

Лабораторная работа №3. Подбор гиперпараметров модели

ГОРБАН АРТЕМИЙ М8О-307Б-23

ВЫБРАННАЯ МОДЕЛЬ: RANDOM FOREST.

ДАТАСЕТ: ПРОДУКТЫ

Подготовка данных

Цель: Предсказать, популярен ли товар (купили >1000 раз).

Изменения в данных:

Date -> Month, DayOfWeek, DayOfMonth.

Создан таргет is_popular

Признаки: Member_number, Month, DayOfWeek, DayOfMonth.

Метод: Stratified split (80/20).

Гиперпараметры Random Forest

n_estimators - Число деревьев

max_depth - Макс. глубина дерева

min_samples_split - Мин. образцов для разделения узла

min_samples_leaf - Мин. образцов в листе

max_features - Число признаков для разбиения

criterion - Критерий (gini/entropy)

Методы подбора

Grid Search: Полный перебор заданных значений.

Плюсы: Находит лучший вариант в сетке.

Минусы: Долго при большой сетке.

The screenshot shows a Jupyter Notebook cell with Python code for a grid search. The code defines a parameter grid with 'n_estimators' from 10 to 1000 and 'max_depth' from 1 to 10. It uses a Random Forest classifier and measures accuracy. The output shows the results of the search, including the best parameters found and their corresponding accuracy.

```
from sklearn.grid_search import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_iris
from sklearn import metrics
import numpy as np

# Load dataset
iris = load_iris()
X = iris.data
y = iris.target

# Define parameter grid
param_grid = {'n_estimators': np.arange(10, 1000, 100),
              'max_depth': np.arange(1, 10, 1)}

# Create classifier
clf = RandomForestClassifier(n_jobs=-1)

# Create grid search
grid_search = GridSearchCV(clf, param_grid, cv=10,
                           scoring='accuracy',
                           n_jobs=-1)
grid_search.fit(X, y)

# Print results
print("Best parameters set found: %s" % grid_search.best_params_)
print("Grid scores on development set: %s" % grid_search.grid_scores_)

print("\nDetailed classification report:\n")
print("The model is trained on the full development set.\n"
      "To make a meaningful comparison, tune this classifier\n"
      "on a smaller subset of the development set before\n"
      "navigating to a final test set.\n\n")

print("Classification report below\n\n")
print(grid_search.cv_results_['mean_test_score'])
```

n_estimators	max_depth	mean_test_score
10	1	0.40
10	2	0.40
10	3	0.40
10	4	0.40
10	5	0.40
10	6	0.40
10	7	0.40
10	8	0.40
10	9	0.40
10	10	0.40
20	1	0.40
20	2	0.40
20	3	0.40
20	4	0.40
20	5	0.40
20	6	0.40
20	7	0.40
20	8	0.40
20	9	0.40
20	10	0.40
30	1	0.40
30	2	0.40
30	3	0.40
30	4	0.40
30	5	0.40
30	6	0.40
30	7	0.40
30	8	0.40
30	9	0.40
30	10	0.40
40	1	0.40
40	2	0.40
40	3	0.40
40	4	0.40
40	5	0.40
40	6	0.40
40	7	0.40
40	8	0.40
40	9	0.40
40	10	0.40
50	1	0.40
50	2	0.40
50	3	0.40
50	4	0.40
50	5	0.40
50	6	0.40
50	7	0.40
50	8	0.40
50	9	0.40
50	10	0.40
60	1	0.40
60	2	0.40
60	3	0.40
60	4	0.40
60	5	0.40
60	6	0.40
60	7	0.40
60	8	0.40
60	9	0.40
60	10	0.40
70	1	0.40
70	2	0.40
70	3	0.40
70	4	0.40
70	5	0.40
70	6	0.40
70	7	0.40
70	8	0.40
70	9	0.40
70	10	0.40
80	1	0.40
80	2	0.40
80	3	0.40
80	4	0.40
80	5	0.40
80	6	0.40
80	7	0.40
80	8	0.40
80	9	0.40
80	10	0.40
90	1	0.40
90	2	0.40
90	3	0.40
90	4	0.40
90	5	0.40
90	6	0.40
90	7	0.40
90	8	0.40
90	9	0.40
90	10	0.40
100	1	0.40
100	2	0.40
100	3	0.40
100	4	0.40
100	5	0.40
100	6	0.40
100	7	0.40
100	8	0.40
100	9	0.40
100	10	0.40

Random Search: Случайная выборка из распределений.

