

Тема: Введение в машинное обучение

Сергей Витальевич Рыбин
svrybin@etu.ru

СПбГЭТУ «ЛЭТИ», кафедра «Алгоритмической математики»

15 октября 2023 г.



СПбГЭТУ «ЛЭТИ»
ПЕРВЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ

Термин «машинное обучение»

Его ввел в 1959 году Артур Самуэль. Он изобрел первую самообучающуюся компьютерную программу по игре в шашки. Машинное обучение он определил как процесс, в результате которого компьютеры способны показать такое поведение, которое в них не было запрограммировано изначально.

Термин «машинное обучение»

Его ввел в 1959 году Артур Самуэль. Он изобрел первую самообучающуюся компьютерную программу по игре в шашки. Машинное обучение он определил как процесс, в результате которого компьютеры способны показать такое поведение, которое в них не было запрограммировано изначально.

Этапы развития

Термин «машинное обучение»

Его ввел в 1959 году Артур Самуэль. Он изобрел первую самообучающуюся компьютерную программу по игре в шашки. Машинное обучение он определил как процесс, в результате которого компьютеры способны показать такое поведение, которое в них не было запрограммировано изначально.

Этапы развития

- 1 1950 год. Алан Тьюринг создал *«тест Тьюринга»* для оценки интеллекта компьютера. Тест определял способность компьютера мыслить подобно человеку.

Термин «машинное обучение»

Его ввел в 1959 году Артур Самуэль. Он изобрел первую самообучающуюся компьютерную программу по игре в шашки. Машинное обучение он определил как процесс, в результате которого компьютеры способны показать такое поведение, которое в них не было запрограммировано изначально.

Этапы развития

- 1 1950 год. Алан Тьюринг создал *«тест Тьюринга»* для оценки интеллекта компьютера. Тест определял способность компьютера мыслить подобно человеку.
- 2 1951 год. Эвелином Фиксом и Джозефом Ходжесом был разработан метрический алгоритм классификации *«метод k ближайших соседей»* (*k-nearest neighbors algorithm* — *KNN*). Алгоритм позволил компьютерам использовать простые шаблоны распознавания.

Термин «машинное обучение»

Его ввел в 1959 году Артур Самуэль. Он изобрел первую самообучающуюся компьютерную программу по игре в шашки. Машинное обучение он определил как процесс, в результате которого компьютеры способны показать такое поведение, которое в них не было запрограммировано изначально.

Этапы развития

- 1 1950 год. Алан Тьюринг создал *«тест Тьюринга»* для оценки интеллекта компьютера. Тест определял способность компьютера мыслить подобно человеку.
- 2 1951 год. Эвелином Фиксом и Джозефом Ходжесом был разработан метрический алгоритм классификации *«метод k ближайших соседей»* (*k-nearest neighbors algorithm* — *KNN*). Алгоритм позволил компьютерам использовать простые шаблоны распознавания.
- 3 1952 год. Артур Самуэль создал первую шашечную программу для IBM 701.

Термин «машинное обучение»

Его ввел в 1959 году Артур Самуэль. Он изобрел первую самообучающуюся компьютерную программу по игре в шашки. Машинное обучение он определил как процесс, в результате которого компьютеры способны показать такое поведение, которое в них не было запрограммировано изначально.

Этапы развития

- 1 1950 год. Алан Тьюринг создал *«тест Тьюринга»* для оценки интеллекта компьютера. Тест определял способность компьютера мыслить подобно человеку.
- 2 1951 год. Эвелином Фиксом и Джозефом Ходжесом был разработан метрический алгоритм классификации *«метод k ближайших соседей»* (*k-nearest neighbors algorithm* — *KNN*). Алгоритм позволил компьютерам использовать простые шаблоны распознавания.
- 3 1952 год. Артур Самуэль создал первую шашечную программу для IBM 701.
- 4 1958 год. Фрэнк Розенблатт предложил первую искусственную нейронную сеть — *«Перцептрон»* и реализовал ее в виде нейрокомпьютера *«Марк-1»*. Издание New York Times назвало его *«эмбрионом электронного компьютера, который в будущем сможет ходить, говорить, видеть, писать, воспроизводить себя и осознавать свое существование»*.

Термин «машинное обучение»

Его ввел в 1959 году Артур Самуэль. Он изобрел первую самообучающуюся компьютерную программу по игре в шашки. Машинное обучение он определил как процесс, в результате которого компьютеры способны показать такое поведение, которое в них не было запрограммировано изначально.

Этапы развития

- 1 1950 год. Алан Тьюринг создал *«тест Тьюринга»* для оценки интеллекта компьютера. Тест определял способность компьютера мыслить подобно человеку.
- 2 1951 год. Эвелином Фиксом и Джозефом Ходжесом был разработан метрический алгоритм классификации *«метод k ближайших соседей»* (*k-nearest neighbors algorithm* — *KNN*). Алгоритм позволил компьютерам использовать простые шаблоны распознавания.
- 3 1952 год. Артур Самуэль создал первую шашечную программу для IBM 701.
- 4 1958 год. Фрэнк Розенblatt предложил первую искусственную нейронную сеть — *«Перцептрон»* и реализовал ее в виде нейрокомпьютера *«Марк-1»*. Издание New York Times назвало его *«эмбрионом электронного компьютера, который в будущем сможет ходить, говорить, видеть, писать, воспроизводить себя и осознавать свое существование»*.
- 5 1959 год. Марвин Минский создал первую машину *«SNARC»* со случайно связанной нейросетью.

Термин «машинное обучение»

Его ввел в 1959 году Артур Самуэль. Он изобрел первую самообучающуюся компьютерную программу по игре в шашки. Машинное обучение он определил как процесс, в результате которого компьютеры способны показать такое поведение, которое в них не было запрограммировано изначально.

Этапы развития

- 1 1950 год. Алан Тьюринг создал *«тест Тьюринга»* для оценки интеллекта компьютера. Тест определял способность компьютера мыслить подобно человеку.
- 2 1951 год. Эвелином Фиксом и Джозефом Ходжесом был разработан метрический алгоритм классификации *«метод k ближайших соседей»* (*k-nearest neighbors algorithm* — *KNN*). Алгоритм позволил компьютерам использовать простые шаблоны распознавания.
- 3 1952 год. Артур Самуэль создал первую шашечную программу для IBM 701.
- 4 1958 год. Фрэнк Розенблатт предложил первую искусственную нейронную сеть — *«Перцептрон»* и реализовал ее в виде нейрокомпьютера *«Марк-1»*. Издание New York Times назвало его *«эмбрионом электронного компьютера, который в будущем сможет ходить, говорить, видеть, писать, воспроизводить себя и осознавать свое существование»*.
- 5 1959 год. Марвин Минский создал первую машину *«SNARC»* со случайно связанной нейросетью.
- 6 1985 год. Терри Сейновски создал *«NetTalk»* — искусственную нейронную сеть.

Термин «машинное обучение»

Его ввел в 1959 году Артур Самуэль. Он изобрел первую самообучающуюся компьютерную программу по игре в шашки. Машинное обучение он определил как процесс, в результате которого компьютеры способны показать такое поведение, которое в них не было запрограммировано изначально.

Этапы развития

- 1 1950 год. Алан Тьюринг создал *«тест Тьюринга»* для оценки интеллекта компьютера. Тест определял способность компьютера мыслить подобно человеку.
- 2 1951 год. Эвелином Фиксом и Джозефом Ходжесом был разработан метрический алгоритм классификации *«метод k ближайших соседей»* (*k-nearest neighbors algorithm* — *KNN*). Алгоритм позволил компьютерам использовать простые шаблоны распознавания.
- 3 1952 год. Артур Самуэль создал первую шашечную программу для IBM 701.
- 4 1958 год. Фрэнк Розенблатт предложил первую искусственную нейронную сеть — *«Перцептрон»* и реализовал ее в виде нейрокомпьютера *«Марк-1»*. Издание New York Times назвало его *«эмбрионом электронного компьютера, который в будущем сможет ходить, говорить, видеть, писать, воспроизводить себя и осознавать свое существование»*.
- 5 1959 год. Марвин Минский создал первую машину *«SNARC»* со случайно связанной нейросетью.
- 6 1985 год. Терри Сейновски создал *«NetTalk»* — искусственную нейронную сеть.
- 7 1997 год. Компьютер *«Deep Blue»* обыграл чемпиона мира, Гарри Каспарова, в шахматы.

Термин «машинное обучение»

Его ввел в 1959 году Артур Самуэль. Он изобрел первую самообучающуюся компьютерную программу по игре в шашки. Машинное обучение он определил как процесс, в результате которого компьютеры способны показать такое поведение, которое в них не было запрограммировано изначально.

Этапы развития

- 1 1950 год. Алан Тьюринг создал *«тест Тьюринга»* для оценки интеллекта компьютера. Тест определял способность компьютера мыслить подобно человеку.
- 2 1951 год. Эвелином Фиксом и Джозефом Ходжесом был разработан метрический алгоритм классификации *«метод k ближайших соседей»* (*k-nearest neighbors algorithm* — *KNN*). Алгоритм позволил компьютерам использовать простые шаблоны распознавания.
- 3 1952 год. Артур Самуэль создал первую шашечную программу для IBM 701.
- 4 1958 год. Фрэнк Розенблатт предложил первую искусственную нейронную сеть — *«Перцептрон»* и реализовал ее в виде нейрокомпьютера *«Марк-1»*. Издание New York Times назвало его *«эмбрионом электронного компьютера, который в будущем сможет ходить, говорить, видеть, писать, воспроизводить себя и осознавать свое существование»*.
- 5 1959 год. Марвин Минский создал первую машину *«SNARC»* со случайно связанной нейросетью.
- 6 1985 год. Терри Сейновски создал *«NetTalk»* — искусственную нейронную сеть.
- 7 1997 год. Компьютер *«Deep Blue»* обыграл чемпиона мира, Гарри Каспарова, в шахматы.
- 8 2006 год. Джеффри Хинтон ввел термин *«глубокое обучение»* (*deep learning*);

Что такое машинное обучение

Машинное обучение (Machine Learning) — это разработка алгоритмов, позволяющие компьютеру путем обобщения заданных примеров находить неявные закономерности в сложных параметрических задачах, не следуя жестко заданным правилам.

Суть машинного обучения — поиск скрытых связей между данными, называемых *латентными связями*.

Что такое машинное обучение

Машинное обучение (Machine Learning) — это разработка алгоритмов, позволяющие компьютеру путем обобщения заданных примеров находить неявные закономерности в сложных параметрических задачах, не следуя жестко заданным правилам.

Суть машинного обучения — поиск скрытых связей между данными, называемых *латентными связями*.

Цель ML

Основная цель машинного обучения — смоделировать реальную задачу с помощью некоторой, заранее неизвестной, математической функции, которая определяется на основе анализа данных. То есть обучение есть связывание входных данных с выходными данными.

Что такое машинное обучение

Машинное обучение (Machine Learning) — это разработка алгоритмов, позволяющие компьютеру путем обобщения заданных примеров находить неявные закономерности в сложных параметрических задачах, не следуя жестко заданным правилам.

Суть машинного обучения — поиск скрытых связей между данными, называемых *латентными связями*.

Цель ML

Основная цель машинного обучения — смоделировать реальную задачу с помощью некоторой, заранее неизвестной, математической функции, которая определяется на основе анализа данных. То есть обучение есть связывание входных данных с выходными данными.

Основные компоненты ML

Что такое машинное обучение

Машинное обучение (Machine Learning) — это разработка алгоритмов, позволяющие компьютеру путем обобщения заданных примеров находить неявные закономерности в сложных параметрических задачах, не следуя жестко заданным правилам.

Суть машинного обучения — поиск скрытых связей между данными, называемых *латентными связями*.

Цель ML

Основная цель машинного обучения — смоделировать реальную задачу с помощью некоторой, заранее неизвестной, математической функции, которая определяется на основе анализа данных. То есть обучение есть связывание входных данных с выходными данными.

Основные компоненты ML

- 1 *Данные* От их объема и качества зависит эффективность и точность будущих результатов.

Что такое машинное обучение

Машинное обучение (Machine Learning) — это разработка алгоритмов, позволяющие компьютеру путем обобщения заданных примеров находить неявные закономерности в сложных параметрических задачах, не следуя жестко заданным правилам.

Суть машинного обучения — поиск скрытых связей между данными, называемых *латентными связями*.

Цель ML

Основная цель машинного обучения — смоделировать реальную задачу с помощью некоторой, заранее неизвестной, математической функции, которая определяется на основе анализа данных. То есть обучение есть связывание входных данных с выходными данными.

Основные компоненты ML

- 1 *Данные*. От их объема и качества зависит эффективность и точность будущих результатов.
- 2 *Признаки*. Определяют параметры, по которым будет строиться машинное обучение.

Что такое машинное обучение

Машинное обучение (Machine Learning) — это разработка алгоритмов, позволяющие компьютеру путем обобщения заданных примеров находить неявные закономерности в сложных параметрических задачах, не следуя жестко заданным правилам.

Суть машинного обучения — поиск скрытых связей между данными, называемых *латентными связями*.

Цель ML

Основная цель машинного обучения — смоделировать реальную задачу с помощью некоторой, заранее неизвестной, математической функции, которая определяется на основе анализа данных. То есть обучение есть связывание входных данных с выходными данными.

Основные компоненты ML

- 1 *Данные*. От их объема и качества зависит эффективность и точность будущих результатов.
- 2 *Признаки*. Определяют параметры, по которым будет строиться машинное обучение.
- 3 *Алгоритмы*. Выбор алгоритма (как и данных) будет влиять на точность, эффективность и качество.

Основные виды машинного обучения

Основные задачи, решаемых при помощи машинного обучения можно отнести к двум видам.

