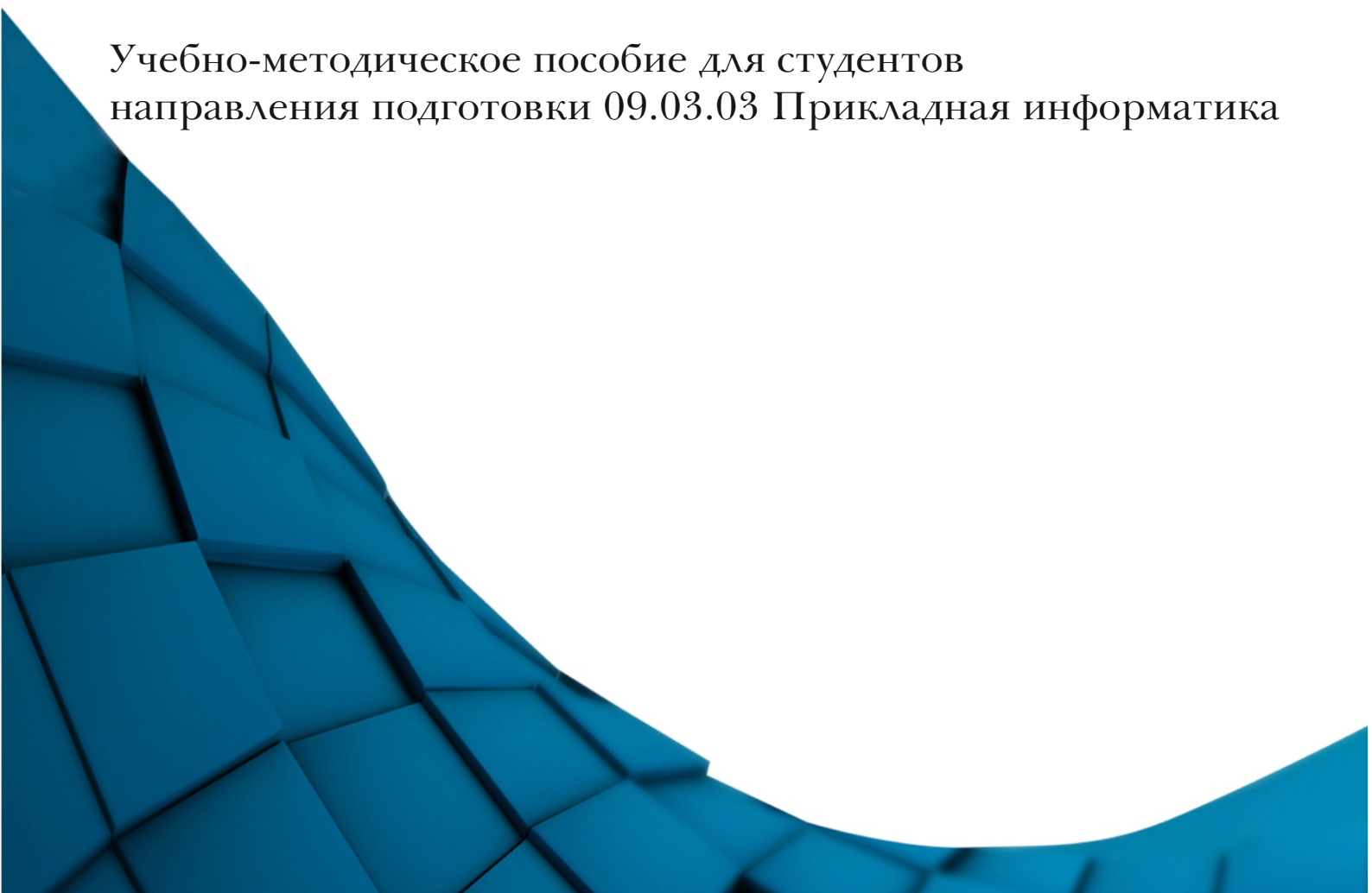


**Греченева А. В.
Ермолаева О. С.**

Машинное обучение

Практикум для студентов аграрных вузов

Учебно-методическое пособие для студентов
направления подготовки 09.03.03 Прикладная информатика



Федеральное государственное бюджетное образовательное учреждение
высшего образования «Российский государственный аграрный университет –
МСХА имени К. А. Тимирязева»

А. В. Греченева, О. С. Ермолаева

**МАШИННОЕ ОБУЧЕНИЕ.
ПРАКТИКУМ ДЛЯ СТУДЕНТОВ АГРАРНЫХ ВУЗОВ**

Учебно-методическое пособие для студентов направления подготовки
09.03.03 Прикладная информатика

Электронное текстовое издание

Санкт-Петербург
Наукоемкие технологии
2023

© Греченева А. В., Ермолаева О. С., 2023
ISBN 978-5-907618-83-1

УДК 004.8
ББК 32.813
Г81

Рецензент:

заведующий кафедрой информационных и робототехнических систем
института инженерных и цифровых технологий
Белгородского государственного национального исследовательского
университета (НИУ БелГУ)
доктор технических наук, профессор **Иващук О. А.**

Греченева А. В., Ермолаева О. С. Машинное обучение. Практикум
Г81 для студентов аграрных ВУЗов [Электронный ресурс]: учебно-
методическое пособие. – СПб.: Научно-технологические технологии, 2023. – 32 с. –
URL: <https://publishing.intelgr.com/archive/Mashinnoe-obuchenie.pdf>.

ISBN 978-5-907618-83-1

Машинное обучение. Практикум для студентов аграрных ВУЗов – учебно-методическое пособие для студентов и профессионалов, которые хотят овладеть навыками машинного обучения и применять их на практике. В пособии подробно рассмотрены основные методы машинного обучения, приведены примеры их использования в различных приложениях и представлены практические задания. Особое внимание уделено выбору признаков, регуляризации моделей, работе с несбалансированными данными и управлению ошибками. Пособие также содержит описание основных инструментов и библиотек машинного обучения, таких как Python, NumPy, Pandas, Scikit-learn и TensorFlow, а также практические советы по работе с данными. В целом, это полезный ресурс для тех, кто хочет научиться применять методы машинного обучения на практике.

УДК 004.8
ББК 32.813

ISBN 978-5-907618-83-1

© Греченева А. В., Ермолаева О. С., 2023

Учебное издание

Греченева Анастасия Владимировна
Ермолаева Ольга Сергеевна

Машинное обучение.
Практикум для студентов аграрных ВУЗов

Учебно-методическое пособие

Электронное текстовое издание

Подписано к использованию 06.09.2023.
Объем издания – 2,2 Мб.