Плюсы: Быстрее, часто находит хороший вариант.

Минусы: Может пропустить оптимальные значения.

The screenshot shows a Jupyter Notebook cell with Python code for a random search. The code defines a parameter grid with 'n_estimators' from 10 to 100 and 'max_depth' from 1 to 10. It uses a Random Forest classifier and measures accuracy. The output shows the results of the search, including the best parameters found and their corresponding accuracy.

```
from sklearn.grid_search import RandomizedSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_iris
from sklearn import metrics
import numpy as np

# Load dataset
iris = load_iris()
X = iris.data
y = iris.target

# Define parameter grid
param_grid = {'n_estimators': np.arange(10, 100, 10),
              'max_depth': np.arange(1, 10, 1)}

# Create classifier
clf = RandomForestClassifier(n_jobs=-1)

# Create random search
random_search = RandomizedSearchCV(clf, param_grid, cv=10,
                                    scoring='accuracy',
                                    n_iter=100,
                                    n_jobs=-1)
random_search.fit(X, y)

# Print results
print("Best parameters set found: %s" % random_search.best_params_)
print("Grid scores on development set: %s" % random_search.grid_scores_)

print("\nDetailed classification report:\n")
print("The model is trained on the full development set.\n"
      "To make a meaningful comparison, tune this classifier\n"
      "on a smaller subset of the development set before\n"
      "navigating to a final test set.\n\n")

print("Classification report below\n\n")
print(random_search.cv_results_['mean_test_score'])
```

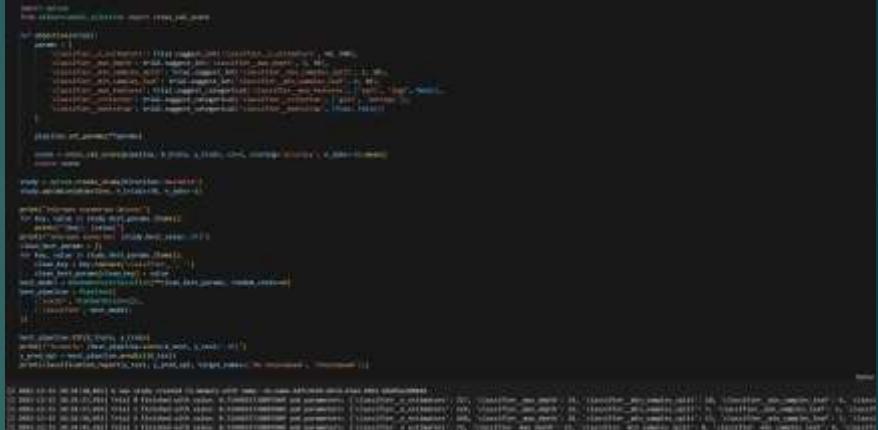
n_estimators	max_depth	mean_test_score
10	1	0.40
10	2	0.40
10	3	0.40
10	4	0.40
10	5	0.40
10	6	0.40
10	7	0.40
10	8	0.40
10	9	0.40
10	10	0.40
20	1	0.40
20	2	0.40
20	3	0.40
20	4	0.40
20	5	0.40
20	6	0.40
20	7	0.40
20	8	0.40
20	9	0.40
20	10	0.40
30	1	0.40
30	2	0.40
30	3	0.40
30	4	0.40
30	5	0.40
30	6	0.40
30	7	0.40
30	8	0.40
30	9	0.40
30	10	0.40
40	1	0.40
40	2	0.40
40	3	0.40
40	4	0.40
40	5	0.40
40	6	0.40
40	7	0.40
40	8	0.40
40	9	0.40
40	10	0.40
50	1	0.40
50	2	0.40
50	3	0.40
50	4	0.40
50	5	0.40
50	6	0.40
50	7	0.40
50	8	0.40
50	9	0.40
50	10	0.40
60	1	0.40
60	2	0.40
60	3	0.40
60	4	0.40
60	5	0.40
60	6	0.40
60	7	0.40
60	8	0.40
60	9	0.40
60	10	0.40
70	1	0.40
70	2	0.40
70	3	0.40
70	4	0.40
70	5	0.40
70	6	0.40
70	7	0.40
70	8	0.40
70	9	0.40
70	10	0.40
80	1	0.40
80	2	0.40
80	3	0.40
80	4	0.40
80	5	0.40
80	6	0.40
80	7	0.40
80	8	0.40
80	9	0.40
80	10	0.40
90	1	0.40
90	2	0.40
90	3	0.40
90	4	0.40
90	5	0.40
90	6	0.40
90	7	0.40
90	8	0.40
90	9	0.40
90	10	0.40
100	1	0.40
100	2	0.40
100	3	0.40
100	4	0.40
100	5	0.40
100	6	0.40
100	7	0.40
100	8	0.40
100	9	0.40
100	10	0.40

