

Генерация признаков. Методы отбора признаков. Подбор гиперпараметров.

Урок 8

На этой лекции вы найдете ответы на такие вопросы как:

- Что такое генерация признаков
- Какие есть методы отбора признаков
- Как осуществляется подбор гиперпараметров



Булгакова Татьяна

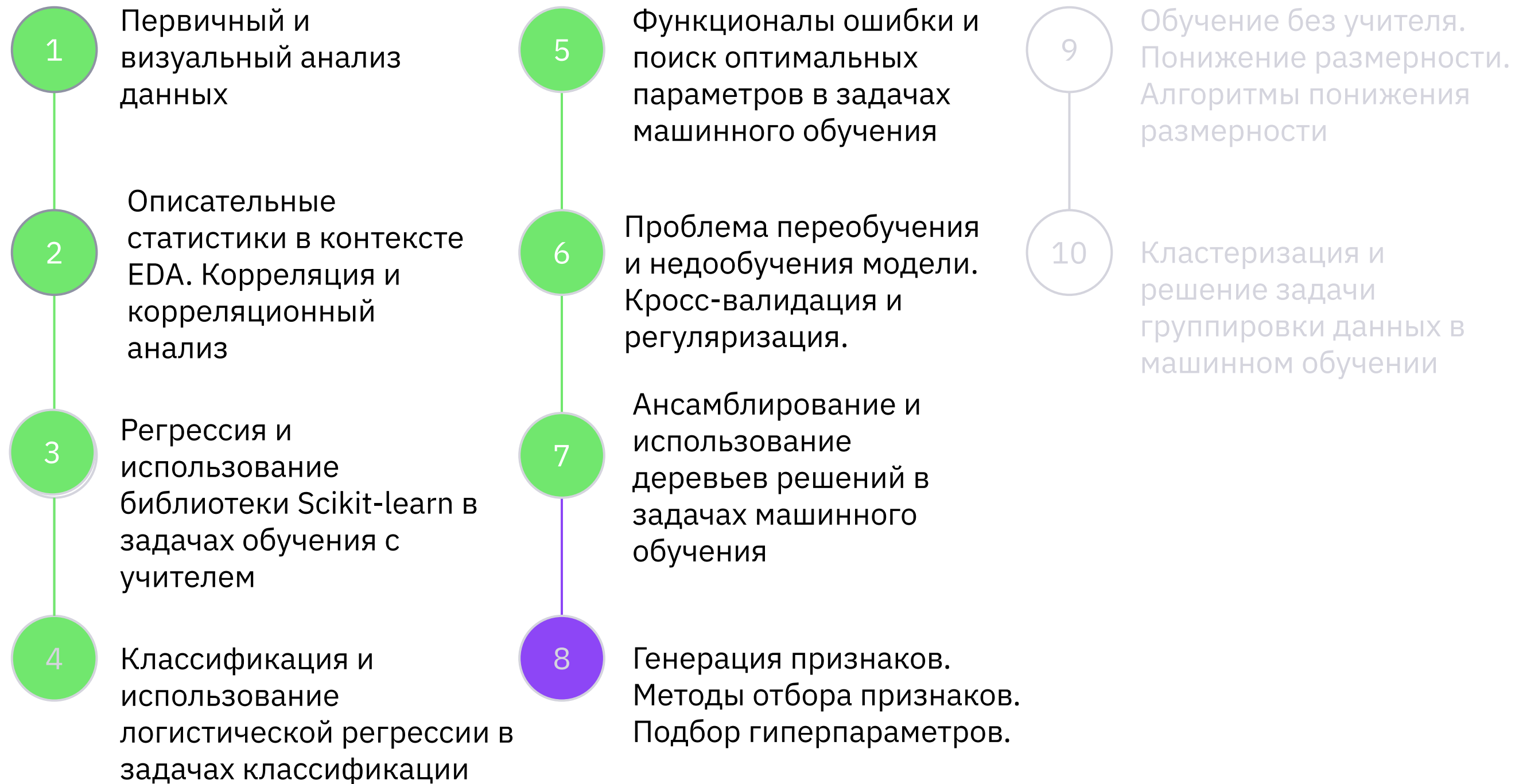
Преподаватель в GeekBrains, Нетология, Skillfactory

С 2010 года занимаюсь DataScience и NN. Фрилансер

- Участвовала в разработке программы по настройке оборудования для исследования пространственного слуха китообразных НИИ ИПЭЭ РАН
- Участвую в разработке рекомендательных систем по настройке нейростимуляторов для медицинских центров
- Работаю над курсом по нейронным сетям



План курса





Что будет на уроке сегодня

- 📌 Что такое генерация признаков
- 📌 Какие есть методы отбора признаков
- 📌 Как осуществляется подбор гиперпараметров



Генерация признаков

Генерация новых признаков является процессом создания новых переменных на основе существующих данных.

Производные от числовых переменных:

- ☒ Математические операции: Новые признаки могут быть получены путем выполнения математических операций на числовых переменных
- ☒ Логарифмы и экспоненты: Взятие логарифма или экспоненты от числовых переменных может
- ☒ Полиномиальные признаки: Создание новых признаков путем возведения в степень



Генерация признаков

Генерация новых признаков на основе взаимодействия признаков

Взаимодействия

Interaction ID	Customer ID	Дата покупки	Тип	Количество
0	0	16/07/2021 09:21:01	Добавить в корзину	N/A
1	0	16/07/2021 09:21:56	Покупка	60
2	0	17/07/2021 17:54:32	Покупка	400
3	1	16/08/2021 10:32:09	Добавить в корзину	N/A
4	1	16/08/2021 10:33:03	Покупка	30000