Издательство «Наукоемкие технологии»
ООО «Корпорация «Интел Групп»
<https://publishing.intelgr.com>
E-mail: publishing@intelgr.com
Тел.: +7 (812) 945-50-63

ISBN 978-5-907618-83-1



9 785907 618831 >

Оглавление

Введение	5
Теоретическая часть	6
Практическая часть.....	14
Практическое задание № 1. Разведочный анализ данных.....	14
Практическое задание № 2. Задача бинарной классификации сбалансированного набора данных.....	16
Практическое задание № 3. Задача многоклассовой классификации сбалансированного набора данных.....	18
Практическое задание № 4. Задача классификации несбалансированного набора данных	20
Практическое задание № 5. Задача восстановления регрессии.....	22
Практическое задание № 6. Задача кластеризации	25
Практическое задание № 7. Задача выявления аномалий	27
Практическое задание № 8. Задача прогнозирования временных рядов..	29
Рекомендуемая литература	32

Введение

Машинное обучение находит все большее применение в сельском хозяйстве, позволяя улучшить производительность и эффективность различных процессов.

В растениеводстве машинное обучение может использоваться для определения оптимальных условий выращивания растений, анализа состояния почвы и определения необходимых удобрений и пестицидов. Также можно использовать машинное обучение для определения заболеваний растений и выбора наиболее эффективного способа борьбы с ними.

В животноводстве и птицеводстве машинное обучение может быть использовано для анализа поведения животных и определения их здоровья. Например, системы машинного обучения могут использоваться для мониторинга питания и активности животных, а также для определения заболеваний и прогнозирования возможных проблем.

В рыбоводстве машинное обучение может использоваться для определения оптимальных условий содержания рыб и предсказания их роста и развития. Можно также использовать машинное обучение для мониторинга качества воды и определения оптимальных условий содержания рыб.

В пчеловодстве машинное обучение может быть использовано для анализа поведения пчел и определения здоровья ульев. Можно использовать системы машинного обучения для мониторинга питания и активности пчел, а также для определения заболеваний и прогнозирования возможных проблем.

В целом, машинное обучение может помочь сельскому хозяйству повысить производительность и эффективность различных процессов, что приведет к улучшению качества и количества продукции.

Теоретическая часть

Методы машинного обучения – это алгоритмы и модели, которые используются для обработки и анализа больших объемов данных. Они позволяют компьютерным системам извлекать знания из данных и использовать их для принятия решений и предсказаний.

Существует несколько основных методов машинного обучения:

1. Обучение с учителем – это метод, при котором система обучается на основе пар «входные данные – выходные данные». Например, если мы хотим создать модель, которая будет определять, является ли электронное письмо спамом или нет, мы будем использовать данные с пометками «спам» или «не спам».

Обучение с учителем (Supervised Learning) – это один из видов машинного обучения, при котором модель обучается на основе предоставленных ей данных, содержащих правильные ответы (метки). Таким образом, модель должна научиться предсказывать правильный ответ для новых данных, которых она не видела ранее.

Процесс обучения с учителем состоит из нескольких этапов:

1. Подготовка данных: данные должны быть представлены в числовом виде и разделены на две части – обучающую и тестовую выборки.

2. Выбор модели: выбирается модель, которая будет использоваться для решения задачи. Это может быть любой алгоритм машинного обучения, такой как линейная регрессия, деревья решений, нейронные сети и т.д.

3. Обучение модели: модель обучается на обучающей выборке. Для этого используется оптимизационный алгоритм, который настраивает параметры модели таким образом, чтобы минимизировать ошибку предсказания на обучающей выборке.

4. Оценка модели: после завершения обучения модель проверяется на тестовой выборке. Оценка производится на основе метрик, таких как точность (accuracy), полнота (recall), F1-мера (F1-score) и т.д.

5. Использование модели: после успешной оценки модель может быть использована для предсказания ответов на новых данных.

Обучение с учителем может применяться в различных областях, таких как распознавание образов, классификация текстов, прогнозирование временных рядов и т.д. Важно понимать, что качество модели зависит от качества данных, на которых она обучается, а также от правильного выбора модели и ее параметров.

Кейс машинного обучения с учителем для сельского хозяйства может быть связан с прогнозированием урожайности определенных культурных растений в зависимости от различных факторов, таких как погода, тип почвы, уровень удобрений и т.д.

Примером такого кейса может быть использование датасета "Crop Yield Prediction Dataset" (<https://www.kaggle.com/atharvaingle/crop-yield-prediction-dataset>), который содержит информацию о кукурузе, пшенице и рисе, выращиваемых в Индии в период с 2000 по 2014 годы. Датасет содержит

следующие признаки: год, штат, округ, площадь посева, количество удобрений, тип почвы, количество осадков, количество солнечного света, средняя температура и урожайность.