Методы подбора

Optuna: Интеллектуальный поиск

Плюсы: Эффективно исследует пространство, умный.

Минусы: Сложнее в настройке.

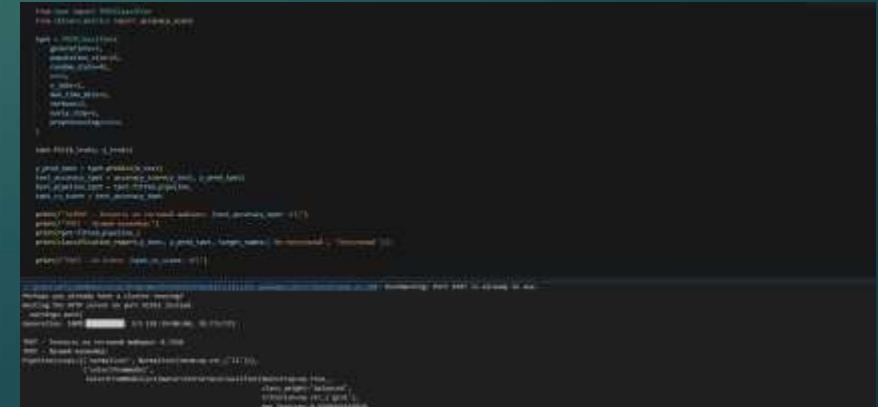


A screenshot of a terminal window displaying the output of an Optuna search process. The terminal shows numerous command-line arguments being passed to a script named 'optuna'. The arguments include various parameters such as 'n_trials=100', 'study_name=example_study', 'storage=sqlite:///example_study.db', and 'storage_type=sqlite'. The output consists of several lines of text, likely representing the progress of the search or the results of the trials.

TROT (AutoML): Автоматически ищет лучший пайплайн

Плюсы: Полная автоматизация, может найти неочевидное решение.