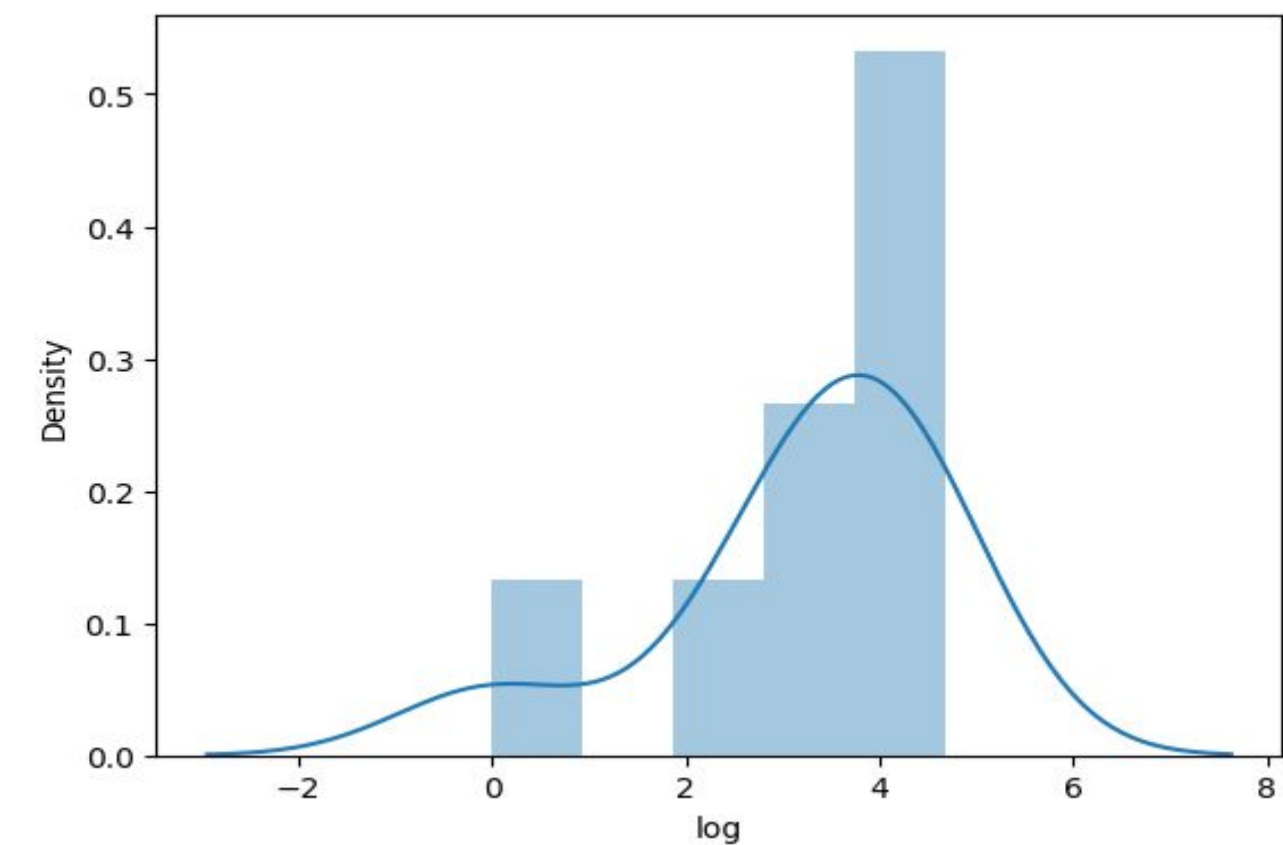
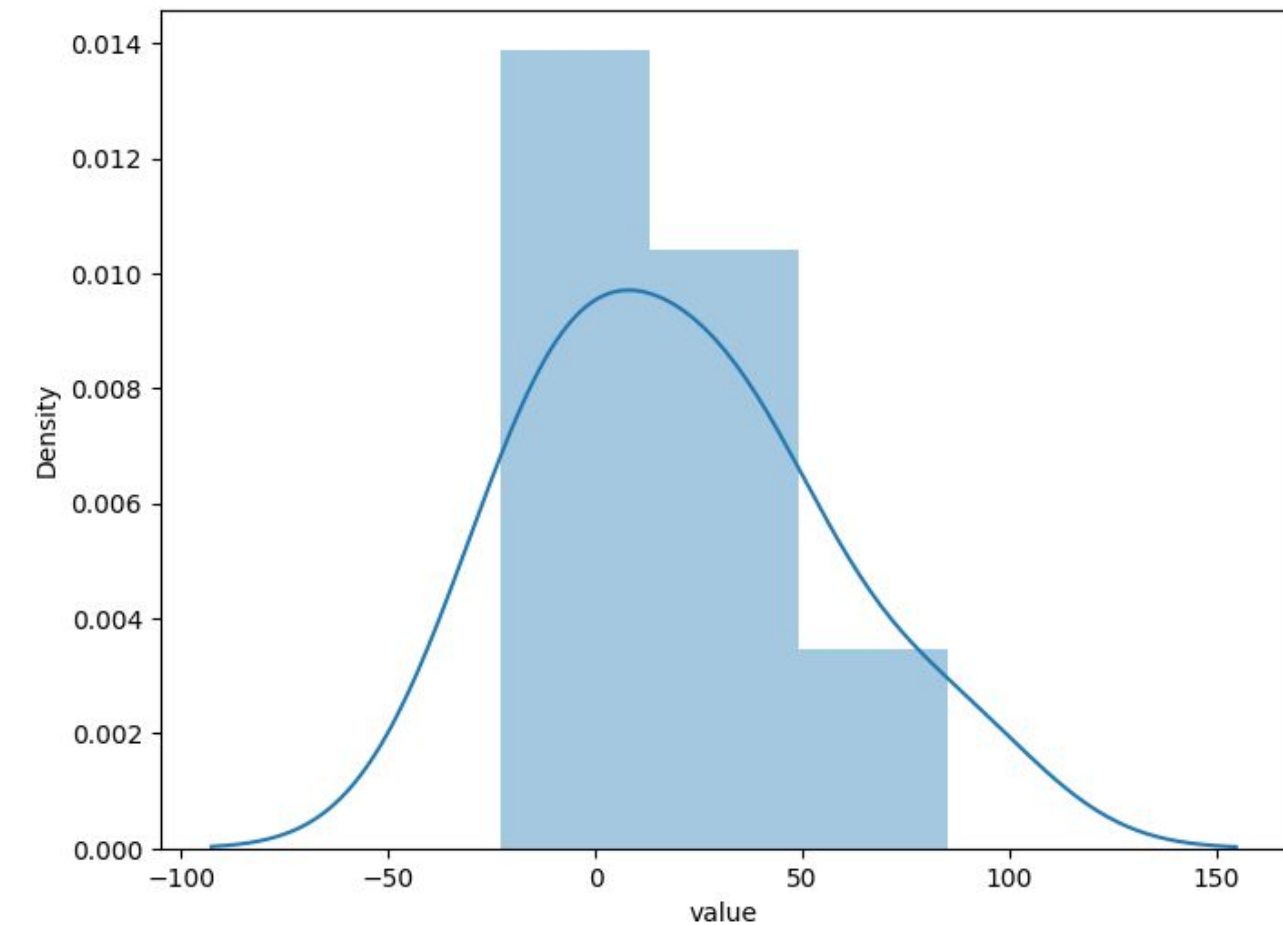
Генерация признаков