Пример кода для обучения модели прогнозирования урожайности кукурузы на основе датасета Crop Yield Prediction Dataset:

```
python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Загрузка датасета
data = pd.read_csv("Crop_yield_prediction.csv")

# Выбор признаков и целевой переменной
X = data[['Rainfall', 'Temperature', 'Humidity',
'Fertilizers']]
y = data['Maize']

# Разбиение на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Обучение модели
model = LinearRegression()
model.fit(X_train, y_train)

# Прогнозирование урожайности кукурузы на тестовой выборке
y_pred = model.predict(X_test)

# Оценка качества модели
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

В данном примере мы использовали линейную регрессию для прогнозирования урожайности кукурузы на основе признаков Rainfall (количество осадков), Temperature (средняя температура), Humidity (влажность) и Fertilizers (количество удобрений). Мы также использовали метрику Mean Squared Error для оценки качества модели.

2. Обучение без учителя – это метод, при котором система обучается на основе не размеченных данных. Это означает, что система должна самостоятельно выявлять закономерности в данных. Примером такого метода может быть кластеризация данных – разделение набора данных на группы на основе их сходства.

Обучение без учителя (Unsupervised Learning) – это один из видов машинного обучения, при котором модель обучается на основе не размеченных данных, то есть без предоставления правильных ответов (меток).

Задача модели заключается в том, чтобы самостоятельно выявить закономерности и структуру в данных.

Процесс обучения без учителя состоит из нескольких этапов:

1. Подготовка данных: данные должны быть представлены в числовом виде и могут быть разделены на обучающую и тестовую выборки.

2. Выбор модели: выбирается модель, которая будет использоваться для решения задачи. Это может быть любой алгоритм машинного обучения, такой как кластеризация, снижение размерности и т.д.

3. Обучение модели: модель обучается на обучающей выборке без предоставления меток. Для этого используется оптимизационный алгоритм, который настраивает параметры модели таким образом, чтобы минимизировать ошибку на данных.

4. Оценка модели: оценка модели может происходить по различным метрикам, таким как инерция кластеров или среднеквадратичное отклонение при снижении размерности.

5. Использование модели: после успешной оценки модель может быть использована для выявления структуры и закономерностей в новых данных.

Основными методами обучения без учителя являются кластеризация и снижение размерности.

Кластеризация – это метод, при котором модель разделяет данные на группы (кластеры) на основе их сходства. Для этого используются различные алгоритмы, такие как k-средних и DBSCAN.

Снижение размерности – это метод, при котором модель уменьшает количество признаков в данных, сохраняя при этом основную информацию. Для этого используются различные алгоритмы, такие как метод главных компонент (PCA) и t-SNE.

Обучение без учителя может применяться в различных областях, таких как анализ данных, поиск аномалий, сжатие данных и т.д. Важно понимать, что качество модели зависит от качества данных и правильного выбора метода и его параметров.

Примером кейса машинного обучения без учителя для сельского хозяйства может быть кластеризация почвенных образцов на основе их физико-химических свойств. Для этого можно использовать датасет "Soil Types of India" (<https://www.kaggle.com/omkargurav/soil-types-of-india>), который содержит информацию о различных типах почвы в Индии.

Пример кода для кластеризации почвенных образцов на основе датасета Soil Types of India:

```
python
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

# Загрузка датасета
data = pd.read_csv("Soil_Types_of_India.csv")
```

```

# Выбор признаков
X = data[['pH', 'EC', 'OC', 'N', 'P', 'K', 'Ca', 'Mg', 'S']]

# Стандартизация признаков
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Кластеризация методом k-средних
kmeans = KMeans(n_clusters=5, random_state=42)
kmeans.fit(X_scaled)

# Визуализация кластеров
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=kmeans.labels_,
            cmap='rainbow')
plt.title('Clustering of Soil Types')
plt.xlabel('pH')
plt.ylabel('EC')
plt.show()

```

В данном примере мы использовали метод k-средних для кластеризации почвенных образцов на основе признаков pH, EC, OC, N, P, K, Ca, Mg и S. Мы также использовали стандартизацию признаков с помощью класса `StandardScaler` из библиотеки `scikit-learn`. Визуализация кластеров была выполнена с помощью библиотеки `matplotlib`.

Обучение с подкреплением – это метод, при котором система обучается на основе опыта, получаемого в процессе взаимодействия с окружающей средой. Примером такого метода может быть создание игрового бота, который обучается на основе игр с другими игроками.

Обучение с подкреплением (`Reinforcement Learning`) – это один из видов машинного обучения, при котором модель обучается на основе взаимодействия с окружающей средой. В отличие от обучения с учителем и без учителя, модель не получает явных меток или наблюдений, а взаимодействует с окружающей средой, получая только награду или штраф за каждое действие.

Процесс обучения с подкреплением состоит из нескольких этапов:

1. Определение задачи: определяется задача, которую необходимо решить. Например, это может быть игра в шахматы или управление роботом.

2. Определение состояний и действий: определяются состояния, в которых может находиться модель, и действия, которые она может совершать в каждом состоянии.

3. Определение функции награды: определяется функция, которая определяет количество награды или штрафа за каждое действие в каждом состоянии.

4. Обучение модели: модель обучается на основе взаимодействия с окружающей средой. Для этого используется алгоритм, который настраивает параметры модели таким образом, чтобы максимизировать награду.

5. Оценка модели: оценка модели может происходить по различным метрикам, таким как средняя награда за эпизод или время, затраченное на решение задачи.

6. Использование модели: после успешной оценки модель может быть использована для решения задачи.

Основными методами обучения с подкреплением являются Q-обучение и методы глубокого обучения, такие как Deep Q-Networks (DQN) и Policy Gradient.

Q-обучение – это метод, при котором модель вычисляет функцию $Q(s,a)$, которая определяет ожидаемую награду за выполнение действия a в состоянии s . DQN – это метод, при котором модель использует нейронную сеть для вычисления функции $Q(s,a)$. Для этого используется алгоритм, который настраивает параметры нейронной сети таким образом, чтобы максимизировать награду.

Policy Gradient – это метод, при котором модель вычисляет функцию политики, которая определяет вероятность выполнения каждого действия в каждом состоянии.

Обучение с подкреплением может применяться в различных областях, таких как игры, управление роботами, финансы и т.д. Важно понимать, что качество модели зависит от правильного выбора алгоритма и его параметров, а также от качества функции награды.

Примером кейса машинного обучения с подкреплением для сельского хозяйства может быть оптимизация удобрений на основе данных о росте растений и содержании питательных веществ в почве. Для этого можно использовать датасет "Corn Yield Data" (<https://www.kaggle.com/atharvaingle/corn-yield-data>), который содержит информацию о урожайности кукурузы и содержании питательных веществ в почве на различных полях в Индии.

Пример кода для оптимизации удобрений на основе датасета Corn Yield Data:

```
python
import pandas as pd
import numpy as np
import gym

# Загрузка датасета
data = pd.read_csv("corn_yield_data.csv")

# Выбор признаков
X = data[['N', 'P', 'K']]
y = data['Yield']

# Определение окружения OpenAI Gym
class CornYieldEnv(gym.Env):
    metadata = {'render.modes': ['human']}
```

```

def __init__(self, X, y):
    super(CornYieldEnv, self).__init__()
    self.X = X
    self.y = y
    self.action_space = gym.spaces.Box(low=0, high=1,
shape=(3,))
    self.observation_space = gym.spaces.Box(low=0,
high=1, shape=(3,))
    self.reset()

def step(self, action):
    reward = -np.mean((self.X @ action - self.y) ** 2)
    self.observation = action
    done = True
    return self.observation, reward, done, {}