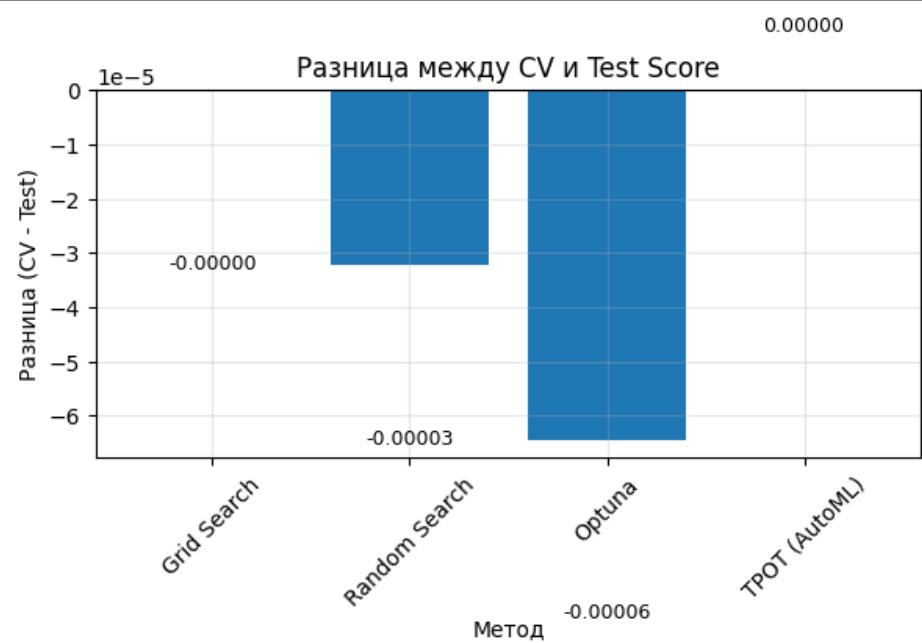
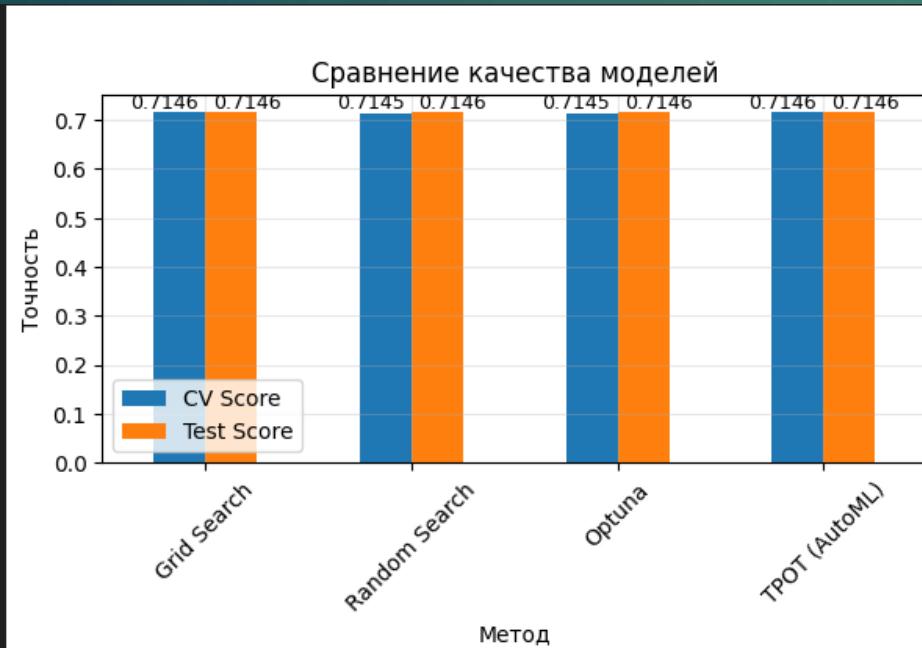
Минусы: Очень долго, чёрный ящик.



A screenshot of a terminal window displaying the output of a TROT search process. The terminal shows numerous command-line arguments being passed to a script named 'trot'. The arguments include 'n_trials=100', 'study_name=trot_study', 'storage=sqlite:///trot_study.db', and 'storage_type=sqlite'. The output consists of several lines of text, likely representing the progress of the search or the results of the trials.

Результаты подбора

	Method	CV Score	Test Score
0	Grid Search	0.714562	0.714562
1	Random Search	0.714530	0.714562
2	Optuna	0.714498	0.714562
3	TPOT (AutoML)	0.714562	0.714562

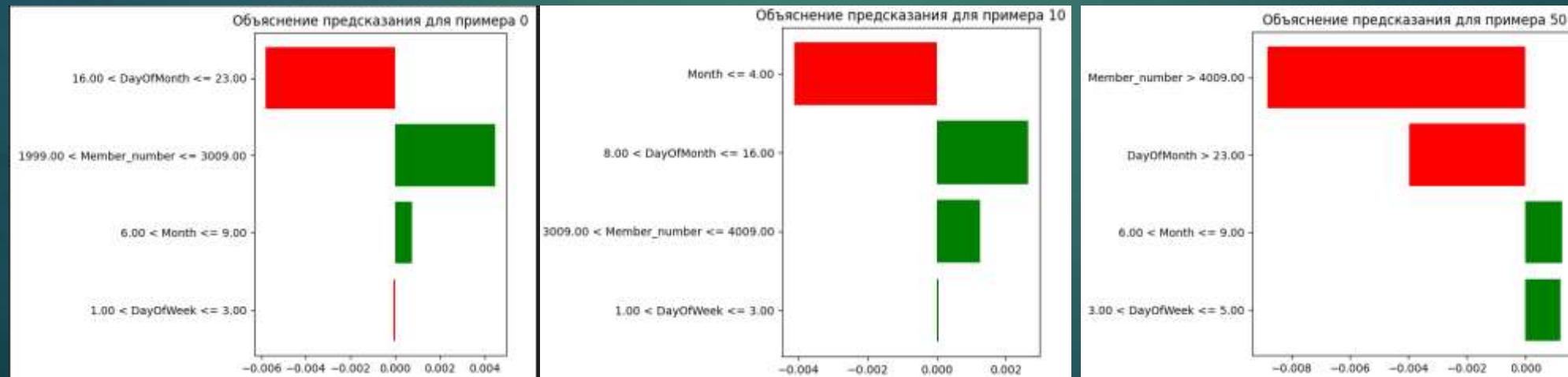


Локальная интерпретация (LIME)

Цель: Объяснить, почему для конкретного клиента модель предсказала популярный или не популярный товар.