Логарифмирование

```
import pandas as pd
import numpy as np

# Пример логарифмической трансформации
data = pd.DataFrame({'value': [2, 45, -23, 85, 28, 2, 35, -12]})
data['log+1'] = (data['value']+1).transform(np.log)

# Обработка отрицательных значений
# (Обратите внимание, что значения разные)
data['log'] = (data['value']-data['value'].min()+1).transform(np.log)
```





Генерация признаков

Построение признаков на табличных данных

- ☒ Избавление от пропущенных значений
- ☒ Заполнение пропущенных значений
- ☒ Изменение масштаба признаков



Генерация признаков

Бинарные и категориальные переменные:

☒ Преобразование категориальных переменных: Категориальные переменные могут быть преобразованы в числовые с помощью методов

Company Name	Categorical value	Price
VW	1	20.000
Acura	2	10.011
Honda	3	50.000
Honda	3	10.000

VW	Acura	Honda	Price
1	0	0	20.000
0	1	0	10.011
0	0	1	50.000
0	0	1	10.000



Генерация признаков

Бинарные и категориальные переменные:

✓ Преобразование категориальных переменных: Категориальные переменные могут быть преобразованы в числовые с помощью методов

User	City
1	Roma
2	Madrid
1	Madrid
3	Istanbul
2	Istanbul
1	Istanbul
1	Roma



User	Istanbul	Madrid
1	0	0
2	0	1
1	0	1
3	1	0
2	1	0
1	1	0
1	0	0

```
train['MARRIAGE'].value_counts()
```

```
не женат/не замужем    4261
женат/замужен          3649
прочее                  90
Name: MARRIAGE, dtype: int64
```

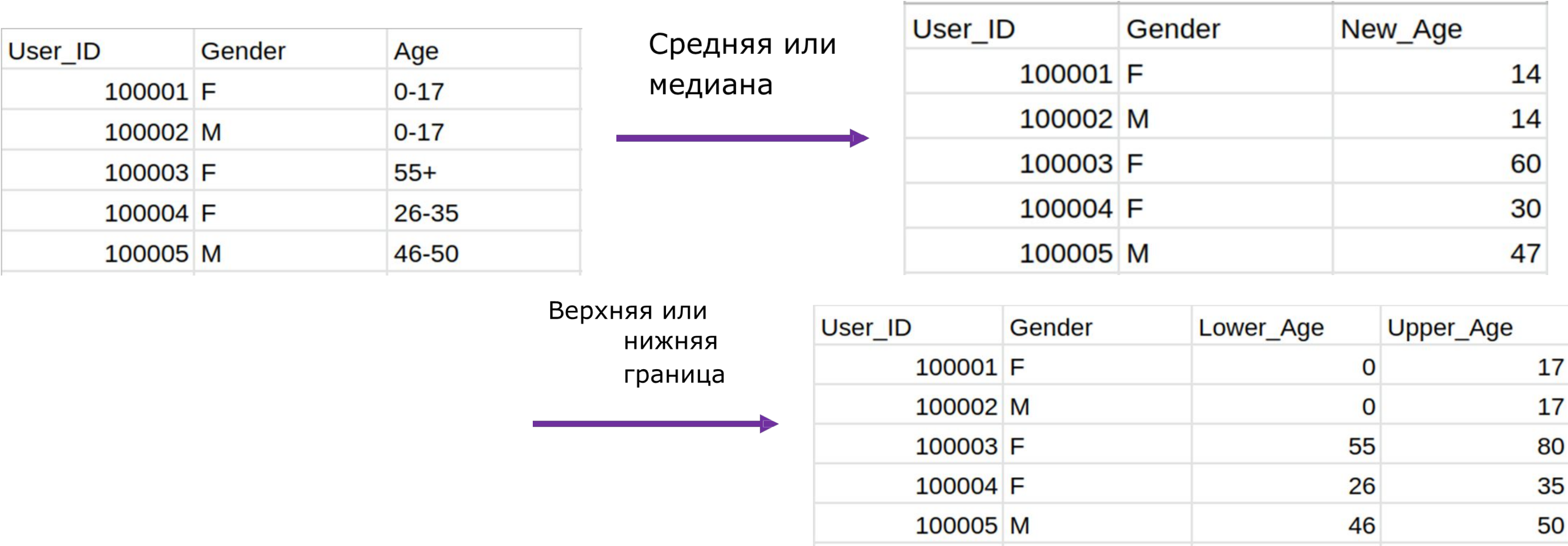
```
1 < 2 < 3
```



Генерация признаков

Бинарные и категориальные переменные:

☒ Создание комбинированных или агрегированных признаков: Новые признаки могут быть созданы на основе комбинаций или агрегации категориальных переменных





Генерация признаков

Извлечение признаков из текстовых данных:

Мама мыла раму

- Токенизация ['Мама', 'мыла', 'раму']
- Нормализация ['Мама', 'мыть', 'рама']
- Векторизация [[1, 0, 0], [0, 1, 0], [0, 0, 1]]



Генерация признаков

Этапы генерации новых признаков






- ☒ Понимание данных
- ☒ Идеи генерации признаков
- ☒ Создание новых признаков
- ☒ Оценка важности новых признаков
- ☒ Итеративный процесс
- ☒ Оптимизация и отбор признаков



Отбор признаков

Отбор признаков (фич) - это оценка важности каждого признака с использованием алгоритмов машинного обучения и удаление ненужных.