def reset(self):
    self.observation = np.zeros(3)
    return self.observation

# Обучение модели с подкреплением
env = CornYieldEnv(X, y)
obs = env.reset()
for i in range(1000):
    action = env.action_space.sample()
    obs, reward, done, _ = env.step(action)

```

В данном примере мы использовали окружение OpenAI Gym для оптимизации удобрений на основе признаков N, P и K и целевой переменной Yield. Мы определили класс CornYieldEnv, который наследуется от класса gym.Env, и реализовали методы step и reset для выполнения действий и получения состояний окружения. Мы также определили случайное действие для каждой эпохи обучения.

Средства для реализации машинного обучения:

1. Python – один из наиболее популярных языков программирования для машинного обучения. В Python существует множество библиотек и фреймворков, таких как TensorFlow, Keras, PyTorch, Scikit-learn и др., которые позволяют создавать и обучать модели машинного обучения.

2. Библиотеки машинного обучения – это наборы инструментов и алгоритмов, которые позволяют создавать и обучать модели машинного обучения. Некоторые из наиболее популярных библиотек машинного обучения включают TensorFlow, Keras, PyTorch, Scikit-learn, Pandas, Numpy и др.

3. Облачные сервисы – это сервисы, которые предоставляют доступ к вычислительным ресурсам и инструментам для машинного обучения через интернет. Некоторые из наиболее популярных облачных сервисов для машинного обучения включают Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure и др.

4. Графические процессоры (GPU) – это ускорители, которые позволяют обрабатывать большие объемы данных и ускорять процесс обучения моделей машинного обучения. Некоторые из наиболее популярных GPU для машинного обучения включают Nvidia GeForce, AMD Radeon и др.

5. Базы данных – это средства хранения и организации данных, которые используются для обучения моделей машинного обучения. Некоторые из наиболее популярных баз данных для машинного обучения включают MySQL, PostgreSQL, MongoDB, Cassandra и др.

6. Инструменты визуализации – это инструменты, которые позволяют визуализировать данные и результаты работы моделей машинного обучения. Некоторые из наиболее популярных инструментов визуализации для машинного обучения включают Matplotlib, Seaborn, Plotly и др.

Важно понимать, что выбор средств для машинного обучения зависит от конкретной задачи и требований к модели. Кроме того, необходимо учитывать доступность и стоимость средств, а также уровень опыта и знаний пользователей.

Ниже перечислены некоторые из наиболее популярных библиотек Python для машинного обучения:

1. TensorFlow – это библиотека от Google, которая позволяет создавать и обучать модели машинного обучения. TensorFlow предоставляет широкий набор инструментов для работы с данными, создания моделей и их оптимизации. Она также поддерживает распределенное обучение на нескольких устройствах.

2. Keras – это высокоуровневый фреймворк для создания нейронных сетей. Он позволяет создавать модели машинного обучения с помощью простых и понятных API. Keras также поддерживает интеграцию с TensorFlow и другими библиотеками машинного обучения.

3. PyTorch – это библиотека от Facebook, которая предоставляет удобный интерфейс для создания и обучения нейронных сетей. PyTorch также поддерживает автоматическое дифференцирование, что позволяет легко оптимизировать модели.

4. Scikit-learn – это библиотека для машинного обучения, которая предоставляет широкий набор алгоритмов для классификации, регрессии, кластеризации и других задач. Scikit-learn также предоставляет инструменты для работы с данными, включая предобработку и визуализацию.

5. Pandas – это библиотека для работы с данными, которая предоставляет инструменты для чтения, записи и манипулирования табличными данными. Pandas также поддерживает работу с временными рядами и многомерными данными.

6. NumPy – это библиотека для работы с массивами данных, которая предоставляет инструменты для математических операций, линейной алгебры и статистики. NumPy также поддерживает работу с многомерными массивами данных.

Важно отметить, что это не полный список библиотек Python для машинного обучения, и выбор конкретной библиотеки зависит от требований к модели и уровня опыта пользователя.

Метрики машинного обучения – это инструменты, которые используются для оценки качества модели машинного обучения. Они позволяют измерить, насколько хорошо модель работает на тестовых данных и как точно она делает прогнозы.

Существует несколько типов метрик машинного обучения, каждый из которых имеет свои преимущества и недостатки. Однако, наиболее распространенные метрики включают в себя:

1. Accuracy (точность) – это метрика, которая показывает, как часто модель правильно классифицирует объекты. Она вычисляется как отношение числа правильно классифицированных объектов к общему числу объектов в тестовом наборе данных.

2. Precision (точность) – это метрика, которая показывает, как много объектов, которые модель отнесла к положительному классу, действительно являются положительными. Она вычисляется как отношение числа верно классифицированных положительных объектов к общему числу объектов, которые модель отнесла к положительному классу.

3. Recall (полнота) – это метрика, которая показывает, как много положительных объектов модель нашла из всех возможных положительных объектов. Она вычисляется как отношение числа верно классифицированных положительных объектов к общему числу положительных объектов в тестовом наборе данных.

4. F1-score (F-мера) – это метрика, которая объединяет точность и полноту в единую метрику. Она вычисляется как гармоническое среднее точности и полноты.

5. ROC AUC (площадь под кривой ROC) – это метрика, которая показывает, насколько хорошо модель разделяет положительные и отрицательные классы. Она вычисляется как площадь под кривой ROC, которая показывает зависимость между чувствительностью и специфичностью модели.

Выбор конкретной метрики зависит от задачи машинного обучения и требований к модели. Например, если задача заключается в классификации медицинских изображений, то recall может быть более важной метрикой, чем precision. Однако, в целом, accuracy и F1-score являются наиболее универсальными метриками для оценки качества модели машинного обучения.

Практическая часть

Практическое задание № 1. Разведочный анализ данных

1. Загрузить набор данных по мировому населению (<https://www.kaggle.com/datasets/iamsouravbanerjee/world-population-dataset?datasetId=2432740&sortBy=voteCount>) как датафрейм библиотеки pandas

2. Получить представления о наборе данных с помощью методов shape, head, describe, info библиотеки pandas. Посчитать количество пустых значений в признаках с помощью библиотеки numpy

3. Произвести разведочный анализ данных по данному набору данных – визуализировать данные с помощью библиотек matplotlib, seaborn, plotly. Построить:

- 1) Парные диаграммы (pairplots)
- 2) Тепловую карту по матрице корреляции
- 3) Диаграммы рассеяния
- 4) Сводные диаграммы
- 5) Гистограммы
- 6) Ящик с усами (диаграмма размаха)
- 7) Скрипичные диаграммы

4. На основе выполненного анализа данных произвести выводы о динамике изменения населения по регионам, процентном соотношении населения по регионам. Выявить корреляцию между признаками

Пример

Допустим, у нас есть датасет с информацией о продажах продуктов в магазине за последний год. Мы хотим провести разведочный анализ данных, чтобы понять, какие продукты пользуются наибольшим спросом и какие факторы влияют на продажи.

1. Импортируем библиотеки и загружаем датасет:

```
python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('sales_data.csv')
```

2. Проверяем данные на наличие пропущенных значений и выбросов:

```
python
# проверяем на пропущенные значения
print(df.isnull().sum())

# проверяем на выбросы
sns.boxplot(x=df['sales'])
```

3. Изучаем распределение продаж по продуктам:

```
python
# группируем данные по продуктам и сумме продаж
product_sales =
df.groupby('product')['sales'].sum().reset_index()

# сортируем по убыванию продаж
product_sales = product_sales.sort_values(by='sales',
ascending=False)

# строим график
plt.figure(figsize=(10, 5))
sns.barplot(x='product', y='sales', data=product_sales)
plt.title('Total Sales by Product')
plt.xlabel('Product')
plt.ylabel('Sales')
plt.show()
```

4. Изучаем влияние цены на продажи:

```
python
# строим график зависимости цены и продаж
sns.scatterplot(x='price', y='sales', data=df)
plt.title('Price vs Sales')
plt.xlabel('Price')
plt.ylabel('Sales')
plt.show()

# строим график распределения цен
sns.distplot(df['price'])
plt.title('Price Distribution')
plt.xlabel('Price')
plt.show()
```

5. Изучаем влияние рекламы на продажи:

```
python
# группируем данные по типу рекламы и сумме продаж
ad_sales =
df.groupby('ad_type')['sales'].sum().reset_index()

# сортируем по убыванию продаж
ad_sales = ad_sales.sort_values(by='sales', ascending=False)

# строим график
plt.figure(figsize=(10, 5))
sns.barplot(x='ad_type', y='sales', data=ad_sales)
plt.title('Total Sales by Ad Type')
plt.xlabel('Ad Type')
plt.ylabel('Sales')
plt.show()
```


Это только некоторые примеры того, как можно провести разведочный анализ данных. Конкретные методы и подходы будут зависеть от конкретного датасета и задачи.

Практическое задание № 2. Задача бинарной классификации сбалансированного набора данных

1. Загрузить набор данных по анализу и прогнозированию вероятности инфаркта (<https://www.kaggle.com/datasets/rashikrahmanpritom/heart-attack-analysis-prediction-dataset?datasetId=1226038&sortBy=voteCount>) как датафрейм библиотеки pandas

2. Получить представление о наборе данных с помощью методов `shape`, `head`, `describe`, `info` библиотеки pandas. Посчитать количество пустых значений в признаках с помощью библиотеки `numpy`

3. Произвести разведочный анализ данных по данному набору данных – визуализировать данные с помощью библиотек `matplotlib`, `seaborn`, `plotly`. Построить:

- 1) Парные диаграммы
- 2) Тепловую карту по матрице корреляции
- 3) Гистограммы
- 4) Ящик с усами (диаграмма размаха)
- 5) Графики распределения

4. На основе выполненного анализа данных произвести выводы о влиянии различных признаков на вероятность инфаркта

5. Произвести предобработку данных – удалить из набора данных признаки, слабо коррелирующие между собой и с вероятностью инфаркта. Закодировать категориальные признаки в дискретные величины с помощью метода `get_dummies`, используя библиотеку `sklearn`. Произвести нормализацию данных с помощью метода `RobustScaler` библиотеки `sklearn`. Разбить обработанный набор данных на обучающую и тестовую выборки с помощью метода `train_test_split` библиотеки `sklearn`

6. Произвести обучение следующих моделей библиотеки `sklearn`:

- 1) Метод опорных векторов (Support vector machine)
- 2) Логистическая регрессия (LogisticRegression)
- 3) Дерево решений (Decision Tree)
- 4) Случайный лес (RandomForest)

7. Отобразить корректность работы каждой модели на тестовой и обучающей выборках с помощью метрики `accuracy` библиотеки `sklearn`

Пример

Допустим, у нас есть датасет с информацией о покупателях интернет-магазина, где каждый покупатель может быть отнесен к одному из двух классов: "клиент" или "не клиент". Наша задача – построить модель бинарной

классификации, которая сможет предсказывать, является ли новый покупатель клиентом или нет.

1. Импортируем библиотеки и загружаем датасет:

```
python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix

df = pd.read_csv('customer_data.csv')
```

2. Проверяем данные на наличие пропущенных значений и выбросов:

```
python
# проверяем на пропущенные значения
print(df.isnull().sum())

# проверяем на выбросы
sns.boxplot(x=df['age'])
```

3. Разделяем данные на тренировочный и тестовый наборы:

```
python
# разделяем данные на признаки и целевую переменную
X = df.drop('is_customer', axis=1)
y = df['is_customer']

# разделяем данные на тренировочный и тестовый наборы
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

4. Обучаем модель логистической регрессии:

```
python
# создаем экземпляр модели
model = LogisticRegression()

# обучаем модель на тренировочных данных
model.fit(X_train, y_train)
```

5. Предсказываем значения на тестовом наборе и оцениваем точность модели:

```
python
# делаем предсказания на тестовых данных
y_pred = model.predict(X_test)
```

```

# оцениваем точность модели
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy:', accuracy)

# строим матрицу ошибок
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

```

Это только общий пример того, как может выглядеть выполнение задачи бинарной классификации. Конкретные методы и подходы будут зависеть от конкретного датасета и задачи. Важно также учитывать сбалансированность набора данных и применять соответствующие методы для решения этой проблемы.