Основные виды машинного обучения

Основные задачи, решаемых при помощи машинного обучения можно отнести к двум видам.

- 1 *Обучение с учителем (supervised learning)*. Здесь входные данные заранее размечены. Необходимо построить математическую модель, которая соответствует входным данным.

Основные виды машинного обучения

Основные задачи, решаемых при помощи машинного обучения можно отнести к двум видам.

❶ *Обучение с учителем (supervised learning)*. Здесь входные данные заранее размечены. Необходимо построить математическую модель, которая соответствует входным данным.

Принцип обучения с учителем похоже на обучение ребенка родителями: "смотри, вот это слон, а это жираф". Машина будет учиться определять тип объекта на конкретных примерах.

Основные виды машинного обучения

Основные задачи, решаемых при помощи машинного обучения можно отнести к двум видам.

❶ *Обучение с учителем (supervised learning)*. Здесь входные данные заранее размечены. Необходимо построить математическую модель, которая соответствует входным данным.

Принцип обучения с учителем похож на обучение ребенка родителями: "смотри, вот это слон, а это жираф". Машина будет учиться определять тип объекта на конкретных примерах.

❷ Частный случай обучения с учителем — *обучение с частичным привлечением учителя (semi-supervised learning)*. Здесь используется небольшое количество размеченных данных и большое количество неразмеченных данных. В данном курсе этот подход не рассматривается.

Основные виды машинного обучения

Основные задачи, решаемых при помощи машинного обучения можно отнести к двум видам.

❶ *Обучение с учителем (supervised learning)*. Здесь входные данные заранее размечены. Необходимо построить математическую модель, которая соответствует входным данным.

Принцип обучения с учителем похоже на обучение ребенка родителями: "смотри, вот это слон, а это жираф". Машина будет учиться определять тип объекта на конкретных примерах.

❷ Частный случай обучения с учителем — *обучение с частичным привлечением учителя (semi-supervised learning)*. Здесь используется небольшое количество размеченных данных и большое количество неразмеченных данных. В данном курсе этот подход не рассматривается.

❸ *Обучение без учителя (unsupervised learning)*. Здесь входные данные не помечены, а результаты неизвестны.

Основные виды машинного обучения

Основные задачи, решаемых при помощи машинного обучения можно отнести к двум видам.

❶ *Обучение с учителем (supervised learning)*. Здесь входные данные заранее размечены. Необходимо построить математическую модель, которая соответствует входным данным.

Принцип обучения с учителем похож на обучение ребенка родителями: "смотри, вот это слон, а это жираф". Машина будет учиться определять тип объекта на конкретных примерах.

❷ Частный случай обучения с учителем — *обучение с частичным привлечением учителя (semi-supervised learning)*. Здесь используется небольшое количество размеченных данных и большое количество неразмеченных данных. В данном курсе этот подход не рассматривается.

❸ *Обучение без учителя (unsupervised learning)*. Здесь входные данные не помечены, а результаты неизвестны.

В обучении без учителя родители дают ребенку картинки слонов, жирафов..., и говорят: "сам разберись, кто на кого похож".

Основные понятия (продолжение)

Основные виды машинного обучения

Основные задачи, решаемых при помощи машинного обучения можно отнести к двум видам.

❶ *Обучение с учителем (supervised learning)*. Здесь входные данные заранее размечены. Необходимо построить математическую модель, которая соответствует входным данным.

Принцип обучения с учителем похоже на обучение ребенка родителями: "смотри, вот это слон, а это жираф". Машина будет учиться определять тип объекта на конкретных примерах.

❷ Частный случай обучения с учителем — *обучение с частичным привлечением учителя (semi-supervised learning)*. Здесь используется небольшое количество размеченных данных и большое количество неразмеченных данных. В данном курсе этот подход не рассматривается.

❸ *Обучение без учителя (unsupervised learning)*. Здесь входные данные не помечены, а результаты неизвестны.

В обучении без учителя родители дают ребенку картинки слонов, жирафов..., и говорят: "сам разберись, кто на кого похож".

Основные задачи ML, рассматриваемые в курсе

Основные понятия (продолжение)

Основные виды машинного обучения

Основные задачи, решаемых при помощи машинного обучения можно отнести к двум видам.

❶ *Обучение с учителем (supervised learning)*. Здесь входные данные заранее размечены. Необходимо построить математическую модель, которая соответствует входным данным.

Принцип обучения с учителем похоже на обучение ребенка родителями: "смотри, вот это слон, а это жираф". Машина будет учиться определять тип объекта на конкретных примерах.

❷ Частный случай обучения с учителем — *обучение с частичным привлечением учителя (semi-supervised learning)*. Здесь используется небольшое количество размеченных данных и большое количество неразмеченных данных. В данном курсе этот подход не рассматривается.

❸ *Обучение без учителя (unsupervised learning)*. Здесь входные данные не помечены, а результаты неизвестны.

В обучении без учителя родители дают ребенку картинки слонов, жирафов..., и говорят: "сам разберись, кто на кого похож".

Основные задачи ML, рассматриваемые в курсе

❶ *Уменьшение размерности*. Сведение большого числа признаков к меньшему.

Основные понятия (продолжение)

Основные виды машинного обучения

Основные задачи, решаемых при помощи машинного обучения можно отнести к двум видам.

1 *Обучение с учителем (supervised learning)*. Здесь входные данные заранее размечены. Необходимо построить математическую модель, которая соответствует входным данным.

Принцип обучения с учителем похож на обучение ребенка родителями: "смотри, вот это слон, а это жираф". Машина будет учиться определять тип объекта на конкретных примерах.

i 2 Частный случай обучения с учителем — *обучение с частичным привлечением учителя (semi-supervised learning)*. Здесь используется небольшое количество размеченных данных и большое количество неразмеченных данных. В данном курсе этот подход не рассматривается.

2 *Обучение без учителя (unsupervised learning)*. Здесь входные данные не помечены, а результаты неизвестны.

В обучении без учителя родители дают ребенку картинки слонов, жирафов..., и говорят: "сам разберись, кто на кого похож".

Основные задачи ML, рассматриваемые в курсе

1 *Уменьшение размерности*. Сведение большого числа признаков к меньшему.

2 *Кластеризация*. Распределение данных на группы (кластеры) по некоторому объединяющему признаку. Все объекты в кластере более похожи друг на друга, чем на объекты других кластеров. Кластеризация — это обучение без учителя, поскольку алгоритм сам определяет общие характеристики элементов в данных. Например, в задаче маркетинга можно сгруппировать людей со схожими запросами.

Основные понятия (продолжение)

Основные виды машинного обучения

Основные задачи, решаемых при помощи машинного обучения можно отнести к двум видам.

❶ *Обучение с учителем (supervised learning)*. Здесь входные данные заранее размечены. Необходимо построить математическую модель, которая соответствует входным данным.

Принцип обучения с учителем похож на обучение ребенка родителями: "смотри, вот это слон, а это жираф". Машина будет учиться определять тип объекта на конкретных примерах.

❷ Частный случай обучения с учителем — *обучение с частичным привлечением учителя (semi-supervised learning)*. Здесь используется небольшое количество размеченных данных и большое количество неразмеченных данных. В данном курсе этот подход не рассматривается.

❸ *Обучение без учителя (unsupervised learning)*. Здесь входные данные не помечены, а результаты неизвестны.

В обучении без учителя родители дают ребенку картинки слонов, жирафов..., и говорят: "сам разберись, кто на кого похож".

Основные задачи ML, рассматриваемые в курсе

❶ *Уменьшение размерности*. Сведение большого числа признаков к меньшему.

❷ *Кластеризация*. Распределение данных на группы (кластеры) по некоторому объединяющему признаку. Все объекты в кластере более похожи друг на друга, чем на объекты других кластеров. Кластеризация — это обучение без учителя, поскольку алгоритм сам определяет общие характеристики элементов в данных. Например, в задаче маркетинга можно сгруппировать людей со схожими запросами.

❸ *Классификация*. Есть некоторое подмножество объектов, разделенных на классы (*обучающая выборка*). Нужно определить к какому классу относится произвольный объект из исходного множества. Классификация — это обучение с учителем, поскольку классы определяются заранее и для обучающей выборки метки классов заданы.

Основные понятия (продолжение)

Основные виды машинного обучения

Основные задачи, решаемых при помощи машинного обучения можно отнести к двум видам.

1 *Обучение с учителем (supervised learning)*. Здесь входные данные заранее размечены. Необходимо построить математическую модель, которая соответствует входным данным.

Принцип обучения с учителем похож на обучение ребенка родителями: "смотри, вот это слон, а это жираф". Машина будет учиться определять тип объекта на конкретных примерах.

i 2 Частный случай обучения с учителем — *обучение с частичным привлечением учителя (semi-supervised learning)*. Здесь используется небольшое количество размеченных данных и большое количество неразмеченных данных. В данном курсе этот подход не рассматривается.

2 *Обучение без учителя (unsupervised learning)*. Здесь входные данные не помечены, а результаты неизвестны.

В обучении без учителя родители дают ребенку картинки слонов, жирафов..., и говорят: "сам разберись, кто на кого похож".

Основные задачи ML, рассматриваемые в курсе

1 *Уменьшение размерности*. Сведение большого числа признаков к меньшему.

2 *Кластеризация*. Распределение данных на группы (кластеры) по некоторому объединяющему признаку. Все объекты в кластере более похожи друг на друга, чем на объекты других кластеров. Кластеризация — это обучение без учителя, поскольку алгоритм сам определяет общие характеристики элементов в данных. Например, в задаче маркетинга можно сгруппировать людей со схожими запросами.

3 *Классификация*. Есть некоторое подмножество объектов, разделенных на классы (*обучающая выборка*). Нужно определить к какому классу относится произвольный объект из исходного множества. Классификация — это обучение с учителем, поскольку классы определяются заранее и для обучающей выборки метки классов заданы.

4 *Регрессия*. Анализ связи между несколькими независимыми переменными и зависимой переменной.

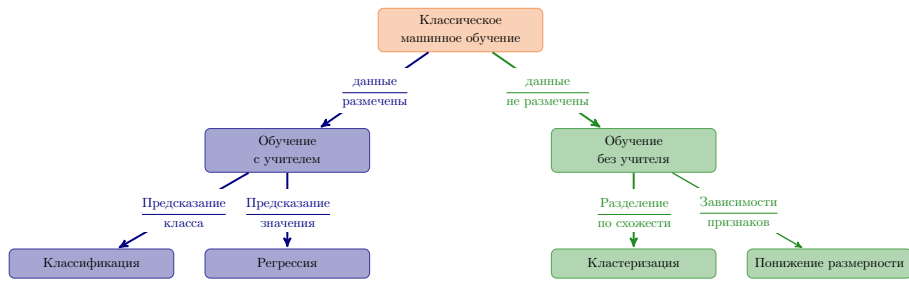


Рис. 1

Сбор данных

- 1 Сбор собственных баз данных. Собирать можно вручную (более трудоемко, но зато мало ошибок) или автоматически (гораздо быстрее, но с большим количеством ошибок). Крупные компании, например Google, иногда используют своих пользователей для бесплатной разметки данных.

Сбор данных

- 1 Сбор собственных баз данных. Собирать можно вручную (более трудоемко, но зато мало ошибок) или автоматически (гораздо быстрее, но с большим количеством ошибок). Крупные компании, например Google, иногда используют своих пользователей для бесплатной разметки данных.
- 2 Использование существующих баз данных.

Сбор данных

- 1 Сбор собственных баз данных. Собирать можно вручную (более трудоемко, но зато мало ошибок) или автоматически (гораздо быстрее, но с большим количеством ошибок). Крупные компании, например Google, иногда используют своих пользователей для бесплатной разметки данных.
- 2 Использование существующих баз данных.

Данные — нефть XXI века!

- 1 Хорошие наборы данных представляют большую коммерческую ценность. Крупные компании иногда раскрывают свои алгоритмы, но данные — крайне редко.

Сбор данных

- 1 Сбор собственных баз данных. Собирать можно вручную (более трудоемко, но зато мало ошибок) или автоматически (гораздо быстрее, но с большим количеством ошибок). Крупные компании, например Google, иногда используют своих пользователей для бесплатной разметки данных.
- 2 Использование существующих баз данных.

Данные — нефть XXI века!

- 1 Хорошие наборы данных представляют большую коммерческую ценность. Крупные компании иногда раскрывают свои алгоритмы, но данные — крайне редко.
- 2 Степень доверия к результатам машинного обучения прямо зависит от доверия к данным, на которых обучался алгоритм.

Сбор данных

- 1 Сбор собственных баз данных. Собирать можно вручную (более трудоемко, но зато мало ошибок) или автоматически (гораздо быстрее, но с большим количеством ошибок). Крупные компании, например Google, иногда используют своих пользователей для бесплатной разметки данных.
- 2 Использование существующих баз данных.

Данные — нефть XXI века!

- 1 Хорошие наборы данных представляют большую коммерческую ценность. Крупные компании иногда раскрывают свои алгоритмы, но данные — крайне редко.
- 2 Степень доверия к результатам машинного обучения прямо зависит от доверия к данным, на которых обучался алгоритм.
- 3 Данные для обучения должны быть репрезентативными. Если данные недостаточно точно соответствуют предметной области, то их обобщение будет неполным и полученная модель не даст хороших результатов.

Сбор данных

- 1 Сбор собственных баз данных. Собирать можно вручную (более трудоемко, но зато мало ошибок) или автоматически (гораздо быстрее, но с большим количеством ошибок). Крупные компании, например Google, иногда используют своих пользователей для бесплатной разметки данных.
- 2 Использование существующих баз данных.

Данные — нефть XXI века!