Методы фильтрации (filter methods)

-  Дисперсионный отбор (Variance Threshold):
-  Корреляционный отбор
-  Отбор на основе статистики
-  Отбор с использованием важности признаков
-  Устойчивость отбора признаков (Stability Selection)



Отбор признаков

Методы фильтрации (filter methods)



Отбор на основе статистики

```
import pandas as pd
import numpy as np
from skfeature.function.similarity_based import fisher_score
import matplotlib.pyplot as plt

# Вычисляем критерий
# Где X, y - входные и выходные данные соответственно.
ranks = fisher_score.fisher_score(X, y)

# Делаем график наших "фич"
# Где data - ваш датасет
feature_importances = pd.Series(ranks, data.columns[0:len(data.columns)-1])
feature_importances.plot(kind='barh', color='teal')
plt.show()
```

```
# Преобразование в категориальные данные путем преобразования в целые числа.
# Где X, y - входные и выходные данные соответственно.
X_categorical = X.astype(int)

# Выбираем 3 признака с наивысшим "хи-квадрат".
chi2_features = SelectKBest(chi2, k = 3)
X_kbest_features = chi2_features.fit_transform(X_categorical, y)

# Вывод "до и после"
print("Количество признаков до преобразования:", X_categorical.shape[1])
print("Количество признаков после преобразования:", X_kbest_features.shape[1])
```



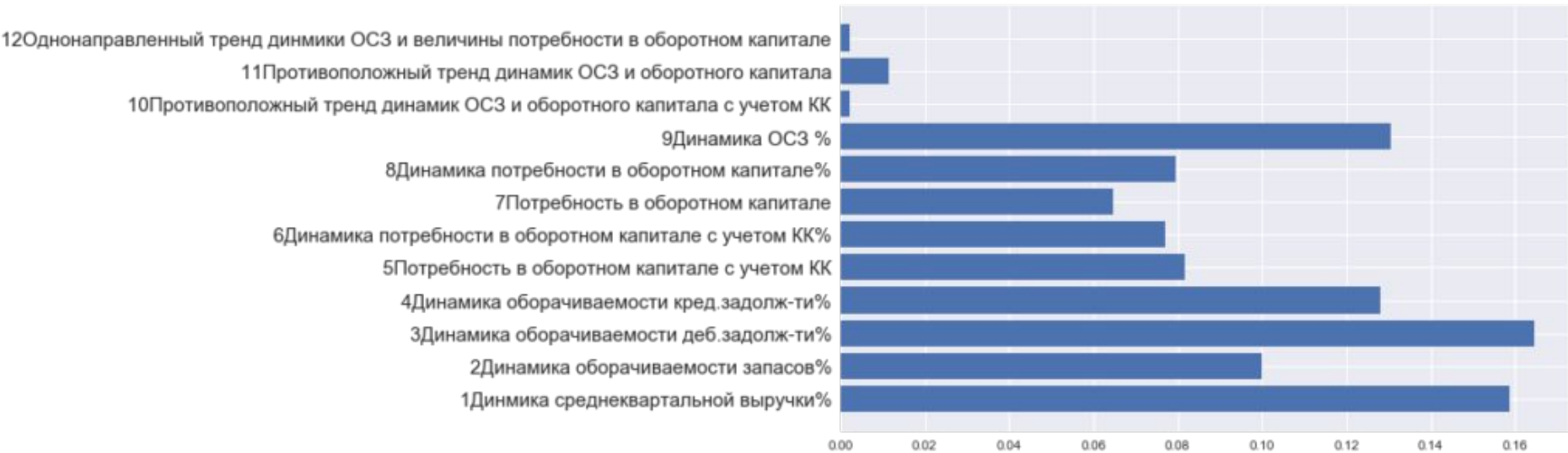
Отбор признаков

Методы фильтрации (filter methods)



Отбор с использованием важности признаков

```
#В список labels запишем названия признаков
#Так как названия признаков довольно длинные, построим горизонтальную гистограмму
position = np.arange(12)
fig, ax = plt.subplots()
ax.barh(position, model.feature_importances_)
#Устанавливаем позиции тиков:
ax.set_yticks(position)
#Устанавливаем подписи тиков
ax.set_yticklabels(labels,
                    fontsize = 15)
fig.set_figwidth(10)
fig.set_figheight(6)
plt.show()
```



Устойчивость отбора признаков (Stability Selection)



Отбор признаков

Методы фильтрации (filter methods)



Устойчивость отбора признаков (Stability Selection)

Основные шаги алгоритма:

- Исходный набор данных случайным образом разбивается на несколько подвыборок.
- На каждой подвыборке обучается модель с регуляризацией (логистическая регрессия, лес случайных деревьев и др.) и производится отбор признаков.
- Для каждого признака рассчитывается частота (сколько раз этот признак был отобран на разных подвыборках). Это называется stability score.
- Признаки сортируются по убыванию stability score.
- Отбираются признаки, у которых stability score выше некоторого порогового значения. Это наиболее стабильно отбираемые и значимые признаки.