Практическое задание № 3. Задача многоклассовой классификации сбалансированного набора данных

1. Загрузить набор данных по классификации цветов ириса (<https://www.kaggle.com/datasets/arshid/iris-flower-dataset?datasetId=17860&sortBy=voteCount>) как датафрейм библиотеки pandas

2. Получить представления о наборе данных с помощью методов shape, head, describe, info библиотеки pandas. Посчитать количество пустых значений в признаках с помощью библиотеки numpy

3. Произвести разведочный анализ данных по данному набору данных – визуализировать данные с помощью библиотек matplotlib, seaborn, plotly. Построить:

- 1) Парные диаграммы
- 2) Гистограммы
- 3) Ящик с усами (диаграмма размаха)

4. На основе выполненного анализа данных произвести выводы о зависимости типа цветов ириса на его физические параметры

5. Разбить обработанный набор данных на обучающую и тестовую выборки с помощью метода train_test_split библиотеки sklearn

6. Произвести обучение следующих моделей библиотеки sklearn:

- 1) Метод опорных векторов (Support vector machine)
- 2) Логистическая регрессия (LogisticRegression)
- 3) Дерево решений (Decision Tree)
- 4) Наивный Байес (Naive Bayes)

7. Отобразить точность работы каждой модели на тестовой и обучающей выборках с помощью метрики accuracy библиотеки sklearn.

Пример.

Допустим, у нас есть датасет с информацией о различных культурах, выращиваемых в сельском хозяйстве, где каждая культура может быть отнесена к одной из нескольких категорий: "зерновые", "овощи", "фрукты" и "ягоды". Наша задача – построить модель многоклассовой классификации, которая сможет предсказывать категорию новой культуры.

1. Импортируем библиотеки и загружаем датасет:

```
python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix

df = pd.read_csv('agriculture_data.csv')
```

2. Проверяем данные на наличие пропущенных значений и выбросов:

```
python
# проверяем на пропущенные значения
print(df.isnull().sum())

# проверяем на выбросы
sns.boxplot(x=df['yield'])
```

3. Разделяем данные на тренировочный и тестовый наборы:

```
python
# разделяем данные на признаки и целевую переменную
X = df.drop('category', axis=1)
y = df['category']

# разделяем данные на тренировочный и тестовый наборы
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

4. Обучаем модель логистической регрессии:

```
python
# создаем экземпляр модели
model = LogisticRegression(multi_class='multinomial',
solver='lbfgs')

# обучаем модель на тренировочных данных
model.fit(X_train, y_train)
```

5. Предсказываем значения на тестовом наборе и оцениваем точность модели:

```
python
# делаем предсказания на тестовых данных
y_pred = model.predict(X_test)

# оцениваем точность модели
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy:', accuracy)

# строим матрицу ошибок
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

6. Интерпретируем результаты и улучшаем модель при необходимости.

В данном случае мы использовали метод многоклассовой классификации "multinomial" и алгоритм оптимизации "lbfgs". Важно также учитывать сбалансированность набора данных и применять соответствующие методы для решения этой проблемы, например, использование метода взвешенной классификации. Также можно проводить дополнительный анализ данных, чтобы выявить взаимосвязи между признаками и целевой переменной, и использовать эти знания для улучшения модели.

Практическое задание № 4. Задача классификации несбалансированного набора данных

1. Загрузить набор данных по обнаружению мошенничества с кредитными картами (<https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud>) как датафрейм библиотеки pandas

2. Получить представления о наборе данных с помощью методов shape, head, describe, info библиотеки pandas. Посчитать количество пустых значений в признаках с помощью библиотеки numpy

3. На основе выполненного анализа данных произвести выводы о структуре набора данных

4. Произвести скалярную стандартизацию данных. Разбить обработанный набор данных на обучающую и тестовую выборки с помощью метода train_test_split библиотеки sklearn

5. Произвести обучение следующих моделей библиотеки sklearn:

1) Логистическая регрессия (Logistic Regression)

6. Отобразить точность работы каждой модели на тестовой и обучающей выборках с помощью метрик precision, recall, f1-score и ROC-кривой библиотеки sklearn

7. Повторно отобразить метрики после проведения кросс-валидации.

Пример

Допустим, у нас есть датасет с информацией о различных культурах, выращиваемых в сельском хозяйстве, где каждая культура может быть отнесена к одной из двух категорий: "ценные" и "невыгодные". Однако, в нашем наборе данных количество культур в категории "ценные" значительно меньше, чем в категории "невыгодные". Наша задача – построить модель бинарной классификации, которая сможет предсказывать категорию новой культуры.

1. Импортируем библиотеки и загружаем датасет:

```
python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix

df = pd.read_csv('agriculture_data.csv')
```

2. Проверяем данные на наличие пропущенных значений и выбросов:

```
python
# проверяем на пропущенные значения
print(df.isnull().sum())

# проверяем на выбросы
sns.boxplot(x=df['yield'])
```

3. Разделяем данные на тренировочный и тестовый наборы:

```
python
# разделяем данные на признаки и целевую переменную
X = df.drop('category', axis=1)
y = df['category']

# разделяем данные на тренировочный и тестовый наборы
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

4. Применяем метод взвешенной классификации:

```
python
# создаем экземпляр модели с учетом взвешенности классов
```

```
model = LogisticRegression(class_weight='balanced')

# обучаем модель на тренировочных данных
model.fit(X_train, y_train)
```

5. Предсказываем значения на тестовом наборе и оцениваем точность модели:

```
python
# делаем предсказания на тестовых данных
y_pred = model.predict(X_test)

# оцениваем точность модели
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy:', accuracy)

# строим матрицу ошибок
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

6. Интерпретируем результаты и улучшаем модель при необходимости.

В данном случае мы использовали метод взвешенной классификации, чтобы учесть несбалансированность набора данных. Важно также проводить дополнительный анализ данных, чтобы выявить взаимосвязи между признаками и целевой переменной, и использовать эти знания для улучшения модели. Также можно попробовать использовать другие методы балансировки классов, например, метод *oversampling* или *undersampling*.