- 1 Хорошие наборы данных представляют большую коммерческую ценность. Крупные компании иногда раскрывают свои алгоритмы, но данные — крайне редко.
- 2 Степень доверия к результатам машинного обучения прямо зависит от доверия к данным, на которых обучался алгоритм.
- 3 Данные для обучения должны быть репрезентативными. Если данные недостаточно точно соответствуют предметной области, то их обобщение будет неполным и полученная модель не даст хороших результатов.

Предобработка данных

Преобразование данных для улучшения качества dataset — *«мусор на входе — мусор на выходе»*. На этом же этапе можно извлечь низкоуровневые признаки (Feature extraction). Данная процедура уникальна для каждого случая использования и набора данных.

Извлечение признаков

Извлечение признаков

Выделение признаков — это процесс сокращения исходного набора *«сырых»* переменных до более управляемых групп (признаков) для дальнейшей обработки, оставаясь при этом достаточным набором для точного и полного описания исходного набора данных.

Извлечение признаков

Выделение признаков — это процесс сокращения исходного набора *«сырых»* переменных до более управляемых групп (признаков) для дальнейшей обработки, оставаясь при этом достаточным набором для точного и полного описания исходного набора данных.

Требования к признакам:

Извлечение признаков

Выделение признаков — это процесс сокращения исходного набора «сырых» переменных до более управляемых групп (признаков) для дальнейшей обработки, оставаясь при этом достаточным набором для точного и полного описания исходного набора данных.

Требования к признакам:

- А Признаки должны отражать зависимости и закономерности предметной области.

Извлечение признаков

Выделение признаков — это процесс сокращения исходного набора «сырых» переменных до более управляемых групп (признаков) для дальнейшей обработки, оставаясь при этом достаточным набором для точного и полного описания исходного набора данных.

Требования к признакам:

- Ⓐ Признаки должны отражать зависимости и закономерности предметной области.
- Ⓑ Признаки не должны коррелировать, т.е. нести одну и ту же информацию (например, расстояния в километрах и милях).

Извлечение признаков

Выделение признаков — это процесс сокращения исходного набора «сырых» переменных до более управляемых групп (признаков) для дальнейшей обработки, оставаясь при этом достаточным набором для точного и полного описания исходного набора данных.

Требования к признакам:

- А Признаки должны отражать зависимости и закономерности предметной области.
- В Признаки не должны коррелировать, т.е. нести одну и ту же информацию (например, расстояния в километрах и милях).

Можно считать этот процесс снижением размерности. Условно его можно разбить на три этапа.

Извлечение признаков

Выделение признаков — это процесс сокращения исходного набора «сырых» переменных до более управляемых групп (признаков) для дальнейшей обработки, оставаясь при этом достаточным набором для точного и полного описания исходного набора данных.

Требования к признакам:

- Ⓐ Признаки должны отражать зависимости и закономерности предметной области.
- Ⓑ Признаки не должны коррелировать, т.е. нести одну и ту же информацию (например, расстояния в километрах и милях).

Можно считать этот процесс снижением размерности. Условно его можно разбить на три этапа.

- 1 **Извлекаем признаки** (feature extraction) — преобразовываем специфические представления для данной предметной области в числовые или категориальные векторы.

Извлечение признаков

Выделение признаков — это процесс сокращения исходного набора «сырых» переменных до более управляемых групп (признаков) для дальнейшей обработки, оставаясь при этом достаточным набором для точного и полного описания исходного набора данных.

Требования к признакам:

- А Признаки должны отражать зависимости и закономерности предметной области.
- В Признаки не должны коррелировать, т.е. нести одну и ту же информацию (например, расстояния в километрах и милях).

Можно считать этот процесс снижением размерности. Условно его можно разбить на три этапа.

- 1 **Извлекаем признаки** (feature extraction) — преобразовываем специфические представления для данной предметной области в числовые или категориальные векторы.
- 2 **Преобразуем признаки** (feature transformation) — изменяем данные для повышения точности алгоритма.

Извлечение признаков

Выделение признаков — это процесс сокращения исходного набора «сырых» переменных до более управляемых групп (признаков) для дальнейшей обработки, оставаясь при этом достаточным набором для точного и полного описания исходного набора данных.

Требования к признакам:

- А Признаки должны отражать зависимости и закономерности предметной области.
- В Признаки не должны коррелировать, т.е. нести одну и ту же информацию (например, расстояния в километрах и милях).

Можно считать этот процесс снижением размерности. Условно его можно разбить на три этапа.

- 1 **Извлекаем признаки** (feature extraction) — преобразовываем специфические представления для данной предметной области в числовые или категориальные векторы.
- 2 **Преобразуем признаки** (feature transformation) — изменяем данные для повышения точности алгоритма.
- 3 **Отбираем признаки** (feature selection) — отсекаем «ненужные» (малоинформативные) признаки. Подробнее методы выбора информативных признаков будут рассмотрены в теме **Регрессия**.

Извлечение признаков

Выделение признаков — это процесс сокращения исходного набора «сырых» переменных до более управляемых групп (признаков) для дальнейшей обработки, оставаясь при этом достаточным набором для точного и полного описания исходного набора данных.

Требования к признакам:

- Ⓐ Признаки должны отражать зависимости и закономерности предметной области.
- Ⓑ Признаки не должны коррелировать, т.е. нести одну и ту же информацию (например, расстояния в километрах и милях).

Можно считать этот процесс снижением размерности. Условно его можно разбить на три этапа.

- 1 **Извлекаем признаки** (feature extraction) — преобразовываем специфические представления для данной предметной области в числовые или категориальные векторы.
- 2 **Преобразуем признаки** (feature transformation) — изменяем данные для повышения точности алгоритма.
- 3 **Отбираем признаки** (feature selection) — отсекаем «ненужные» (малоинформативные) признаки. Подробнее методы выбора информативных признаков будут рассмотрены в теме **Регрессия**.

Примеры

Извлечение признаков

Выделение признаков — это процесс сокращения исходного набора «сырых» переменных до более управляемых групп (признаков) для дальнейшей обработки, оставаясь при этом достаточным набором для точного и полного описания исходного набора данных.

Требования к признакам:

- А Признаки должны отражать зависимости и закономерности предметной области.
- В Признаки не должны коррелировать, т.е. нести одну и ту же информацию (например, расстояния в километрах и милях).

Можно считать этот процесс снижением размерности. Условно его можно разбить на три этапа.

- 1 **Извлекаем признаки** (feature extraction) — преобразовываем специфические представления для данной предметной области в числовые или категориальные векторы.
- 2 **Преобразуем признаки** (feature transformation) — изменяем данные для повышения точности алгоритма.
- 3 **Отбираем признаки** (feature selection) — отсекаем «ненужные» (малоинформативные) признаки. Подробнее методы выбора информативных признаков будут рассмотрены в теме **Регрессия**.

Примеры

- 1 Предварительная обработка цифрового изображения: поиск лицевых фрагментов, выделение ключевых точек, применение геометрических преобразований (поворот, масштабирование).

Извлечение признаков

Выделение признаков — это процесс сокращения исходного набора «сырых» переменных до более управляемых групп (признаков) для дальнейшей обработки, оставаясь при этом достаточным набором для точного и полного описания исходного набора данных.

Требования к признакам:

- А Признаки должны отражать зависимости и закономерности предметной области.
- В Признаки не должны коррелировать, т.е. нести одну и ту же информацию (например, расстояния в километрах и милях).

Можно считать этот процесс снижением размерности. Условно его можно разбить на три этапа.

- 1 **Извлекаем признаки** (feature extraction) — преобразовываем специфические представления для данной предметной области в числовые или категориальные векторы.
- 2 **Преобразуем признаки** (feature transformation) — изменяем данные для повышения точности алгоритма.
- 3 **Отбираем признаки** (feature selection) — отсекаем «ненужные» (малоинформативные) признаки. Подробнее методы выбора информативных признаков будут рассмотрены в теме **Регрессия**.

Примеры

- 1 Предварительная обработка цифрового изображения: поиск лицевых фрагментов, выделение ключевых точек, применение геометрических преобразований (поворот, масштабирование).
- 2 Предварительная обработка речевого сигнала: шумоочистка, определение областей речевой активности (речь/не речь), выделение низкоуровневых признаков, характеризующих сигнал на основе его спектральных и временных характеристик.

- 3 Предварительная обработка текстов. Активно используется в задачах *NLP (Natural Language Processing)* — машинный перевод, чат-боты и пр., задаче *TTS (Text to Speech)*.

3 Предварительная обработка текстов. Активно используется в задачах *NLP (Natural Language Processing)* — машинный перевод, чат-боты и пр., задаче *TTS (Text to Speech)*.

✓ *Токенизация* — сегментация текста на предложения и слова (*токены*).

3 Предварительная обработка текстов. Активно используется в задачах *NLP (Natural Language Processing)* — машинный перевод, чат-боты и пр., задаче *TTS (Text to Speech)*.

- ✓ *Токенизация* — сегментация текста на предложения и слова (*токены*).
- ✓ Расшифровка числительных, специальных символов (\$, €, £, ©, §, ☺, ☹) или их удаление из текста.

3 Предварительная обработка текстов. Активно используется в задачах *NLP (Natural Language Processing)* — машинный перевод, чат-боты и пр., задаче *TTS (Text to Speech)*.

- ✓ *Токенизация* — сегментация текста на предложения и слова (*токены*).
- ✓ Расшифровка числительных, специальных символов (\$, €, £, ©, §, ☺, ☹) или их удаление из текста.
- ✓ Удаление или учет знаков пунктуации и эмфазы.

3 Предварительная обработка текстов. Активно используется в задачах *NLP (Natural Language Processing)* — машинный перевод, чат-боты и пр., задаче *TTS (Text to Speech)*.

- ✓ *Токенизация* — сегментация текста на предложения и слова (*токены*).
- ✓ Расшифровка числительных, специальных символов (\$, €, £, ©, §, ☺, ☹) или их удаление из текста.
- ✓ Удаление или учет знаков пунктуации и эмфазы.
- ✓ Удаление стоп слов — фонетически незначимых слов (предлоги, междометия и т. д).

3 Предварительная обработка текстов. Активно используется в задачах *NLP (Natural Language Processing)* — машинный перевод, чат-боты и пр., задаче *TTS (Text to Speech)*.

- ✓ *Токенизация* — сегментация текста на предложения и слова (*токены*).
- ✓ Расшифровка числительных, специальных символов (\$, €, £, ©, §, ☺, ☹) или их удаление из текста.
- ✓ Удаление или учет знаков пунктуации и эмфазы.
- ✓ Удаление стоп слов — фонетически незначимых слов (предлоги, междометия и т. д).
- ✓ *Стеммизация (стемминг)* — приведение слова к основной форме: удаление суффиксов, приставок, окончаний: **Перелесок** → **лес**.

3 Предварительная обработка текстов. Активно используется в задачах *NLP (Natural Language Processing)* — машинный перевод, чат-боты и пр., задаче *TTS (Text to Speech)*.

- ✓ *Токенизация* — сегментация текста на предложения и слова (*токены*).
- ✓ Расшифровка числительных, специальных символов (\$, €, £, ©, §, ☺, ☹) или их удаление из текста.
- ✓ Удаление или учет знаков пунктуации и эмфазы.
- ✓ Удаление стоп слов — фонетически незначимых слов (предлоги, междометия и т. д).
- ✓ *Стеммизация (стемминг)* — приведение слова к основной форме: удаление суффиксов, приставок, окончаний: **Перелесок** → **лес**.
- ✓ *Лемматизация* — приведении слова к его словарной форме (лемме). Альтернатива стеммингу: **отправляешь** → **отправлять**.

Пример предобработки текста

3 Предварительная обработка текстов. Активно используется в задачах *NLP (Natural Language Processing)* — машинный перевод, чат-боты и пр., задаче *TTS (Text to Speech)*.

- ✓ **Токенизация** — сегментация текста на предложения и слова (*токены*).
- ✓ Расшифровка числительных, специальных символов (\$, €, £, ©, §, ☺, ☹) или их удаление из текста.
- ✓ Удаление или учет знаков пунктуации и эмфазы.
- ✓ Удаление стоп слов — фонетически незначимых слов (предлоги, междометия и т. д).
- ✓ **Стеммизация (стемминг)** — приведение слова к основной форме: удаление суффиксов, приставок, окончаний: **Перелесок** → **лес**.
- ✓ **Лемматизация** — приведении слова к его словарной форме (лемме). Альтернатива стеммингу: **отправляешь** → **отправлять**.
- ✓ Векторное представление слов. Основной подход — *мешок слов (bag-of-words, BOW)*. Документ записан в виде матрицы: строки — тексты (предложения), столбцы — слова из словаря, на пересечении — число вхождений слова в текст.

Таблица 1. Пример мешка слов

	квартиру	купить	лекарства	нужно	продукты	срочно	
срочно купить квартиру	1	1	0	0	0	1	(1, 1, 0, 0, 0, 1)
нужно купить лекарства	0	1	1	1	0	0	(0, 1, 1, 1, 0, 0)
нужно купить продукты	0	1	0	1	1	0	(0, 1, 0, 1, 1, 0)

Алгоритм называется мешком слов, потому что вся информация о порядке или структуре слов в документе отбрасывается. Определяется только встречаются ли известные слова в документе.

Пример предобработки текста


3 Предварительная обработка текстов. Активно используется в задачах *NLP (Natural Language Processing)* — машинный перевод, чат-боты и пр., задаче *TTS (Text to Speech)*.

- ✓ **Токенизация** — сегментация текста на предложения и слова (*токены*).
- ✓ Расшифровка числительных, специальных символов (\$, €, £, ©, §, ☺, ☹) или их удаление из текста.
- ✓ Удаление или учет знаков пунктуации и эмфазы.
- ✓ Удаление стоп слов — фонетически незначимых слов (предлоги, междометия и т. д.).
- ✓ **Стеммизация (стемминг)** — приведение слова к основной форме: удаление суффиксов, приставок, окончаний: **Перелесок** → **лес**.
- ✓ **Лемматизация** — приведении слова к его словарной форме (лемме). Альтернатива стеммингу: **отправляешь** → **отправлять**.
- ✓ Векторное представление слов. Основной подход — *мешок слов (bag-of-words, BOW)*. Документ записан в виде матрицы: строки — тексты (предложения), столбцы — слова из словаря, на пересечении — число вхождений слова в текст.