Отбор признаков

Методы обертки (wrapper methods)

Прямой отбор признаков

Начальная модель без переменных



Добавление наиболее значимой переменной



Добавление переменных продолжается пока не достигнуто
правило остановки, или переменные не закончатся





Отбор признаков

Методы обертки (wrapper methods)

Прямой отбор признаков

```
import pandas as pd
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_regression

# Загрузим данные
df = pd.read_csv('data.csv')
X = df.drop('SalePrice', axis=1)
y = df['SalePrice']

# Применим прямой отбор признаков
selector = SelectKBest(score_func=f_regression, k=5) # Используем корреляцию и выбираем 5 признаков
X_new = selector.fit_transform(X, y)

# Получим индексы выбранных признаков
selected_indices = selector.get_support(indices=True)
selected_features = X.columns[selected_indices]

# Выведем выбранные признаки
print(selected_features)
```

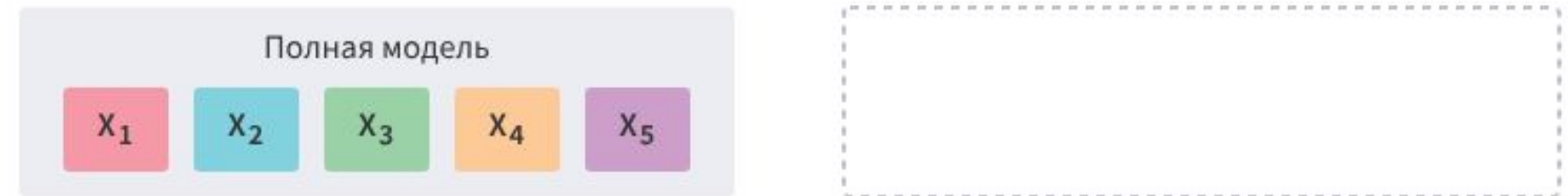


Отбор признаков

Методы обертки (wrapper methods)

Метод последовательного отбора

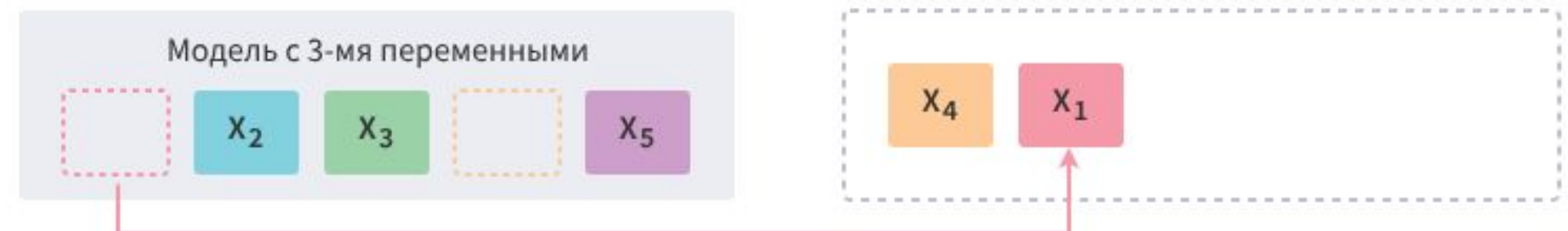
Начальная модель включает все переменные



Исключить наименее значимую переменную



Продолжить исключение пока не будет выполнено правило остановки

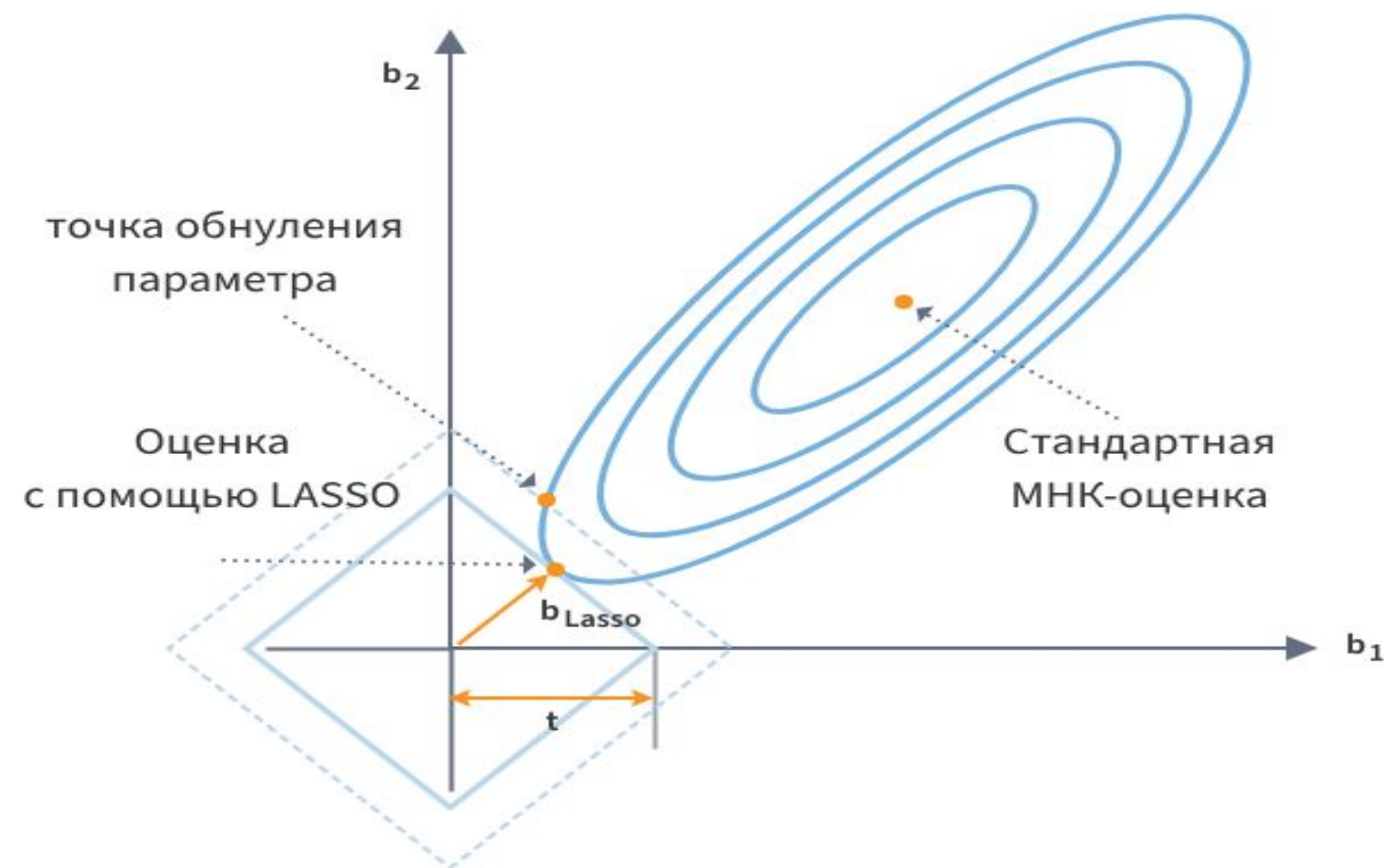




Отбор признаков

Методы встроенные (embedded methods)