LIME создаёт объясняющую модель для конкретного предсказания.

График, показывающий, какие признаки сильнее всего повлияли на это конкретное решение модели

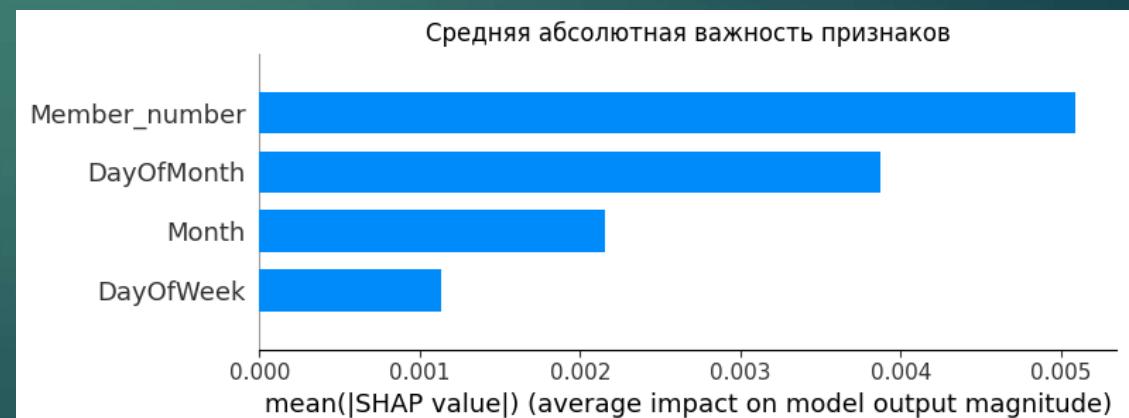
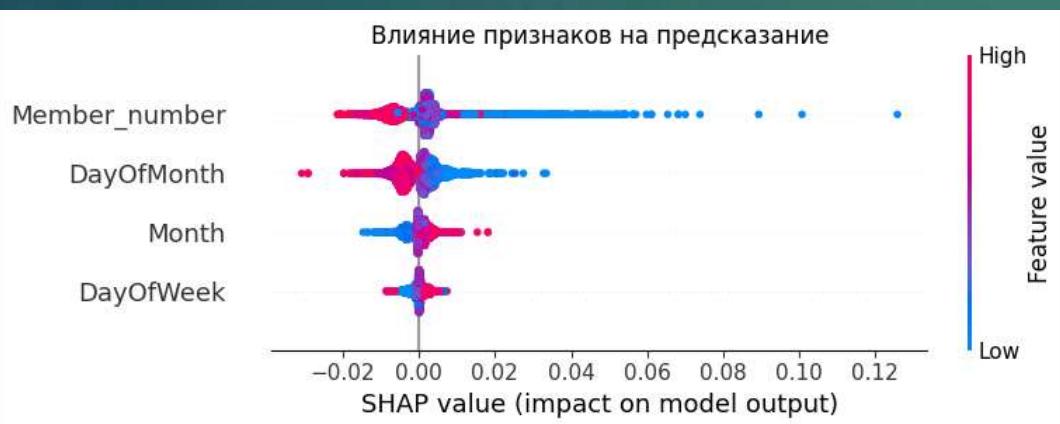


Глобальная интерпретация (SHAP)

Цель: Объяснить, почему для конкретного клиента модель предсказала популярный или не популярный товар.

LIME создаёт объясняющую модель для конкретного предсказания.

График, показывающий, какие признаки сильнее всего повлияли на это конкретное решение модели



Выводы

Grid search оказался лучшим методом подбора гиперпараметров для нашей задачи.

Качество всех методов схожее, но Grid search показала максимальную точность на тесте 0.7146.

Локальная интерпретация (LIME) помогает понять отдельные предсказания модели.

Глобальная интерпретация (SHAP) выявила, что Member_number - ключевой признак для прогноза.

Итог: Работа выполнена, модель обучена, её решения можно объяснить.