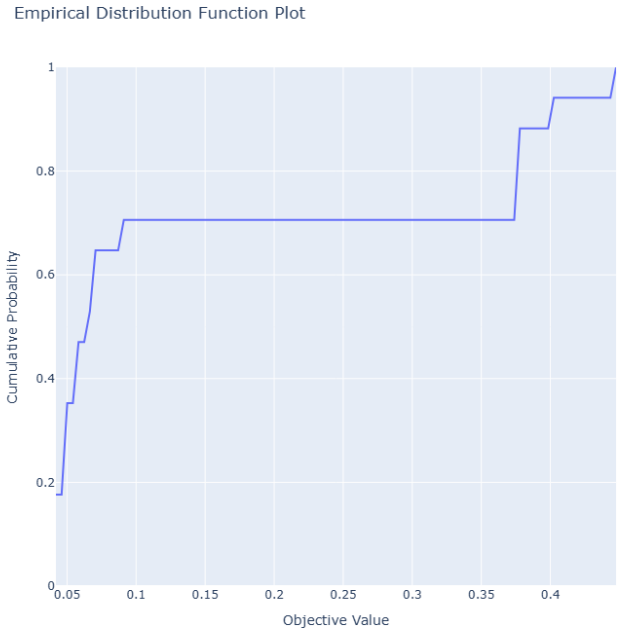

Рис. 18. Визуализация эмпирической функции распределения при семплере с семплером RandomSampler и прунером MedianPruner

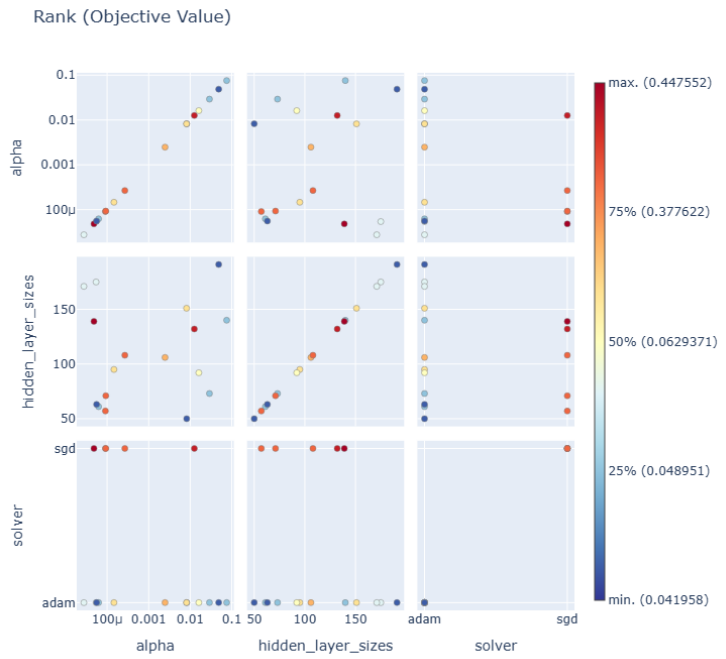

Рис. 19. Визуализация взаимосвязи параметров с помощью диаграмм рассеяния, окрашенных в соответствии с объективными значениями при семплере с семплером RandomSampler и прунером MedianPruner

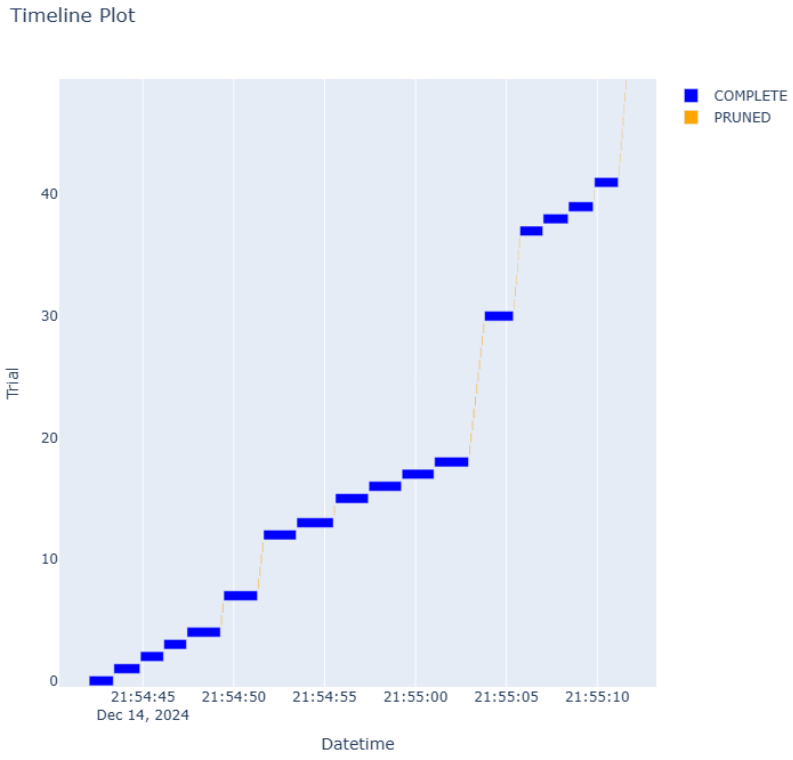

Рис. 20. Визуализация временной шкалы оптимизации выполненных испытаний при семплере с семплером RandomSampler и прунером MedianPruner