Таблица 1. Пример мешка слов

	квартиру	купить	лекарства	нужно	продукты	срочно	
срочно купить квартиру	1	1	0	0	0	1	(1, 1, 0, 0, 0, 1)
нужно купить лекарства	0	1	1	1	0	0	(0, 1, 1, 1, 0, 0)
нужно купить продукты	0	1	0	1	1	0	(0, 1, 0, 1, 1, 0)

Алгоритм называется мешком слов, потому что вся информация о порядке или структуре слов в документе отбрасывается. Определяется только встречаются ли известные слова в документе.

 Что удалять из исходных данных, а что оставлять зависит от постановки задачи.

Data set

Data set

Data set (правильно писать отдельно, хотя сейчас часто пишут слитно) — это обработанный и структурированный набор данных, представленный в табличном виде. Строки такой таблицы называются **объектами**, а столбцы — **признаками** (фичами (features)). В совокупности они составляют размеченные данные, на основе которых происходит машинное обучение.

Data set

Data set (правильно писать отдельно, хотя сейчас часто пишут слитно) — это обработанный и структурированный набор данных, представленный в табличном виде. Строки такой таблицы называются **объектами**, а столбцы — **признаками** (фичами (features)). В совокупности они составляют размеченные данные, на основе которых происходит машинное обучение.

Виды признаков

Data set

Data set (правильно писать отдельно, хотя сейчас часто пишут слитно) — это обработанный и структурированный набор данных, представленный в табличном виде. Строки такой таблицы называются **объектами**, а столбцы — **признаками** (фичами (features)). В совокупности они составляют размеченные данные, на основе которых происходит машинное обучение.

Виды признаков

В Dataset различают два вида признаков:

- 1 независимые переменные — обычно их называют *предикторами*, *факторами*,

Data set

Data set (правильно писать отдельно, хотя сейчас часто пишут слитно) — это обработанный и структурированный набор данных, представленный в табличном виде. Строки такой таблицы называются **объектами**, а столбцы — **признаками** (фичами (features)). В совокупности они составляют размеченные данные, на основе которых происходит машинное обучение.

Виды признаков

В Dataset различают два вида признаков:

- 1 независимые переменные — обычно их называют *предикторами*, *факторами*,
- 2 целевые (зависимые) переменные – признаки, которые вычисляются на основе одного или нескольких предикторов.

Data set

Data set (правильно писать отдельно, хотя сейчас часто пишут слитно) — это обработанный и структурированный набор данных, представленный в табличном виде. Строки такой таблицы называются **объектами**, а столбцы — **признаками** (фичами (features)). В совокупности они составляют размеченные данные, на основе которых происходит машинное обучение.

Виды признаков

В Dataset различают два вида признаков:

- 1 независимые переменные — обычно их называют *предикторами*, *факторами*,
- 2 целевые (зависимые) переменные – признаки, которые вычисляются на основе одного или нескольких предикторов.

Типы признаков

Data set

Data set (правильно писать отдельно, хотя сейчас часто пишут слитно) — это обработанный и структурированный набор данных, представленный в табличном виде. Строки такой таблицы называются **объектами**, а столбцы — **признаками** (фичами (features)). В совокупности они составляют размеченные данные, на основе которых происходит машинное обучение.

Виды признаков

В Dataset различают два вида признаков:

- 1 независимые переменные — обычно их называют *предикторами*, *факторами*,
- 2 целевые (зависимые) переменные – признаки, которые вычисляются на основе одного или нескольких предикторов.

Типы признаков

Входные (независимые, предикторы) и выходные (целевые) переменные могут быть следующих типов.

- 1 *Количественные* (численные, вариационные). Измеряются в непрерывной (например, температура) или в интервальной шкале (попадание в определенный интервал значений, например, возраст). Частный случай — бинарные (дихотомические) признаки, имеющие два значения $\{0, 1\}$.

Data set

Data set (правильно писать отдельно, хотя сейчас часто пишут слитно) — это обработанный и структурированный набор данных, представленный в табличном виде. Строки такой таблицы называются **объектами**, а столбцы — **признаками** (фичами (features)). В совокупности они составляют размеченные данные, на основе которых происходит машинное обучение.

Виды признаков

В Dataset различают два вида признаков:

- 1 независимые переменные — обычно их называют *предикторами, факторами*,
- 2 целевые (зависимые) переменные – признаки, которые вычисляются на основе одного или нескольких предикторов.

Типы признаков

Входные (независимые, предикторы) и выходные (целевые) переменные могут быть следующих типов.

- 1 *Количественные* (численные, вариационные). Измеряются в непрерывной (например, температура) или в интервальной шкале (попадание в определенный интервал значений, например, возраст). Частный случай — бинарные (дихотомические) признаки, имеющие два значения $\{0, 1\}$.
- 2 *Порядковые* (признаки с упорядоченными состояниями, ординальные признаки), например: горячо, тепло, холодно. Здесь имеет значение порядок.

Data set

Data set (правильно писать отдельно, хотя сейчас часто пишут слитно) — это обработанный и структурированный набор данных, представленный в табличном виде. Строки такой таблицы называются **объектами**, а столбцы — **признаками** (фичами (features)). В совокупности они составляют размеченные данные, на основе которых происходит машинное обучение.

Виды признаков

В Dataset различают два вида признаков:

- 1 независимые переменные — обычно их называют *предикторами, факторами*,
- 2 целевые (зависимые) переменные – признаки, которые вычисляются на основе одного или нескольких предикторов.

Типы признаков

Входные (независимые, предикторы) и выходные (целевые) переменные могут быть следующих типов.

- 1 *Количественные* (численные, вариационные). Измеряются в непрерывной (например, температура) или в интервальной шкале (попадание в определенный интервал значений, например, возраст). Частный случай — бинарные (дихотомические) признаки, имеющие два значения $\{0, 1\}$.
- 2 *Порядковые* (признаки с упорядоченными состояниями, ординальные признаки), например: горячо, тепло, холодно. Здесь имеет значение порядок.
- 3 *Номинальные признаки* (признаки с неупорядоченными состояниями, классификационные, категориальные, факторные), например: яблоко, груша, арбуз. Взаимный порядок здесь уже не имеет значения. Важна принадлежность классу.

Data set

Data set (правильно писать отдельно, хотя сейчас часто пишут слитно) — это обработанный и структурированный набор данных, представленный в табличном виде. Строки такой таблицы называются **объектами**, а столбцы — **признаками** (фичами (features)). В совокупности они составляют размеченные данные, на основе которых происходит машинное обучение.

Виды признаков

В Dataset различают два вида признаков:

- 1 независимые переменные — обычно их называют *предикторами, факторами*,
- 2 целевые (зависимые) переменные – признаки, которые вычисляются на основе одного или нескольких предикторов.

Типы признаков

Входные (независимые, предикторы) и выходные (целевые) переменные могут быть следующих типов.

- 1 *Количественные* (численные, вариационные). Измеряются в непрерывной (например, температура) или в интервальной шкале (попадание в определенный интервал значений, например, возраст). Частный случай — бинарные (дихотомические) признаки, имеющие два значения $\{0, 1\}$.
- 2 *Порядковые* (признаки с упорядоченными состояниями, ординальные признаки), например: горячо, тепло, холодно. Здесь имеет значение порядок.
- 3 *Номинальные признаки* (признаки с неупорядоченными состояниями, классификационные, категориальные, факторные), например: яблоко, груша, арбуз. Взаимный порядок здесь уже не имеет значения. Важна принадлежность классу.

Вообще существуют различные принципы классификации признаков.


Мотивация

- 1 Многие алгоритмы машинного обучения не могут обрабатывать категориальные переменные, если они не преобразованы в числовые.


Мотивация

- 1 Многие алгоритмы машинного обучения не могут обрабатывать категориальные переменные, если они не преобразованы в числовые.
- 2 Производительность многих алгоритмов машинного обучения зависит от того, как закодированы категориальные переменные.

Мотивация


- 1 Многие алгоритмы машинного обучения не могут обрабатывать категориальные переменные, если они не преобразованы в числовые.
- 2 Производительность многих алгоритмов машинного обучения зависит от того, как закодированы категориальные переменные.
-  Однако существует множество алгоритмов, которые поддерживают категориальные значения без дополнительных преобразований. Задача: когда нужно преобразовывать признаки?

Мотивация

- 1 Многие алгоритмы машинного обучения не могут обрабатывать категориальные переменные, если они не преобразованы в числовые.
 - 2 Производительность многих алгоритмов машинного обучения зависит от того, как закодированы категориальные переменные.
-  Однако существует множество алгоритмов, которые поддерживают категориальные значения без дополнительных преобразований. Задача: когда нужно преобразовывать признаки?

Методы кодирования

Мотивация

- 1 Многие алгоритмы машинного обучения не могут обрабатывать категориальные переменные, если они не преобразованы в числовые.
 - 2 Производительность многих алгоритмов машинного обучения зависит от того, как закодированы категориальные переменные.
-  Однако существует множество алгоритмов, которые поддерживают категориальные значения без дополнительных преобразований. Задача: когда нужно преобразовывать признаки?

Методы кодирования

- 1 Кодирование меток (Label Encoding)

Мотивация

- ❶ Многие алгоритмы машинного обучения не могут обрабатывать категориальные переменные, если они не преобразованы в числовые.
- ❷ Производительность многих алгоритмов машинного обучения зависит от того, как закодированы категориальные переменные.
- ❸ Однако существует множество алгоритмов, которые поддерживают категориальные значения без дополнительных преобразований. Задача: когда нужно преобразовывать признаки?

Методы кодирования

- ❶ Кодирование меток (Label Encoding)
 - ✓ Каждой категории присваивается значение от 1 до N . Например, апельсин — 1, яблоко — 2, груша — 3, мандарин — 4. Таблица 2, третий столбец

Мотивация

- ❶ Многие алгоритмы машинного обучения не могут обрабатывать категориальные переменные, если они не преобразованы в числовые.
- ❷ Производительность многих алгоритмов машинного обучения зависит от того, как закодированы категориальные переменные.
- ❸ Однако существует множество алгоритмов, которые поддерживают категориальные значения без дополнительных преобразований. Задача: когда нужно преобразовывать признаки?

Методы кодирования

- ❶ Кодирование меток (Label Encoding)
 - ✓ Каждой категории присваивается значение от 1 до N . Например, апельсин — 1, яблоко — 2, груша — 3, мандарин — 4. Таблица 2, третий столбец
 - ✓ Недостаток: между этими признаками нет отношения порядка, но алгоритм машинного обучения может рассматривать их как своего рода порядок. Например, значение 1 меньше значения 4, но действительно ли это соответствует набору данных в реальной задаче?

Мотивация

- 1 Многие алгоритмы машинного обучения не могут обрабатывать категориальные переменные, если они не преобразованы в числовые.
- 2 Производительность многих алгоритмов машинного обучения зависит от того, как закодированы категориальные переменные.
- i Однако существует множество алгоритмов, которые поддерживают категориальные значения без дополнительных преобразований. Задача: когда нужно преобразовывать признаки?

Методы кодирования

- 1 Кодирование меток (Label Encoding)
 - ✓ Каждой категории присваивается значение от 1 до N . Например, апельсин — 1, яблоко — 2, груша — 3, мандарин — 4. Таблица 2, третий столбец
 - ✓ Недостаток: между этими признаками нет отношения порядка, но алгоритм машинного обучения может рассматривать их как своего рода порядок. Например, значение 1 меньше значения 4, но действительно ли это соответствует набору данных в реальной задаче?
- 2 Унитарное кодирование (One Hot Encoding)

Мотивация

- ❶ Многие алгоритмы машинного обучения не могут обрабатывать категориальные переменные, если они не преобразованы в числовые.
- ❷ Производительность многих алгоритмов машинного обучения зависит от того, как закодированы категориальные переменные.
- ❸ Однако существует множество алгоритмов, которые поддерживают категориальные значения без дополнительных преобразований. Задача: когда нужно преобразовывать признаки?

Методы кодирования

- ❶ Кодирование меток (Label Encoding)
 - ✓ Каждой категории присваивается значение от 1 до N . Например, апельсин — 1, яблоко — 2, груша — 3, мандарин — 4. Таблица 2, третий столбец
 - ✓ Недостаток: между этими признаками нет отношения порядка, но алгоритм машинного обучения может рассматривать их как своего рода порядок. Например, значение 1 меньше значения 4, но действительно ли это соответствует набору данных в реальной задаче?
- ❷ Унитарное кодирование (One Hot Encoding)
 - ✓ Каждое значение категориального признака преобразуется в новый столбец со значениями 1 или 0. Таблица 2, четвертый — седьмой столбцы.

Мотивация

- 1 Многие алгоритмы машинного обучения не могут обрабатывать категориальные переменные, если они не преобразованы в числовые.
- 2 Производительность многих алгоритмов машинного обучения зависит от того, как закодированы категориальные переменные.
- i Однако существует множество алгоритмов, которые поддерживают категориальные значения без дополнительных преобразований. Задача: когда нужно преобразовывать признаки?