L1-регуляризация (регуляризация LASSO)





Отбор признаков

Методы встроенные (embedded methods)

Рекурсивное устранение объектов (RFE)

```
from sklearn.datasets import load_iris
from sklearn.feature_selection import RFECV
from sklearn.tree import DecisionTreeClassifier

# Load the iris dataset
X, y = load_iris(return_X_y=True)

# Create a decision tree classifier
estimator = DecisionTreeClassifier()

# Use RFE with cross-validation to
# find the optimal number of features
selector = RFECV(estimator, cv=5)
selector = selector.fit(X, y)

# Print the optimal number of features
print("Optimal number of features: %d" % selector.n_features_)

# Print the selected features
print("Selected features: %s" % selector.support_)
```

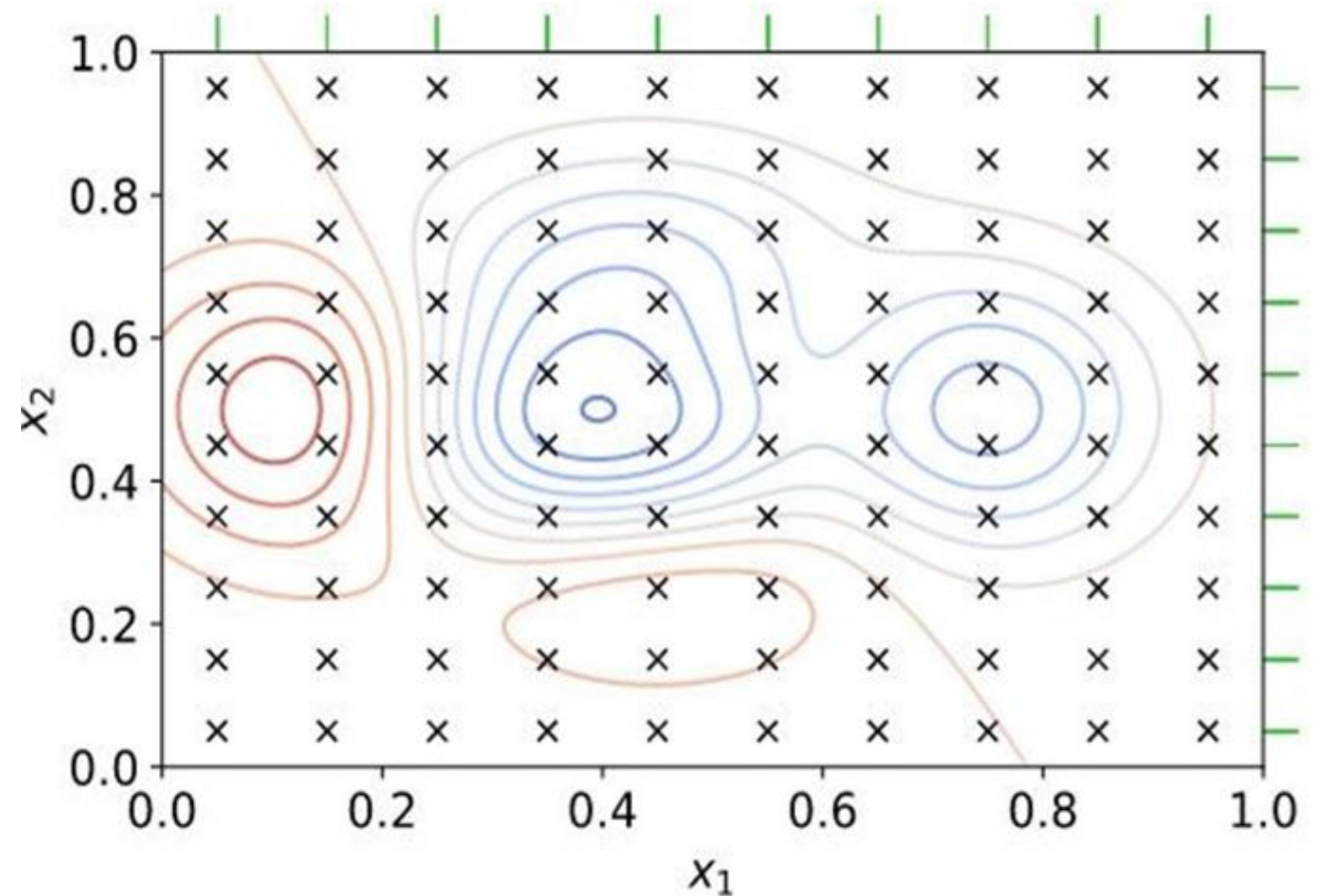
```
Optimal number of features: 3
Selected features: [False  True  True  True]
```




Подбор гиперпараметров

Grid Search

1. Фиксируются несколько значений для каждого гиперпараметра.
2. Перебираются все возможные комбинации значений различных гиперпараметров.
3. На каждой из этих комбинаций модель обучается и тестируется.
4. Выбирается комбинация, на которой модель показывает лучшее качество.