Практическое задание № 5. Задача восстановления регрессии

1. Загрузить набор данных по прогнозированию стоимости недвижимости

(<https://www.kaggle.com/datasets/harlfoxem/housesalesprediction?datasetId=128&sortBy=voteCount>) как датафрейм библиотеки *pandas*

2. Получить представления о наборе данных с помощью методов *shape*, *head*, *describe*, *info* библиотеки *pandas*. Посчитать количество пустых значений в признаках с помощью библиотеки *numpy*

3. Произвести разведочный анализ данных по данному набору данных – визуализировать данные с помощью библиотек *matplotlib*, *seaborn*, *plotly*. Построить:

- 1) Гистограммы
- 2) Ящик с усами (диаграмма размаха)

- 3) Точечные 3D-графики распределения признаков
- 4) Тепловая карта корреляционной матрицы
4. На основе выполненного анализа данных произвести выводы о влиянии различных характеристик на стоимость недвижимости
5. Объединить значения по периодам в признаках даты постройки и даты последнего капитального ремонта. Произвести нормализацию данных. Разбить обработанный набор данных на обучающую и тестовую выборки с помощью метода `train_test_split` библиотеки `sklearn`
6. Произвести обучение следующих моделей библиотеки `sklearn`:
 - 1) Линейная регрессия (Linear Regression)
 - 2) Регрессия LASSO (Lasso Regression)
 - 3) Регрессия Ridge (Ridge Regression)
 - 4) Полиномиальная регрессия (Polynomial Regression)
7. Отобразить точность работы каждой модели на тестовой и обучающей выборках с помощью метрик `r_squared` (p-квадрат, коэффициент детерминации), `mean absolute error` (средняя абсолютная ошибка), `mean squared error` (средняя квадратичная ошибка) библиотеки `sklearn`
8. Повторно отобразить метрики после проведения кросс-валидации
9. Произвести поиск оптимальных гиперпараметров с помощью метода `GridSearchCV` по модели с наилучшими показателями.

Пример

Допустим, у нас есть датасет с информацией о различных культурах, выращиваемых в сельском хозяйстве, где каждая культура характеризуется набором признаков (например, тип почвы, количество осадков, температура и т.д.) и целевой переменной – урожайность данной культуры. Наша задача – построить модель регрессии, которая сможет предсказывать урожайность новой культуры на основе ее признаков.

1. Импортируем библиотеки и загружаем датасет:

```
python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

df = pd.read_csv('agriculture_data.csv')
```

2. Проверяем данные на наличие пропущенных значений и выбросов:

```
python
# проверяем на пропущенные значения
print(df.isnull().sum())
```



```
# проверяем на выбросы
sns.boxplot(x=df['yield'])
```

3. Разделяем данные на тренировочный и тестовый наборы:

```
python
# разделяем данные на признаки и целевую переменную
X = df.drop('yield', axis=1)
y = df['yield']

# разделяем данные на тренировочный и тестовый наборы
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

4. Обучаем модель регрессии:

```
python
# создаем экземпляр модели
model = LinearRegression()

# обучаем модель на тренировочных данных
model.fit(X_train, y_train)
```

5. Предсказываем значения на тестовом наборе и оцениваем точность модели:

```
python
# делаем предсказания на тестовых данных
y_pred = model.predict(X_test)

# оцениваем точность модели
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print('Mean Squared Error:', mse)
print('R2 Score:', r2)
```

6. Интерпретируем результаты и улучшаем модель при необходимости.

В данном случае мы использовали модель линейной регрессии для предсказания урожайности на основе признаков культуры. Важно также проводить дополнительный анализ данных, чтобы выявить взаимосвязи между признаками и целевой переменной, и использовать эти знания для улучшения модели. Также можно попробовать использовать другие модели регрессии, например, модель случайного леса или градиентного бустинга.

Практическое задание № 6. Задача кластеризации

1. Загрузить набор данных сегментации покупателей по индивидуальным особенностям (<https://www.kaggle.com/datasets/imakash3011/customer-personality-analysis>) как датафрейм библиотеки pandas

2. Получить представления о наборе данных с помощью методов `shape`, `head`, `describe`, `info` библиотеки pandas. Посчитать количество пустых значений в признаках с помощью библиотеки `numpy`

3. Произвести разведочный анализ данных по данному набору данных – визуализировать данные с помощью библиотек `matplotlib`, `seaborn`, `plotly`. Построить:

1) Графики распределения признаков

2) Тепловая карта корреляционной матрицы

4. На основе выполненного анализа данных произвести выводы о распределении индивидуальных качеств покупателей

5. Закодировать категориальные данные в дискретные величины, произвести скалярную стандартизацию данных

6. Произвести снижение размерности набора данных

7. Произвести обучение следующих моделей библиотеки `sklearn`:

1) Метод k-средних (`k-means clustering`)

8. Произвести разведочный анализ данных по кластерам для оценки качества обучения модели.

Пример.

Допустим, у нас есть датасет с информацией о различных фермерских хозяйствах, где каждое хозяйство характеризуется набором признаков (например, тип почвы, используемые удобрения, типы выращиваемых культур и т.д.). Наша задача – провести кластеризацию хозяйств на группы схожих по признакам хозяйств.

1. Импортируем библиотеки и загружаем датасет:

```
python
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

df = pd.read_csv('farm_data.csv')
```

2. Проверяем данные на наличие пропущенных значений и выбросов:

```
python
# проверяем на пропущенные значения
print(df.isnull().sum())
```

```
# проверяем на выбросы
for col in df.columns:
    sns.boxplot(x=df[col])
```

3. Стандартизируем данные:

```
python
# стандартизируем данные
scaler = StandardScaler()
X = scaler.fit_transform(df)
```

4. Определяем оптимальное количество кластеров с помощью метода "локтя":

```
python
# определяем оптимальное количество кластеров
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++',
random_state=42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```

5. Обучаем модель кластеризации:

```
python
# создаем экземпляр модели
kmeans = KMeans(n_clusters=3, init='k-means++',
random_state=42)

# обучаем модель на стандартизированных данных
kmeans.fit(X)
```

6. Предсказываем кластеры для каждого хозяйства и визуализируем результаты:

```
python
# предсказываем кластеры для каждого хозяйства
labels = kmeans.predict(X)

# визуализируем результаты
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis')
plt.title('Clusters of Farms')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.show()
```

7. Интерпретируем результаты и улучшаем модель при необходимости.

В данном случае мы использовали модель кластеризации KMeans для разделения хозяйств на группы схожих по признакам. Важно также проводить дополнительный анализ данных, чтобы выявить взаимосвязи между признаками и кластерами, и использовать эти знания для улучшения модели. Также можно попробовать использовать другие методы кластеризации, например, иерархическую кластеризацию или DBSCAN.

Практическое задание № 7. Задача выявления аномалий

1. Загрузить набор данных выявления аномалий (<https://www.kaggle.com/datasets/boltzmannbrain/nab?datasetId=110&sortBy=voteCount>) как датафрейм библиотеки pandas

2. Получить представления о наборе данных с помощью методов shape, head, describe, info библиотеки pandas. Посчитать количество пустых значений в признаках с помощью библиотеки numpy

3. Произвести разведочный анализ данных по данному набору данных – визуализировать данные с помощью библиотек matplotlib, seaborn, plotly. Построить:

1) Гистограммы распределения данных

2) Графики распределения данных

4. На основе выполненного анализа данных произвести выводы об особенностях набора данных

5. Провести кластеризацию данных

6. Произвести обучение следующих моделей библиотеки sklearn:

1) Метод k-средних (k-means clustering)

2) Цепь Маркова (Markov Chain)

3) Изолирующий лес (Isolation forest)

4) Одноклассовый метод опорных векторов (One class SVM)

7. Произвести разведочный анализ данных по набору данных для оценки качества обучения модели.