Методы кодирования

- 1 Кодирование меток (Label Encoding)
 - ✓ Каждой категории присваивается значение от 1 до N . Например, апельсин — 1, яблоко — 2, груша — 3, мандарин — 4. Таблица 2, третий столбец
 - ✓ Недостаток: между этими признаками нет отношения порядка, но алгоритм машинного обучения может рассматривать их как своего рода порядок. Например, значение 1 меньше значения 4, но действительно ли это соответствует набору данных в реальной задаче?
- 2 Унитарное кодирование (One Hot Encoding)
 - ✓ Каждое значение категориального признака преобразуется в новый столбец со значениями 1 или 0. Таблица 2, четвертый — седьмой столбцы.
 - ✓ Теперь значение признака учитывается правильно, но имеет обратную сторону добавления дополнительных столбцов в набор данных.

Мотивация

- ❶ Многие алгоритмы машинного обучения не могут обрабатывать категориальные переменные, если они не преобразованы в числовые.
- ❷ Производительность многих алгоритмов машинного обучения зависит от того, как закодированы категориальные переменные.
- ❸ Однако существует множество алгоритмов, которые поддерживают категориальные значения без дополнительных преобразований. Задача: когда нужно преобразовывать признаки?

Методы кодирования

- ❶ Кодирование меток (Label Encoding)
 - ✓ Каждой категории присваивается значение от 1 до N . Например, апельсин — 1, яблоко — 2, груша — 3, мандарин — 4. Таблица 2, третий столбец
 - ✓ Недостаток: между этими признаками нет отношения порядка, но алгоритм машинного обучения может рассматривать их как своего рода порядок. Например, значение 1 меньше значения 4, но действительно ли это соответствует набору данных в реальной задаче?
- ❷ Унитарное кодирование (One Hot Encoding)
 - ✓ Каждое значение категориального признака преобразуется в новый столбец со значениями 1 или 0. Таблица 2, четвертый — седьмой столбцы.
 - ✓ Теперь значение признака учитывается правильно, но имеет обратную сторону добавления дополнительных столбцов в набор данных.
 - ✓ Недостаток: в набор данных добавляются дополнительные столбцы.

Кодирование категориальных признаков

Мотивация

- 1 Многие алгоритмы машинного обучения не могут обрабатывать категориальные переменные, если они не преобразованы в числовые.
- 2 Производительность многих алгоритмов машинного обучения зависит от того, как закодированы категориальные переменные.
- 1 Однако существует множество алгоритмов, которые поддерживают категориальные значения без дополнительных преобразований. Задача: когда нужно преобразовывать признаки?

Методы кодирования

- 1 Кодирование меток (Label Encoding)
 - ✓ Каждой категории присваивается значение от 1 до N . Например, апельсин — 1, яблоко — 2, груша — 3, мандарин — 4. Таблица 2, третий столбец
 - ✓ Недостаток: между этими признаками нет отношения порядка, но алгоритм машинного обучения может рассматривать их как своего рода порядок. Например, значение 1 меньше значения 4, но действительно ли это соответствует набору данных в реальной задаче?
- 2 Унитарное кодирование (One Hot Encoding)
 - ✓ Каждое значение категориального признака преобразуется в новый столбец со значениями 1 или 0. Таблица 2, четвертый — седьмой столбцы.
 - ✓ Теперь значение признака учитывается правильно, но имеет обратную сторону добавления дополнительных столбцов в набор данных.
 - ✓ Недостаток: в набор данных добавляются дополнительные столбцы.
- 3 Частотное кодирование (Frequency Encoding)

Мотивация

- 1 Многие алгоритмы машинного обучения не могут обрабатывать категориальные переменные, если они не преобразованы в числовые.
- 2 Производительность многих алгоритмов машинного обучения зависит от того, как закодированы категориальные переменные.
- 1 Однако существует множество алгоритмов, которые поддерживают категориальные значения без дополнительных преобразований. Задача: когда нужно преобразовывать признаки?

Методы кодирования

- 1 Кодирование меток (Label Encoding)
 - ✓ Каждой категории присваивается значение от 1 до N . Например, апельсин — 1, яблоко — 2, груша — 3, мандарин — 4. Таблица 2, третий столбец
 - ✓ Недостаток: между этими признаками нет отношения порядка, но алгоритм машинного обучения может рассматривать их как своего рода порядок. Например, значение 1 меньше значения 4, но действительно ли это соответствует набору данных в реальной задаче?
- 2 Унитарное кодирование (One Hot Encoding)
 - ✓ Каждое значение категориального признака преобразуется в новый столбец со значениями 1 или 0. Таблица 2, четвертый — седьмой столбцы.
 - ✓ Теперь значение признака учитывается правильно, но имеет обратную сторону добавления дополнительных столбцов в набор данных.
 - ✓ Недостаток: в набор данных добавляются дополнительные столбцы.
- 3 Частотное кодирование (Frequency Encoding)
 - ✓ Категории заменяем на количество наблюдений, которые показывают эту категорию в наборе данных или на частоту наблюдений в наборе данных.

Мотивация

- 1 Многие алгоритмы машинного обучения не могут обрабатывать категориальные переменные, если они не преобразованы в числовые.
- 2 Производительность многих алгоритмов машинного обучения зависит от того, как закодированы категориальные переменные.
- 1 Однако существует множество алгоритмов, которые поддерживают категориальные значения без дополнительных преобразований. Задача: когда нужно преобразовывать признаки?

Методы кодирования

- 1 Кодирование меток (Label Encoding)
 - ✓ Каждой категории присваивается значение от 1 до N . Например, апельсин — 1, яблоко — 2, груша — 3, мандарин — 4. Таблица 2, третий столбец
 - ✓ Недостаток: между этими признаками нет отношения порядка, но алгоритм машинного обучения может рассматривать их как своего рода порядок. Например, значение 1 меньше значения 4, но действительно ли это соответствует набору данных в реальной задаче?
- 2 Унитарное кодирование (One Hot Encoding)
 - ✓ Каждое значение категориального признака преобразуется в новый столбец со значениями 1 или 0. Таблица 2, четвертый — седьмой столбцы.
 - ✓ Теперь значение признака учитывается правильно, но имеет обратную сторону добавления дополнительных столбцов в набор данных.
 - ✓ Недостаток: в набор данных добавляются дополнительные столбцы.
- 3 Частотное кодирование (Frequency Encoding)
 - ✓ Категории заменяем на количество наблюдений, которые показывают эту категорию в наборе данных или на частоту наблюдений в наборе данных.
 - ✓ Достоинства: не расширяет пространство признаков, хорошо работает с алгоритмами на основе деревьев.

Кодирование категориальных признаков

Мотивация

- 1 Многие алгоритмы машинного обучения не могут обрабатывать категориальные переменные, если они не преобразованы в числовые.
- 2 Производительность многих алгоритмов машинного обучения зависит от того, как закодированы категориальные переменные.
- 1 Однако существует множество алгоритмов, которые поддерживают категориальные значения без дополнительных преобразований. Задача: когда нужно преобразовывать признаки?

Методы кодирования

- 1 Кодирование меток (Label Encoding)
 - ✓ Каждой категории присваивается значение от 1 до N . Например, апельсин — 1, яблоко — 2, груша — 3, мандарин — 4. Таблица 2, третий столбец
 - ✓ Недостаток: между этими признаками нет отношения порядка, но алгоритм машинного обучения может рассматривать их как своего рода порядок. Например, значение 1 меньше значения 4, но действительно ли это соответствует набору данных в реальной задаче?
- 2 Унитарное кодирование (One Hot Encoding)
 - ✓ Каждое значение категориального признака преобразуется в новый столбец со значениями 1 или 0. Таблица 2, четвертый — седьмой столбцы.
 - ✓ Теперь значение признака учитывается правильно, но имеет обратную сторону добавления дополнительных столбцов в набор данных.
 - ✓ Недостаток: в набор данных добавляются дополнительные столбцы.
- 3 Частотное кодирование (Frequency Encoding)
 - ✓ Категории заменяем на количество наблюдений, которые показывают эту категорию в наборе данных или на частоту наблюдений в наборе данных.
 - ✓ Достоинства: не расширяет пространство признаков, хорошо работает с алгоритмами на основе деревьев.
 - ✓ Недостаток: несколько разных категорий с одинаковым количеством наблюдений.

Таблица 2. Кодирование категориальных признаков

	Фрукты	Label Encoding	One Hot Encoding			
			Апельсин	Яблоко	Груша	Мандарин
X ₁	Апельсин	1	1	0	0	0
X ₂	Яблоко	2	0	1	0	0
X ₃	Груша	3	0	0	1	0
X ₄	Мандарин	4	0	0	0	1
X ₅	Апельсин	1	1	0	0	0
X ₆	Мандарин	4	0	0	0	1
X ₇	Яблоко	2	0	1	0	0
X ₈	Груша	3	0	0	1	0
X ₉	Мандарин	4	0	0	0	1
X ₁₀	Груша	4	0	0	1	0
X ₁₁	Яблоко	4	0	1	0	0
X ₁₂	Яблоко	4	0	1	0	0

- 1 Наборы данных, которые используются для построения модели, обычно собираются из различных источников.

- 1 Наборы данных, которые используются для построения модели, обычно собираются из различных источников.
- 2 Часто набор данных содержит признаки разных масштабов и даже разные единицы измерения, например килограммы, километры, часы, их сложно сравнивать. Что такое килограммы по сравнению с часами?

- 1 Наборы данных, которые используются для построения модели, обычно собираются из различных источников.
- 2 Часто набор данных содержит признаки разных масштабов и даже разные единицы измерения, например килограммы, километры, часы, их сложно сравнивать. Что такое килограммы по сравнению с часами?
- 3 Различные масштабы данных отрицательно влияют на моделирование набора данных.

- 1 Наборы данных, которые используются для построения модели, обычно собираются из различных источников.
- 2 Часто набор данных содержит признаки разных масштабов и даже разные единицы измерения, например килограммы, километры, часы, их сложно сравнивать. Что такое килограммы по сравнению с часами?
- 3 Различные масштабы данных отрицательно влияют на моделирование набора данных.
- 4 Чтобы модель машинного обучения работала хорошо, важно, чтобы данные имели **одинаковый масштаб**, чтобы избежать предвзятого результата прогнозирования.

- 1 Наборы данных, которые используются для построения модели, обычно собираются из различных источников.
- 2 Часто набор данных содержит признаки разных масштабов и даже разные единицы измерения, например килограммы, километры, часы, их сложно сравнивать. Что такое килограммы по сравнению с часами?
- 3 Различные масштабы данных отрицательно влияют на моделирование набора данных.
- 4 Чтобы модель машинного обучения работала хорошо, важно, чтобы данные имели **одинаковый масштаб**, чтобы избежать предвзятого результата прогнозирования.
- 5 Заметим, что разница в масштабе входных переменных влияет не на все алгоритмы машинного обучения.

- 1 Наборы данных, которые используются для построения модели, обычно собираются из различных источников.
- 2 Часто набор данных содержит признаки разных масштабов и даже разные единицы измерения, например килограммы, километры, часы, их сложно сравнивать. Что такое килограммы по сравнению с часами?
- 3 Различные масштабы данных отрицательно влияют на моделирование набора данных.
- 4 Чтобы модель машинного обучения работала хорошо, важно, чтобы данные имели **одинаковый масштаб**, чтобы избежать предвзятого результата прогнозирования.
- 5 Заметим, что разница в масштабе входных переменных влияет не на все алгоритмы машинного обучения.
- 6 Например, это затрагивает алгоритмы, использующие взвешенную сумму входных переменных, такие как линейная и логистическая регрессия.

- 1 Наборы данных, которые используются для построения модели, обычно собираются из различных источников.
- 2 Часто набор данных содержит признаки разных масштабов и даже разные единицы измерения, например килограммы, километры, часы, их сложно сравнивать. Что такое килограммы по сравнению с часами?
- 3 Различные масштабы данных отрицательно влияют на моделирование набора данных.
- 4 Чтобы модель машинного обучения работала хорошо, важно, чтобы данные имели **одинаковый масштаб**, чтобы избежать предвзятого результата прогнозирования.
- 5 Заметим, что разница в масштабе входных переменных влияет не на все алгоритмы машинного обучения.
- 6 Например, это затрагивает алгоритмы, использующие взвешенную сумму входных переменных, такие как линейная и логистическая регрессия.
- 7 Также, это существенно для алгоритмов, использующие меры расстояния между примерами, таких как метод k-ближайших соседей и машина опорных векторов.

- 1 Наборы данных, которые используются для построения модели, обычно собираются из различных источников.
- 2 Часто набор данных содержит признаки разных масштабов и даже разные единицы измерения, например килограммы, километры, часы, их сложно сравнивать. Что такое килограммы по сравнению с часами?
- 3 Различные масштабы данных отрицательно влияют на моделирование набора данных.
- 4 Чтобы модель машинного обучения работала хорошо, важно, чтобы данные имели **одинаковый масштаб**, чтобы избежать предвзятого результата прогнозирования.
- 5 Заметим, что разница в масштабе входных переменных влияет не на все алгоритмы машинного обучения.
- 6 Например, это затрагивает алгоритмы, использующие взвешенную сумму входных переменных, такие как линейная и логистическая регрессия.
- 7 Также, это существенно для алгоритмов, использующие меры расстояния между примерами, таких как метод k-ближайших соседей и машина опорных векторов.
- 8 Существуют также алгоритмы, на которые не влияет масштаб числовых входных переменных, в первую очередь деревья решений и ансамбли деревьев, такие как случайный лес.

A Имеется выборка объектов со следующими значениями: *«Оценка знания иностранного языка»* и *«Возраст»* (таблица 3).

Таблица 3

	Оценка	Возраст
A	3	60
B	3	40
C	4	40
D	4.5	50
E	4.2	52

A Имеется выборка объектов со следующими значениями: *«Оценка знания иностранного языка»* и *«Возраст»* (таблица 3).