Подбор гиперпараметров

Grid Search

1. Фиксируются несколько значений для каждого гиперпараметра.
2. Перебираются все возможные комбинации значений различных гиперпараметров.
3. На каждой из этих комбинаций модель обучается и тестируется.
4. Выбирается комбинация, на которой модель показывает лучшее качество.

```
# Определение параметров и их значений для перебора
param_grid = {
    'n_estimators': [50, 100, 150],
    'max_depth': [None, 10, 20],
    'min_samples_leaf': [1, 2, 4]
}

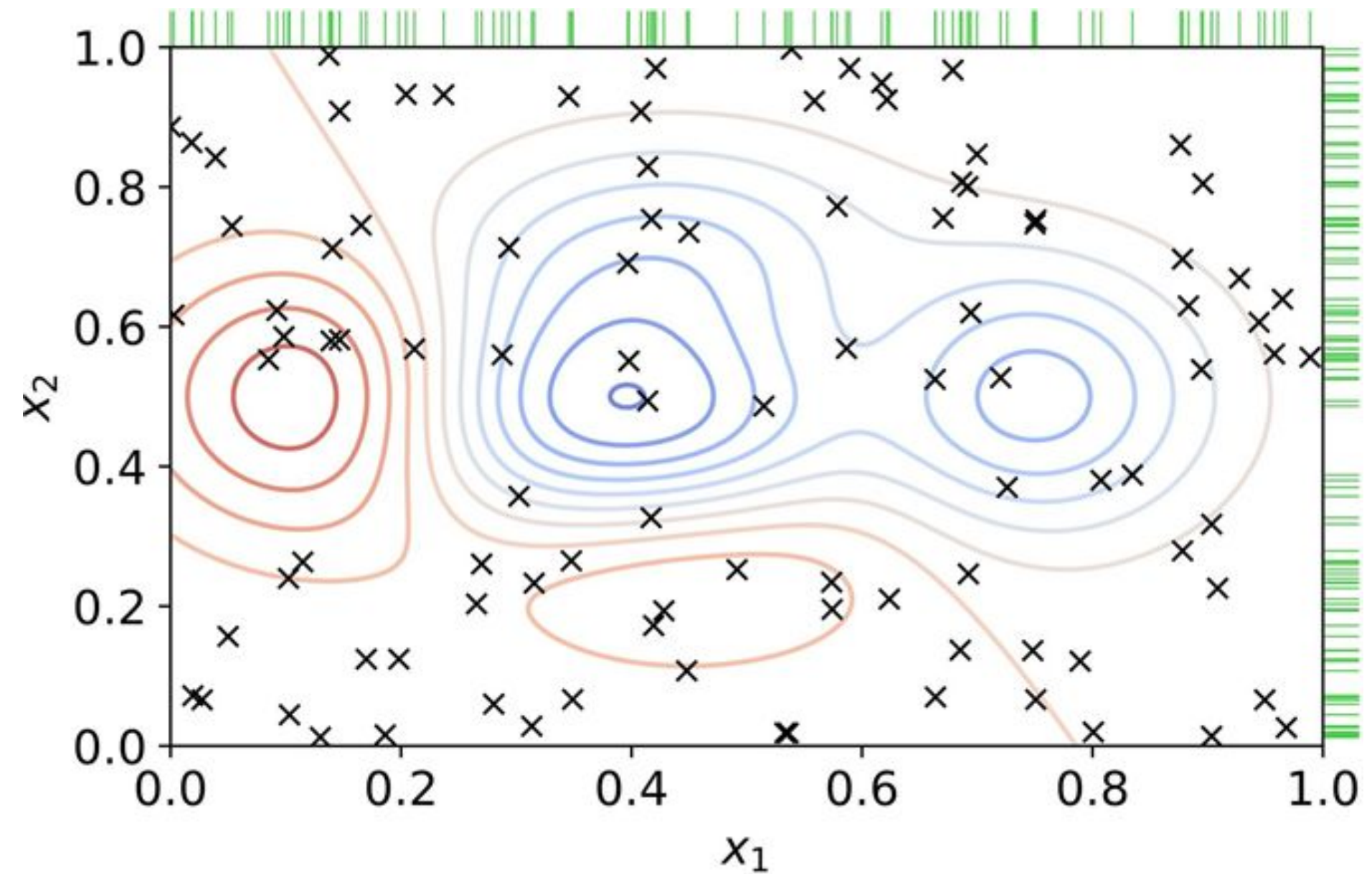
# Создание модели и настройка с использованием решетчатого поиска
rf_model = RandomForestClassifier()
grid_search = GridSearchCV(rf_model, param_grid, cv=5)
grid_search.fit(X_train, y_train)
```



Подбор гиперпараметров

Random Search

Перебирать не все комбинации гиперпараметров, а только случайное подмножество.





Подбор гиперпараметров

Random Search

Перебирать не все комбинации гиперпараметров, а только случайное подмножество.

```
# Определение диапазонов значений для случайного поиска
param_dist = {
    'n_estimators': randint(50, 200),
    'max_depth': [None, 10, 20, 30, 40, 50],
    'min_samples_leaf': [1, 2, 4]
}

# Создание модели и настройка с использованием случайного поиска
rf_model = RandomForestClassifier()
random_search = RandomizedSearchCV(rf_model, param_distributions=param_dist, n_iter=100, cv=5)
random_search.fit(X_train, y_train)

# Вывод наилучших гиперпараметров и оценки
print("Best Hyperparameters:", random_search.best_params_)
print("Best Cross-Validation Score:", random_search.best_score_)
```