Пример

Допустим, у нас есть датасет с информацией о производстве молока в различных фермерских хозяйствах, где каждое хозяйство характеризуется набором признаков (например, количество коров, уровень продуктивности, тип кормления и т.д.). Наша задача – выявить аномалии в производстве молока и определить причины их возникновения.

1. Импортируем библиотеки и загружаем датасет:

```
python
import pandas as pd
import numpy as np
from sklearn.ensemble import IsolationForest
```

```
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

df = pd.read_csv('milk_production.csv')
```

2. Проверяем данные на наличие пропущенных значений и выбросов:

```
python
# проверяем на пропущенные значения
print(df.isnull().sum())

# проверяем на выбросы
for col in df.columns:
    sns.boxplot(x=df[col])
```

3. Стандартизируем данные:

```
python
# стандартизируем данные
scaler = StandardScaler()
X = scaler.fit_transform(df)
```

4. Обучаем модель выявления аномалий с помощью Isolation Forest:

```
python
# создаем экземпляр модели
clf = IsolationForest(random_state=42)

# обучаем модель на стандартизированных данных
clf.fit(X)
```

5. Предсказываем аномалии и визуализируем результаты:

```
python
# предсказываем аномалии для каждого хозяйства
labels = clf.predict(X)

# визуализируем результаты
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis')
plt.title('Anomalies in Milk Production')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.show()
```

6. Анализируем результаты и определяем причины возникновения аномалий. Можно также использовать другие методы выявления аномалий, например, Local Outlier Factor или One-Class SVM.

В данном случае мы использовали модель Isolation Forest для выявления аномалий в производстве молока. Важно также проводить дополнительный

анализ данных, чтобы выявить причины возникновения аномалий и принять меры для их устранения.

Практическое задание № 8. Задача прогнозирования временных рядов

1. Загрузить набор данных прогнозирования погодных условий (<https://www.kaggle.com/datasets/sumanthvrao/daily-climate-time-series-data?datasetId=312121&sortBy=voteCount>) как датафрейм библиотеки pandas

2. Получить представления о наборе данных с помощью методов `shape`, `head`, `describe`, `info` библиотеки pandas. Посчитать количество пустых значений в признаках с помощью библиотеки `numpy`

3. Произвести разведочный анализ данных по данному набору данных – визуализировать данные с помощью библиотек `matplotlib`, `seaborn`, `plotly`. Построить:

1) Гистограммы распределения данных

2) Графики распределения данных

3) Ящики с усами (диаграммы размаха)

4. На основе выполненного анализа данных произвести выводы об особенностях набора данных

5. Провести кластеризацию данных

6. Произвести обучение следующих моделей библиотеки `sklearn`:

1) `XGBoost Regressor`

7. Отобразить точность работы модели на тестовой и обучающей выборках с помощью метрики `r squared` (r-квадрат, коэффициент детерминации) библиотеки `sklearn`.

Допустим, у нас есть датасет с информацией о производстве зерновых культур в различных регионах за последние 10 лет. Наша задача – прогнозировать производство зерновых культур в следующем году для каждого региона.

1. Импортируем библиотеки и загружаем датасет:

```
python
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt

df = pd.read_csv('grain_production.csv')
```

2. Проверяем данные на наличие пропущенных значений и выбросов:

```
python
# проверяем на пропущенные значения
print(df.isnull().sum())

# проверяем на выбросы
for col in df.columns:
    sns.boxplot(x=df[col])
```

3. Разделяем данные на обучающую и тестовую выборки:

```
python
# разделяем данные на признаки и целевую переменную
X = df.drop('production', axis=1)
y = df['production']

# разделяем данные на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

4. Обучаем модель линейной регрессии:

```
python
# создаем экземпляр модели
model = LinearRegression()

# обучаем модель на обучающих данных
model.fit(X_train, y_train)
```

5. Предсказываем значения для тестовой выборки и оцениваем качество модели:

```
python
# предсказываем значения для тестовой выборки
y_pred = model.predict(X_test)

# оцениваем качество модели
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
print('RMSE:', rmse)
```

6. Применяем модель для прогнозирования производства зерновых культур в следующем году:

```
python
# создаем датасет с данными за последние 10 лет
last_10_years = df.iloc[-10: , :-1]
```

```
# предсказываем производство зерновых культур для следующего
года
next_year_production = model.predict(last_10_years)

# визуализируем результаты
plt.plot(next_year_production)
plt.title('Grain Production Forecast for Next Year')
plt.xlabel('Region')
plt.ylabel('Production')
plt.show()
```

7. Анализируем результаты и принимаем решения по улучшению производства зерновых культур в каждом регионе. Можно также использовать другие методы прогнозирования временных рядов, например, ARIMA или Prophet.

Рекомендуемая литература

1. Москвитин, А.А. Данные, информация, знания: методология, теория, технологии : монография / А.А. Москвитин. — Санкт-Петербург : Лань, 2019 — 236 с. — ISBN 978-5-8114-3232-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/113937> — Режим доступа: для авториз. пользователей.

2. Макшанов, А.В. Технологии интеллектуального анализа данных : учебное пособие / А.В. Макшанов, А.Е. Журавлев. — 2-е изд., стер. — Санкт-Петербург : Лань, 2019 — 212 с. — ISBN 978-5-8114-4493-9. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/120063> — Режим доступа: для авториз. пользователей.

3. Остроух, А.В. Системы искусственного интеллекта : монография / А.В. Остроух, Н.Е. Суркова. — Санкт-Петербург : Лань, 2019 — 228 с. — ISBN 978-5-8114-3427-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/11340> — Режим доступа: для авториз. пользователей.

4. Остроух, А.В. Интеллектуальные информационные системы и технологии : монография / А.В. Остроух, А.Б. Николаев. — Санкт-Петербург : Лань, 2019 — 308 с. — ISBN 978-5-8114-3409-1. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/115518> — Режим доступа: для авториз. пользователей.