B Евклидово расстояние между объектами:

$$\begin{aligned}\rho(A, B) &= \sqrt{(60 - 40)^2 + (3 - 3)^2} = 20, \\ \rho(B, C) &= \sqrt{(40 - 40)^2 + (3 - 4)^2} = 1\end{aligned}\tag{1}$$

Таблица 3

	Оценка	Возраст
A	3	60
B	3	40
C	4	40
D	4.5	50
E	4.2	52

Нормализация и стандартизация данных. Пример

A Имеется выборка объектов со следующими значениями: «Оценка знания иностранного языка» и «Возраст» (таблица 3).

B Евклидово расстояние между объектами:

$$\begin{aligned}\rho(A, B) &= \sqrt{(60 - 40)^2 + (3 - 3)^2} = 20, \\ \rho(B, C) &= \sqrt{(40 - 40)^2 + (3 - 4)^2} = 1\end{aligned}\quad (1)$$

C Проведем **стандартизацию** данных (таблица 4).

Таблица 3

	Оценка	Возраст
A	3	60
B	3	40
C	4	40
D	4.5	50
E	4.2	52

Таблица 4

	Оценка	Возраст
A	-1.18434	1.520013
B	-1.18434	-1.10069
C	0.416120	-1.10069
D	1.216350	0.209657
E	0.736212	0.471728

A Имеется выборка объектов со следующими значениями: «Оценка знания иностранного языка» и «Возраст» (таблица 3).

B Евклидово расстояние между объектами:

$$\begin{aligned}\rho(A, B) &= \sqrt{(60 - 40)^2 + (3 - 3)^2} = 20, \\ \rho(B, C) &= \sqrt{(40 - 40)^2 + (3 - 4)^2} = 1\end{aligned}\tag{1}$$

C Проведем **стандартизацию** данных (таблица 4).

D Пересчитаем евклидово расстояние после стандартизации:

$$\begin{aligned}\rho(A, B) &\approx \sqrt{(1.5 + 1.1)^2 + (1.18 - 1.18)^2} = 2.6, \\ \rho(B, C) &\approx \sqrt{(1.1 - 1.1)^2 + (0.41 + 1.18)^2} = 1.59\end{aligned}\tag{2}$$

Таблица 3

	Оценка	Возраст
A	3	60
B	3	40
C	4	40
D	4.5	50
E	4.2	52

Таблица 4

	Оценка	Возраст
A	-1.18434	1.520013
B	-1.18434	-1.10069
C	0.416120	-1.10069
D	1.216350	0.209657
E	0.736212	0.471728

1 **Нормализация** — это изменение масштаба исходных данных так, чтобы все значения признаков x_1, \dots, x_k во входном векторе \mathbf{X} находились в диапазоне $[0, 1]$. Иногда процесс называют *масштабирование Min-Max*.

- 1 **Нормализация** — это изменение масштаба исходных данных так, чтобы все значения признаков x_1, \dots, x_k во входном векторе \mathbf{X} находились в диапазоне $[0, 1]$. Иногда процесс называют *масштабирование Min-Max*.
- 2 Проводим линейное преобразование данных в диапазоне $[0, 1]$, где минимальное и максимальное масштабируемые значения соответствуют 0 и 1 соответственно.

$$\tilde{x}_i = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}. \quad (3)$$

1 **Нормализация** — это изменение масштаба исходных данных так, чтобы все значения признаков x_1, \dots, x_k во входном векторе \mathbf{X} находились в диапазоне $[0, 1]$. Иногда процесс называют *масштабирование Min-Max*.

2 Проводим линейное преобразование данных в диапазоне $[0, 1]$, где минимальное и максимальное масштабируемые значения соответствуют 0 и 1 соответственно.

$$\tilde{x}_i = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}. \quad (3)$$

3 Формулу (3) можно обобщить на приведение исходного набора значений к произвольному диапазону $[a, b]$:

$$\tilde{x}_i = a + \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}(b - a). \quad (4)$$

1 **Нормализация** — это изменение масштаба исходных данных так, чтобы все значения признаков x_1, \dots, x_k во входном векторе \mathbf{X} находились в диапазоне $[0, 1]$. Иногда процесс называют *масштабирование Min-Max*.

2 Проводим линейное преобразование данных в диапазоне $[0, 1]$, где минимальное и максимальное масштабируемые значения соответствуют 0 и 1 соответственно.

$$\tilde{x}_i = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}. \quad (3)$$

3 Формулу (3) можно обобщить на приведение исходного набора значений к произвольному диапазону $[a, b]$:

$$\tilde{x}_i = a + \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}(b - a). \quad (4)$$

4 Пример нормализации набора (из трех признаков) в диапазон $[-2, 1.5]$ модулем *MinMaxScaler* из библиотеки *sklearn*:

```
import numpy as np
from sklearn.preprocessing import MinMaxScaler
dataset = np.array([[85, 72, 80], [64, 35, 26], [67, 48, 29], [100, 11, 102], [130, 14, 151]])
scaler = MinMaxScaler(feature_range=(-2.0, 1.5))
scaler.fit(dataset)
normalized_dataset = scaler.transform(dataset)
```

1 **Нормализация** — это изменение масштаба исходных данных так, чтобы все значения признаков x_1, \dots, x_k во входном векторе \mathbf{X} находились в диапазоне $[0, 1]$. Иногда процесс называют *масштабирование Min-Max*.

2 Проводим линейное преобразование данных в диапазоне $[0, 1]$, где минимальное и максимальное масштабируемые значения соответствуют 0 и 1 соответственно.

$$\tilde{x}_i = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}. \quad (3)$$

3 Формулу (3) можно обобщить на приведение исходного набора значений к произвольному диапазону $[a, b]$:

$$\tilde{x}_i = a + \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}(b - a). \quad (4)$$

4 Пример нормализации набора (из трех признаков) в диапазон $[-2, 1.5]$ модулем *MinMaxScaler* из библиотеки *sklearn*:

```
import numpy as np
from sklearn.preprocessing import MinMaxScaler
dataset = np.array([[85, 72, 80], [64, 35, 26], [67, 48, 29], [100, 11, 102], [130, 14, 151]])
scaler = MinMaxScaler(feature_range=(-2.0, 1.5))
scaler.fit(dataset)
normalized_dataset = scaler.transform(dataset)
```

i Преобразование полезно, когда распределение данных неизвестно или не является гауссовым.

1 **Нормализация** — это изменение масштаба исходных данных так, чтобы все значения признаков x_1, \dots, x_k во входном векторе \mathbf{X} находились в диапазоне $[0, 1]$. Иногда процесс называют *масштабирование Min-Max*.

2 Проводим линейное преобразование данных в диапазоне $[0, 1]$, где минимальное и максимальное масштабируемые значения соответствуют 0 и 1 соответственно.

$$\tilde{x}_i = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}. \quad (3)$$

3 Формулу (3) можно обобщить на приведение исходного набора значений к произвольному диапазону $[a, b]$:

$$\tilde{x}_i = a + \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}(b - a). \quad (4)$$

4 Пример нормализации набора (из трех признаков) в диапазон $[-2, 1.5]$ модулем *MinMaxScaler* из библиотеки *sklearn*:

```
import numpy as np
from sklearn.preprocessing import MinMaxScaler
dataset = np.array([[85, 72, 80], [64, 35, 26], [67, 48, 29], [100, 11, 102], [130, 14, 151]])
scaler = MinMaxScaler(feature_range=(-2.0, 1.5))
scaler.fit(dataset)
normalized_dataset = scaler.transform(dataset)
```

i Преобразование полезно, когда распределение данных неизвестно или не является гауссовым.

i Сохраняет форму исходного распределения данных.

1 **Нормализация** — это изменение масштаба исходных данных так, чтобы все значения признаков x_1, \dots, x_k во входном векторе \mathbf{X} находились в диапазоне $[0, 1]$. Иногда процесс называют *масштабирование Min-Max*.

2 Проводим линейное преобразование данных в диапазоне $[0, 1]$, где минимальное и максимальное масштабируемые значения соответствуют 0 и 1 соответственно.

$$\tilde{x}_i = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}. \quad (3)$$


3 Формулу (3) можно обобщить на приведение исходного набора значений к произвольному диапазону $[a, b]$:


$$\tilde{x}_i = a + \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}(b - a). \quad (4)$$

4 Пример нормализации набора (из трех признаков) в диапазон $[-2, 1.5]$ модулем *MinMaxScaler* из библиотеки *sklearn*:

```
import numpy as np
from sklearn.preprocessing import MinMaxScaler
dataset = np.array([[85, 72, 80], [64, 35, 26], [67, 48, 29], [100, 11, 102], [130, 14, 151]])
scaler = MinMaxScaler(feature_range=(-2.0, 1.5))
scaler.fit(dataset)
normalized_dataset = scaler.transform(dataset)
```

 Преобразование полезно, когда распределение данных неизвестно или не является гауссовым.

 Сохраняет форму исходного распределения данных.

 Недостаток: при наличии «выбросов» (больших значениях x_{\max}) нормализованные значения будут сконцентрированы в небольшом диапазоне около нуля.

1 **Нормализация** — это изменение масштаба исходных данных так, чтобы все значения признаков x_1, \dots, x_k во входном векторе \mathbf{X} находились в диапазоне $[0, 1]$. Иногда процесс называют *масштабирование Min-Max*.

2 Проводим линейное преобразование данных в диапазоне $[0, 1]$, где минимальное и максимальное масштабируемые значения соответствуют 0 и 1 соответственно.

$$\tilde{x}_i = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}. \quad (3)$$


3 Формулу (3) можно обобщить на приведение исходного набора значений к произвольному диапазону $[a, b]$:


$$\tilde{x}_i = a + \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}(b - a). \quad (4)$$


4 Пример нормализации набора (из трех признаков) в диапазон $[-2, 1.5]$ модулем *MinMaxScaler* из библиотеки *sklearn*:

```
import numpy as np
from sklearn.preprocessing import MinMaxScaler
dataset = np.array([[85, 72, 80], [64, 35, 26], [67, 48, 29], [100, 11, 102], [130, 14, 151]])
scaler = MinMaxScaler(feature_range=(-2.0, 1.5))
scaler.fit(dataset)
normalized_dataset = scaler.transform(dataset)
```

 Преобразование полезно, когда распределение данных неизвестно или не является гауссовым.

 Сохраняет форму исходного распределения данных.

 Недостаток: при наличии «выбросов» (больших значениях x_{\max}) нормализованные значения будут сконцентрированы в небольшом диапазоне около нуля.

 Недостаток: может не сохранять связи между векторами данных.

1 **Стандартизация** — это метод масштабирования, при котором данные центрируются вокруг среднего значения с единичным стандартным отклонением (дисперсией). Иногда процесс называют *Z-нормализацией*.

1 **Стандартизация** — это метод масштабирования, при котором данные центрируются вокруг среднего значения с единичным стандартным отклонением (дисперсией). Иногда процесс называют *Z-нормализацией*.

2 Преобразование данных определяется с помощью среднего (μ_x) и стандартного отклонения ($\sigma_x = \sqrt{D(x)}$):

$$\tilde{x}_i = \frac{x_i - \mu_x}{\sigma_x}. \quad (5)$$

❶ **Стандартизация** — это метод масштабирования, при котором данные центрируются вокруг среднего значения с единичным стандартным отклонением (дисперсией). Иногда процесс называют *Z-нормализацией*.

❷ Преобразование данных определяется с помощью среднего (μ_x) и стандартного отклонения ($\sigma_x = \sqrt{D(x)}$):

$$\tilde{x}_i = \frac{x_i - \mu_x}{\sigma_x}. \quad (5)$$

❸ Пример стандартизации набора (из трех признаков) в диапазон $[-2, 1.5]$ модулем *StandardScaler* из библиотеки *sklearn*:

```
import numpy as np
from sklearn.preprocessing import StandardScaler
dataset = np.array([[85, 72, 80], [64, 35, 26], [67, 48, 29], [100, 11, 102], [130, 14, 151]])
standard_scaler = StandardScaler()
standard_scaler.fit(dataset)
dataset_new = standard_scaler.transform(dataset)
```


1 **Стандартизация** — это метод масштабирования, при котором данные центрируются вокруг среднего значения с единичным стандартным отклонением (дисперсией). Иногда процесс называют *Z-нормализацией*.

2 Преобразование данных определяется с помощью среднего (μ_x) и стандартного отклонения ($\sigma_x = \sqrt{D(x)}$):

$$\tilde{x}_i = \frac{x_i - \mu_x}{\sigma_x}. \quad (5)$$

3 Пример стандартизации набора (из трех признаков) в диапазон $[-2, 1.5]$ модулем *StandardScaler* из библиотеки *sklearn*:

```
import numpy as np
from sklearn.preprocessing import StandardScaler
dataset = np.array([[85, 72, 80], [64, 35, 26], [67, 48, 29], [100, 11, 102], [130, 14, 151]])
standard_scaler = StandardScaler()
standard_scaler.fit(dataset)
dataset_new = standard_scaler.transform(dataset)
```

 Преобразование полезно для данных с нормальным распределением.

❶ **Стандартизация** — это метод масштабирования, при котором данные центрируются вокруг среднего значения с единичным стандартным отклонением (дисперсией). Иногда процесс называют *Z-нормализацией*.

❷ Преобразование данных определяется с помощью среднего (μ_x) и стандартного отклонения ($\sigma_x = \sqrt{D(x)}$):

$$\tilde{x}_i = \frac{x_i - \mu_x}{\sigma_x}. \quad (5)$$

❸ Пример стандартизации набора (из трех признаков) в диапазон $[-2, 1.5]$ модулем *StandardScaler* из библиотеки *sklearn*:

```
import numpy as np
from sklearn.preprocessing import StandardScaler
dataset = np.array([[85, 72, 80], [64, 35, 26], [67, 48, 29], [100, 11, 102], [130, 14, 151]])
standard_scaler = StandardScaler()
standard_scaler.fit(dataset)
dataset_new = standard_scaler.transform(dataset)
```

❶ Преобразование полезно для данных с нормальным распределением.