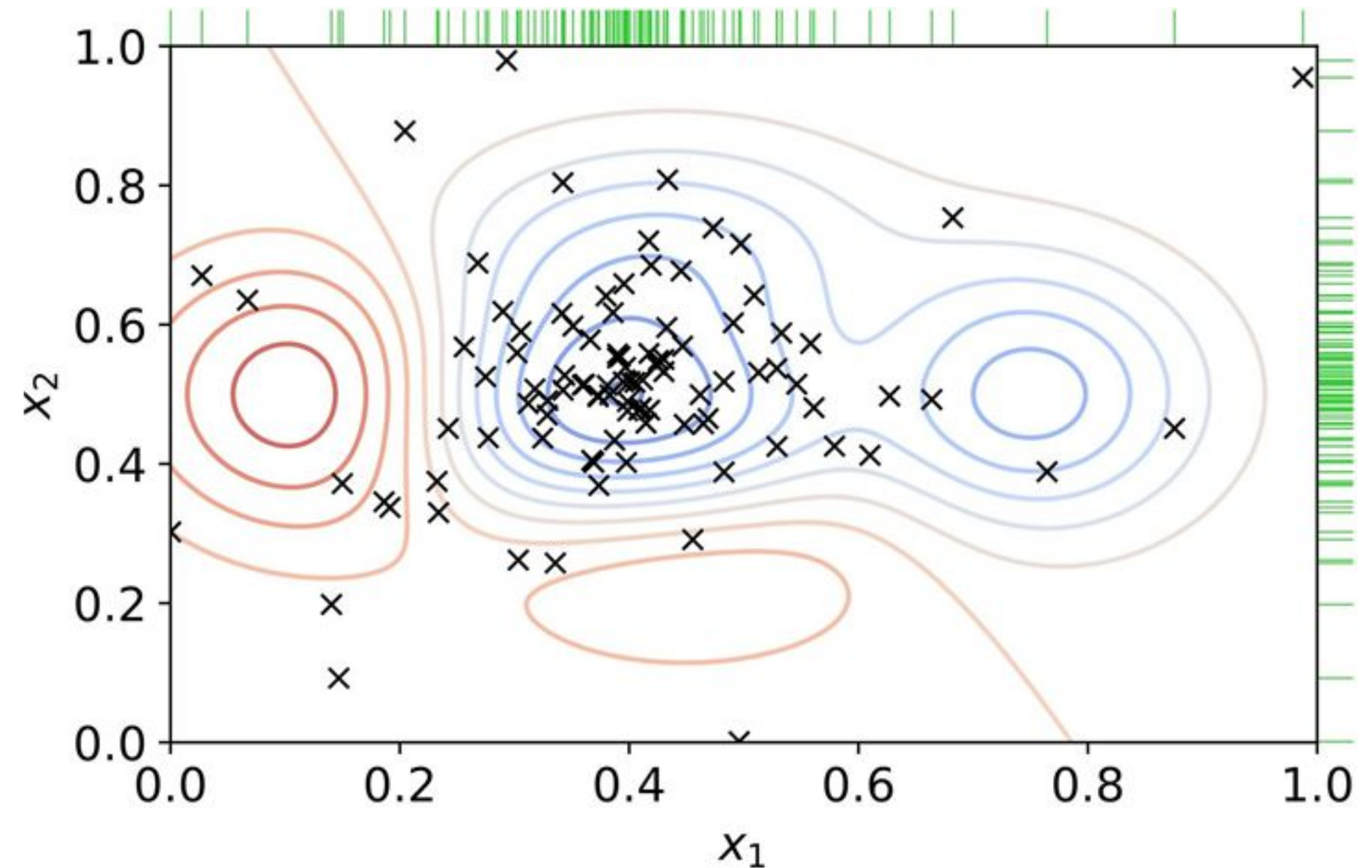



Подбор гиперпараметров

Байесовская оптимизация

Модель состоит из двух важных компонентов: гауссового процесса и алгоритма выбора следующей точки для проверки.

Алгоритм выбора следующей точки использует предсказания гауссовского процесса, чтобы найти наилучшую точку для дальнейшей проверки.





Подбор гиперпараметров

Байесовская оптимизация

Модель состоит из двух важных компонентов: гауссового процесса и алгоритма выбора следующей точки для проверки.

Алгоритм выбора следующей точки использует предсказания гауссовского процесса, чтобы найти наилучшую точку для дальнейшей проверки.

```
from skopt import BayesSearchCV
from skopt.space import Integer, Real

# Определение пространства поиска гиперпараметров
param_space = {
    'n_estimators': Integer(50, 200),
    'max_depth': Integer(10, 50),
    'min_samples_leaf': Integer(1, 4),
    'max_features': Real(0.1, 1.0, prior='uniform')
}

# Создание модели и настройка с использованием байесовской оптимизации
rf_model = RandomForestClassifier()
bayes_search = BayesSearchCV(rf_model, param_space, n_iter=50, cv=5)
bayes_search.fit(X_train, y_train)

# Вывод наилучших гиперпараметров и оценки
print("Best Hyperparameters:", bayes_search.best_params_)
print("Best Cross-Validation Score:", bayes_search.best_score_)
```




Итоги



Генерация признаков: Эффективная генерация признаков является одним из ключевых моментов в анализе данных. Хорошо подобранные признаки могут значительно улучшить качество модели. Для генерации признаков можно использовать различные методы, такие как полиномиальные и тригонометрические преобразования, производные и логарифмы, агрегацию данных и многое другое.



Методы отбора признаков: Методы отбора признаков помогают выявить наиболее информативные и значимые признаки для построения модели. Отбор признаков может осуществляться на основе статистических метрик, таких как корреляция, mutual information, chi-square и др., а также на основе моделей машинного обучения, таких как случайный лес или логистическая регрессия. Это помогает улучшить интерпретируемость модели и уменьшить размерность данных.



Итоги



Подбор гиперпараметров: Гиперпараметры это параметры модели, которые должны быть настроены до обучения и влияют на ее производительность. Подбор гиперпараметров может быть осуществлен с использованием методов перебора, таких как сеточный поиск и случайный поиск, а также с использованием более сложных алгоритмов оптимизации, таких как градиентный спуск или байесовская оптимизация. Правильно подобранные гиперпараметры позволяют достичь более высокой точности и улучшить обобщающую способность модели.



Генерация признаков, методы отбора признаков и подбор гиперпараметров являются важными этапами в процессе построения модели машинного обучения. Правильный выбор и оптимизация признаков и гиперпараметров позволяют достичь лучших результатов и повысить качество модели. Теоретические основы этих подходов также важны для понимания принципов работы моделей и их способности улавливать информацию и делать предсказания.



Спасибо за внимание