❶ Сохраняет связи между векторами данных.

1 **Стандартизация** — это метод масштабирования, при котором данные центрируются вокруг среднего значения с единичным стандартным отклонением (дисперсией). Иногда процесс называют *Z-нормализацией*.

2 Преобразование данных определяется с помощью среднего (μ_x) и стандартного отклонения ($\sigma_x = \sqrt{D(x)}$):

$$\tilde{x}_i = \frac{x_i - \mu_x}{\sigma_x}. \quad (5)$$

3 Пример стандартизации набора (из трех признаков) в диапазон $[-2, 1.5]$ модулем *StandardScaler* из библиотеки *sklearn*:

```
import numpy as np
from sklearn.preprocessing import StandardScaler
dataset = np.array([[85, 72, 80], [64, 35, 26], [67, 48, 29], [100, 11, 102], [130, 14, 151]])
standard_scaler = StandardScaler()
standard_scaler.fit(dataset)
dataset_new = standard_scaler.transform(dataset)
```

i Преобразование полезно для данных с нормальным распределением.

i Сохраняет связи между векторами данных.

i Преобразование менее чувствительно к выбросам, чем нормализация.

1 **Стандартизация** — это метод масштабирования, при котором данные центрируются вокруг среднего значения с единичным стандартным отклонением (дисперсией). Иногда процесс называют *Z-нормализацией*.

2 Преобразование данных определяется с помощью среднего (μ_x) и стандартного отклонения ($\sigma_x = \sqrt{D(x)}$):

$$\tilde{x}_i = \frac{x_i - \mu_x}{\sigma_x}. \quad (5)$$

3 Пример стандартизации набора (из трех признаков) в диапазон $[-2, 1.5]$ модулем *StandardScaler* из библиотеки *sklearn*:

```
import numpy as np
from sklearn.preprocessing import StandardScaler
dataset = np.array([[85, 72, 80], [64, 35, 26], [67, 48, 29], [100, 11, 102], [130, 14, 151]])
standard_scaler = StandardScaler()
standard_scaler.fit(dataset)
dataset_new = standard_scaler.transform(dataset)
```

i Преобразование полезно для данных с нормальным распределением.

i Сохраняет связи между векторами данных.

i Преобразование менее чувствительно к выбросам, чем нормализация.

i Недостаток: изменяет форму исходного распределения.



- 1 Репозиторий университета Ирвина (UCI Machine Learning Repository) — крупнейший репозиторий реальных и модельных задач машинного обучения (<http://archive.ics.uci.edu/ml>). Содержит несколько сотен задач, в основном классификации, из самых разных предметных областей.

- 1 Репозиторий университета Ирвина (UCI Machine Learning Repository) — крупнейший репозиторий реальных и модельных задач машинного обучения (<http://archive.ics.uci.edu/ml>). Содержит несколько сотен задач, в основном классификации, из самых разных предметных областей.
- 2 Free Music Archive (<https://freemusicarchive.org/>, <https://github.com/mdeff/fma>). Это открытый набор данных предназначен для анализа музыки и состоит из HQ-аудио, предварительно вычисленных характеристик. Размер почти 1000 ГБ.

- 1 Репозиторий университета Ирвина (UCI Machine Learning Repository) — крупнейший репозиторий реальных и модельных задач машинного обучения (<http://archive.ics.uci.edu/ml>). Содержит несколько сотен задач, в основном классификации, из самых разных предметных областей.
- 2 Free Music Archive (<https://freemusicarchive.org/>, <https://github.com/mdeff/fma>). Это открытый набор данных предназначен для анализа музыки и состоит из HQ-аудио, предварительно вычисленных характеристик. Размер почти 1000 ГБ.
- 3 Million Song Dataset (<http://millionsongdataset.com/>). Открытая коллекция характеристик и метаданных для миллиона треков. Набор не содержит аудио, а только извлеченные характеристики. Размер набора — около 280 ГБ.

- 1 Репозиторий университета Ирвина (UCI Machine Learning Repository) — крупнейший репозиторий реальных и модельных задач машинного обучения (<http://archive.ics.uci.edu/ml>). Содержит несколько сотен задач, в основном классификации, из самых разных предметных областей.
- 2 Free Music Archive (<https://freemusicarchive.org/>, <https://github.com/mdeff/fma>). Это открытый набор данных предназначен для анализа музыки и состоит из HQ-аудио, предварительно вычисленных характеристик. Размер почти 1000 ГБ.
- 3 Million Song Dataset (<http://millionsongdataset.com/>). Открытая коллекция характеристик и метаданных для миллиона треков. Набор не содержит аудио, а только извлеченные характеристики. Размер набора — около 280 ГБ.
- 4 AudioSet (<https://research.google.com/audioset/>, <https://github.com/audioset/ontology>). Содержит 632 класса звуковых событий и коллекцию из 2084320 помеченных вручную звуковых клипов длиной по 10 секунд, взятых из видео на YouTube.

- 1 Репозиторий университета Ирвина (UCI Machine Learning Repository) — крупнейший репозиторий реальных и модельных задач машинного обучения (<http://archive.ics.uci.edu/ml>). Содержит несколько сотен задач, в основном классификации, из самых разных предметных областей.
- 2 Free Music Archive (<https://freemusicarchive.org/>, <https://github.com/mdeff/fma>). Это открытый набор данных предназначен для анализа музыки и состоит из HQ-аудио, предварительно вычисленных характеристик. Размер почти 1000 ГБ.
- 3 Million Song Dataset (<http://millionsongdataset.com/>). Открытая коллекция характеристик и метаданных для миллиона треков. Набор не содержит аудио, а только извлеченные характеристики. Размер набора — около 280 ГБ.
- 4 AudioSet (<https://research.google.com/audioset/>, <https://github.com/audioset/ontology>). Содержит 632 класса звуковых событий и коллекцию из 2084320 помеченных вручную звуковых клипов длиной по 10 секунд, взятых из видео на YouTube.
- 5 The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) (<https://zenodo.org/record/1188976>). Аудиовизуальная база данных эмоциональной речи, содержит 7356 файлов, размер — 24, 8 ГБ.

- 1 Репозиторий университета Ирвина (UCI Machine Learning Repository) — крупнейший репозиторий реальных и модельных задач машинного обучения (<http://archive.ics.uci.edu/ml>). Содержит несколько сотен задач, в основном классификации, из самых разных предметных областей.
- 2 Free Music Archive (<https://freemusicarchive.org/>, <https://github.com/mdeff/fma>). Это открытый набор данных предназначен для анализа музыки и состоит из HQ-аудио, предварительно вычисленных характеристик. Размер почти 1000 ГБ.
- 3 Million Song Dataset (<http://millionsongdataset.com/>). Открытая коллекция характеристик и метаданных для миллиона треков. Набор не содержит аудио, а только извлеченные характеристики. Размер набора — около 280 ГБ.
- 4 AudioSet (<https://research.google.com/audioset/>, <https://github.com/audioset/ontology>). Содержит 632 класса звуковых событий и коллекцию из 2084320 помеченных вручную звуковых клипов длиной по 10 секунд, взятых из видео на YouTube.
- 5 The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) (<https://zenodo.org/record/1188976>). Аудиовизуальная база данных эмоциональной речи, содержит 7356 файлов, размер — 24, 8 ГБ.
- 6 База данных MNIST (Modified National Institute of Standards and Technology) (<http://yann.lecun.com/exdb/mnist/>). Содержит 60 тысяч картинок рукописных цифр, с размером изображений 28×28 пикселей. Это небольшая база — отлично подходит для учебных целей. Классификаторы получаются небольшими, и для их обучения не нужны огромные вычислительные мощности, достаточно ноутбука. При этом, база наглядна — после обучения классификатора можно попробовать распознать свои рукописные цифры и проанализировать качество распознавания.

- 1 Репозиторий университета Ирвина (UCI Machine Learning Repository) — крупнейший репозиторий реальных и модельных задач машинного обучения (<http://archive.ics.uci.edu/ml>). Содержит несколько сотен задач, в основном классификации, из самых разных предметных областей.
- 2 Free Music Archive (<https://freemusicarchive.org/>, <https://github.com/mdeff/fma>). Это открытый набор данных предназначен для анализа музыки и состоит из HQ-аудио, предварительно вычисленных характеристик. Размер почти 1000 ГБ.
- 3 Million Song Dataset (<http://millionsongdataset.com/>). Открытая коллекция характеристик и метаданных для миллиона треков. Набор не содержит аудио, а только извлеченные характеристики. Размер набора — около 280 ГБ.
- 4 AudioSet (<https://research.google.com/audioset/>, <https://github.com/audioset/ontology>). Содержит 632 класса звуковых событий и коллекцию из 2084320 помеченных вручную звуковых клипов длиной по 10 секунд, взятых из видео на YouTube.
- 5 The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) (<https://zenodo.org/record/1188976>). Аудиовизуальная база данных эмоциональной речи, содержит 7356 файлов, размер — 24, 8 ГБ.
- 6 База данных MNIST (Modified National Institute of Standards and Technology) (<http://yann.lecun.com/exdb/mnist/>). Содержит 60 тысяч картинок рукописных цифр, с размером изображений 28×28 пикселей. Это небольшая база — отлично подходит для учебных целей. Классификаторы получаются небольшими, и для их обучения не нужны огромные вычислительные мощности, достаточно ноутбука. При этом, база наглядна — после обучения классификатора можно попробовать распознать свои рукописные цифры и проанализировать качество распознавания.
- 7 База данных MovieLens (<https://grouplens.org/datasets/movielens/>). Содержит 26 миллионов оценок и 750000 тегов для 45 тысяч фильмов, поставленных 270000 пользователей. Это отличная база для обучения построению рекомендательных систем

- 1 Репозиторий университета Ирвина (UCI Machine Learning Repository) — крупнейший репозиторий реальных и модельных задач машинного обучения (<http://archive.ics.uci.edu/ml>). Содержит несколько сотен задач, в основном классификации, из самых разных предметных областей.
- 2 Free Music Archive (<https://freemusicarchive.org/>, <https://github.com/mdeff/fma>). Это открытый набор данных предназначен для анализа музыки и состоит из HQ-аудио, предварительно вычисленных характеристик. Размер почти 1000 ГБ.
- 3 Million Song Dataset (<http://millionsongdataset.com/>). Открытая коллекция характеристик и метаданных для миллиона треков. Набор не содержит аудио, а только извлеченные характеристики. Размер набора — около 280 ГБ.
- 4 AudioSet (<https://research.google.com/audioset/>, <https://github.com/audioset/ontology>). Содержит 632 класса звуковых событий и коллекцию из 2084320 помеченных вручную звуковых клипов длиной по 10 секунд, взятых из видео на YouTube.
- 5 The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) (<https://zenodo.org/record/1188976>). Аудиовизуальная база данных эмоциональной речи, содержит 7356 файлов, размер — 24, 8 ГБ.
- 6 База данных MNIST (Modified National Institute of Standards and Technology) (<http://yann.lecun.com/exdb/mnist/>). Содержит 60 тысяч картинок рукописных цифр, с размером изображений 28×28 пикселей. Это небольшая база — отлично подходит для учебных целей. Классификаторы получаются небольшими, и для их обучения не нужны огромные вычислительные мощности, достаточно ноутбука. При этом, база наглядна — после обучения классификатора можно попробовать распознать свои рукописные цифры и проанализировать качество распознавания.
- 7 База данных MovieLens (<https://grouplens.org/datasets/movielens/>). Содержит 26 миллионов оценок и 750000 тегов для 45 тысяч фильмов, поставленных 270000 пользователей. Это отличная база для обучения построению рекомендательных систем
- 8 Mivia Audio Events Dataset (<https://mivia.unisa.it/datasets/audio-analysis/mivia-audio-events/>). Включает 6000 событий, таких как разбиение стекла, выстрелы и крики, разделенных на обучающий набор из 4200 событий и тестовый — из 1800.

- 1 Репозиторий университета Ирвина (UCI Machine Learning Repository) — крупнейший репозиторий реальных и модельных задач машинного обучения (<http://archive.ics.uci.edu/ml>). Содержит несколько сотен задач, в основном классификации, из самых разных предметных областей.
- 2 Free Music Archive (<https://freemusicarchive.org/>, <https://github.com/mdeff/fma>). Это открытый набор данных предназначен для анализа музыки и состоит из HQ-аудио, предварительно вычисленных характеристик. Размер почти 1000 ГБ.
- 3 Million Song Dataset (<http://millionsongdataset.com/>). Открытая коллекция характеристик и метаданных для миллиона треков. Набор не содержит аудио, а только извлеченные характеристики. Размер набора — около 280 ГБ.
- 4 AudioSet (<https://research.google.com/audioset/>, <https://github.com/audioset/ontology>). Содержит 632 класса звуковых событий и коллекцию из 2084320 помеченных вручную звуковых клипов длиной по 10 секунд, взятых из видео на YouTube.
- 5 The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) (<https://zenodo.org/record/1188976>). Аудиовизуальная база данных эмоциональной речи, содержит 7356 файлов, размер — 24, 8 ГБ.
- 6 База данных MNIST (Modified National Institute of Standards and Technology) (<http://yann.lecun.com/exdb/mnist/>). Содержит 60 тысяч картинок рукописных цифр, с размером изображений 28×28 пикселей. Это небольшая база — отлично подходит для учебных целей. Классификаторы получаются небольшими, и для их обучения не нужны огромные вычислительные мощности, достаточно ноутбука. При этом, база наглядна — после обучения классификатора можно попробовать распознать свои рукописные цифры и проанализировать качество распознавания.
- 7 База данных MovieLens (<https://grouplens.org/datasets/movielens/>). Содержит 26 миллионов оценок и 750000 тегов для 45 тысяч фильмов, поставленных 270000 пользователей. Это отличная база для обучения построению рекомендательных систем
- 8 Mivia Audio Events Dataset (<https://mivia.unisa.it/datasets/audio-analysis/mivia-audio-events/>). Включает 6000 событий, таких как разбиение стекла, выстрелы и крики, разделенных на обучающий набор из 4200 событий и тестовый — из 1800.
- 9 База акустических событий последнего DCASE Challenge (<https://dcase.community/challenge2021/task-acoustic-scene-classification>). Каждый год для очередного соревнования создается новый набор данных.

- 1 Репозиторий университета Ирвина (UCI Machine Learning Repository) — крупнейший репозиторий реальных и модельных задач машинного обучения (<http://archive.ics.uci.edu/ml>). Содержит несколько сотен задач, в основном классификации, из самых разных предметных областей.
- 2 Free Music Archive (<https://freemusicarchive.org/>, <https://github.com/mdeff/fma>). Это открытый набор данных предназначен для анализа музыки и состоит из HQ-аудио, предварительно вычисленных характеристик. Размер почти 1000 ГБ.
- 3 Million Song Dataset (<http://millionsongdataset.com/>). Открытая коллекция характеристик и метаданных для миллиона треков. Набор не содержит аудио, а только извлеченные характеристики. Размер набора — около 280 ГБ.
- 4 AudioSet (<https://research.google.com/audioset/>, <https://github.com/audioset/ontology>). Содержит 632 класса звуковых событий и коллекцию из 2084320 помеченных вручную звуковых клипов длиной по 10 секунд, взятых из видео на YouTube.
- 5 The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) (<https://zenodo.org/record/1188976>). Аудиовизуальная база данных эмоциональной речи, содержит 7356 файлов, размер — 24, 8 ГБ.
- 6 База данных MNIST (Modified National Institute of Standards and Technology) (<http://yann.lecun.com/exdb/mnist/>). Содержит 60 тысяч картинок рукописных цифр, с размером изображений 28×28 пикселей. Это небольшая база — отлично подходит для учебных целей. Классификаторы получаются небольшими, и для их обучения не нужны огромные вычислительные мощности, достаточно ноутбука. При этом, база наглядна — после обучения классификатора можно попробовать распознать свои рукописные цифры и проанализировать качество распознавания.
- 7 База данных MovieLens (<https://grouplens.org/datasets/movielens/>). Содержит 26 миллионов оценок и 750000 тегов для 45 тысяч фильмов, поставленных 270000 пользователей. Это отличная база для обучения построению рекомендательных систем
- 8 Mivia Audio Events Dataset (<https://mivia.unisa.it/datasets/audio-analysis/mivia-audio-events/>). Включает 6000 событий, таких как разбиение стекла, выстрелы и крики, разделенных на обучающий набор из 4200 событий и тестовый — из 1800.
- 9 База акустических событий последнего DCASE Challenge (<https://dcase.community/challenge2021/task-acoustic-scene-classification>). Каждый год для очередного соревнования создается новый набор данных.
- 10 European Soccer Database (<https://www.kaggle.com/datasets/hugomathien/soccer>). База европейского футбола: чемпионаты 11 европейских стран, более 25000 матчей, более 10000 игроков. Сезоны с 2008 по 2016 годы.

- 11 LSUN (<https://www.yf.io/p/lsun>) — dataset изображений, разбитых по сценам и категориям с частичной разметкой данных.

- 11 LSUN (<https://www.yf.io/p/lsun>) — dataset изображений, разбитых по сценам и категориям с частичной разметкой данных.
- 12 The Data Zoo: Center for Coastal Studies (<http://zoo.ucsd.edu/>). Различные dataset о морях и их биологических характеристиках составе.

- 11 LSUN (<https://www.yf.io/p/lsun>) — dataset изображений, разбитых по сценам и категориям с частичной разметкой данных.
- 12 The Data Zoo: Center for Coastal Studies (<http://zoo.ucsd.edu/>). Различные dataset о морях и их биологических характеристиках составе.
- 13 Stanford Dogs Dataset (<http://vision.stanford.edu/aditya86/ImageNetDogs/>). Набор данных содержит более 20000 аннотированных изображений 120 пород собак со всего мира.

- 11 LSUN (<https://www.yf.io/p/lsun>) — dataset изображений, разбитых по сценам и категориям с частичной разметкой данных.
- 12 The Data Zoo: Center for Coastal Studies (<http://zoo.ucsd.edu/>). Различные dataset о морях и их биологических характеристиках составе.
- 13 Stanford Dogs Dataset (<http://vision.stanford.edu/aditya86/ImageNetDogs/>). Набор данных содержит более 20000 аннотированных изображений 120 пород собак со всего мира.
- 14 Amazon Reviews (<https://snap.stanford.edu/data/web-Amazon.html>). Dataset содержит около 35 млн. отзывов с Amazon за 18 лет. Данные включают информацию о продукте и пользователе, оценки и сам текст отзыва.

- 11 LSUN (<https://www.yf.io/p/lsun>) — dataset изображений, разбитых по сценам и категориям с частичной разметкой данных.
- 12 The Data Zoo: Center for Coastal Studies (<http://zoo.ucsd.edu/>). Различные dataset о морях и их биологических характеристиках составе.
- 13 Stanford Dogs Dataset (<http://vision.stanford.edu/aditya86/ImageNetDogs/>). Набор данных содержит более 20000 аннотированных изображений 120 пород собак со всего мира.
- 14 Amazon Reviews (<https://snap.stanford.edu/data/web-Amazon.html>). Dataset содержит около 35 млн. отзывов с Amazon за 18 лет. Данные включают информацию о продукте и пользователе, оценки и сам текст отзыва.
- 15 Labelled Faces in the Wild (<http://vis-www.cs.umass.edu/lfw/>). Набор данных фотографий содержит более 13000 аннотированных изображений лиц, собранных из Интернета. Среди них 1680 изображенных людей имеют две и более различных фотографий.

- 11 LSUN (<https://www.yf.io/p/lsun>) — dataset изображений, разбитых по сценам и категориям с частичной разметкой данных.
- 12 The Data Zoo: Center for Coastal Studies (<http://zoo.ucsd.edu/>). Различные dataset о морях и их биологических характеристиках составе.
- 13 Stanford Dogs Dataset (<http://vision.stanford.edu/aditya86/ImageNetDogs/>). Набор данных содержит более 20000 аннотированных изображений 120 пород собак со всего мира.
- 14 Amazon Reviews (<https://snap.stanford.edu/data/web-Amazon.html>). Dataset содержит около 35 млн. отзывов с Amazon за 18 лет. Данные включают информацию о продукте и пользователе, оценки и сам текст отзыва.
- 15 Labelled Faces in the Wild (<http://vis-www.cs.umass.edu/lfw/>). Набор данных фотографий содержит более 13000 аннотированных изображений лиц, собранных из Интернета. Среди них 1680 изображенных людей имеют две и более различных фотографий.
- 16 Cityscape Dataset (<https://www.cityscapes-dataset.com/>) — dataset, содержащий аннотированные 3D записи более сотни уличных сцен в 50 различных городах.

- 11 LSUN (<https://www.yf.io/p/lsun>) — dataset изображений, разбитых по сценам и категориям с частичной разметкой данных.
- 12 The Data Zoo: Center for Coastal Studies (<http://zoo.ucsd.edu/>). Различные dataset о морях и их биологических характеристиках составе.
- 13 Stanford Dogs Dataset (<http://vision.stanford.edu/aditya86/ImageNetDogs/>). Набор данных содержит более 20000 аннотированных изображений 120 пород собак со всего мира.
- 14 Amazon Reviews (<https://snap.stanford.edu/data/web-Amazon.html>). Dataset содержит около 35 млн. отзывов с Amazon за 18 лет. Данные включают информацию о продукте и пользователе, оценки и сам текст отзыва.
- 15 Labelled Faces in the Wild (<http://vis-www.cs.umass.edu/lfw/>). Набор данных фотографий содержит более 13000 аннотированных изображений лиц, собранных из Интернета. Среди них 1680 изображенных людей имеют две и более различных фотографий.
- 16 Cityscape Dataset (<https://www.cityscapes-dataset.com/>) — dataset, содержащий аннотированные 3D записи более сотни уличных сцен в 50 различных городах.
- 17 The Boston Housing Dataset (<https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html>) — dataset содержит информацию о домах в Бостоне: количество квартир, стоимость аренды, индекс преступлений.

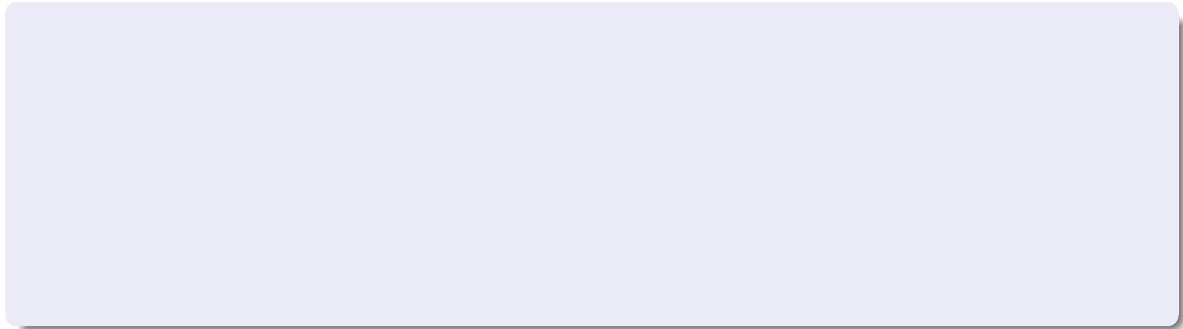
- 11 LSUN (<https://www.yf.io/p/lsun>) — dataset изображений, разбитых по сценам и категориям с частичной разметкой данных.
- 12 The Data Zoo: Center for Coastal Studies (<http://zoo.ucsd.edu/>). Различные dataset о морях и их биологических характеристиках составе.
- 13 Stanford Dogs Dataset (<http://vision.stanford.edu/aditya86/ImageNetDogs/>). Набор данных содержит более 20000 аннотированных изображений 120 пород собак со всего мира.
- 14 Amazon Reviews (<https://snap.stanford.edu/data/web-Amazon.html>). Dataset содержит около 35 млн. отзывов с Amazon за 18 лет. Данные включают информацию о продукте и пользователе, оценки и сам текст отзыва.
- 15 Labelled Faces in the Wild (<http://vis-www.cs.umass.edu/lfw/>). Набор данных фотографий содержит более 13000 аннотированных изображений лиц, собранных из Интернета. Среди них 1680 изображенных людей имеют две и более различных фотографий.
- 16 Cityscape Dataset (<https://www.cityscapes-dataset.com/>) — dataset, содержащий аннотированные 3D записи более сотни уличных сцен в 50 различных городах.
- 17 The Boston Housing Dataset (<https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html>) — dataset содержит информацию о домах в Бостоне: количество квартир, стоимость аренды, индекс преступлений.
- 18 Wine quality (<https://archive.ics.uci.edu/ml/datasets/wine+quality>) — dataset — содержит информацию о вине: 4898 записей с 14 параметрами.

Открытые данные (2)

- 11 LSUN (<https://www.yf.io/p/lsun>) — dataset изображений, разбитых по сценам и категориям с частичной разметкой данных.
- 12 The Data Zoo: Center for Coastal Studies (<http://zoo.ucsd.edu/>). Различные dataset о морях и их биологических характеристиках составе.
- 13 Stanford Dogs Dataset (<http://vision.stanford.edu/aditya86/ImageNetDogs/>). Набор данных содержит более 20000 аннотированных изображений 120 пород собак со всего мира.
- 14 Amazon Reviews (<https://snap.stanford.edu/data/web-Amazon.html>). Dataset содержит около 35 млн. отзывов с Amazon за 18 лет. Данные включают информацию о продукте и пользователе, оценки и сам текст отзыва.
- 15 Labelled Faces in the Wild (<http://vis-www.cs.umass.edu/lfw/>). Набор данных фотографий содержит более 13000 аннотированных изображений лиц, собранных из Интернета. Среди них 1680 изображенных людей имеют две и более различных фотографий.
- 16 Cityscape Dataset (<https://www.cityscapes-dataset.com/>) — dataset, содержащий аннотированные 3D записи более сотни уличных сцен в 50 различных городах.
- 17 The Boston Housing Dataset (<https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html>) — dataset содержит информацию о домах в Бостоне: количество квартир, стоимость аренды, индекс преступлений.
- 18 Wine quality (<https://archive.ics.uci.edu/ml/datasets/wine+quality>) — dataset — содержит информацию о вине: 4898 записей с 14 параметрами.
- 19 SOCR Data Dinov (http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_Dinov_020108_HeightsWeights) — dataset содержит 25000 записей о росте и весе молодых людей.

- 11 LSUN (<https://www.yf.io/p/lsun>) — dataset изображений, разбитых по сценам и категориям с частичной разметкой данных.
- 12 The Data Zoo: Center for Coastal Studies (<http://zoo.ucsd.edu/>). Различные dataset о морях и их биологических характеристиках составе.
- 13 Stanford Dogs Dataset (<http://vision.stanford.edu/aditya86/ImageNetDogs/>). Набор данных содержит более 20000 аннотированных изображений 120 пород собак со всего мира.
- 14 Amazon Reviews (<https://snap.stanford.edu/data/web-Amazon.html>). Dataset содержит около 35 млн. отзывов с Amazon за 18 лет. Данные включают информацию о продукте и пользователе, оценки и сам текст отзыва.
- 15 Labelled Faces in the Wild (<http://vis-www.cs.umass.edu/lfw/>). Набор данных фотографий содержит более 13000 аннотированных изображений лиц, собранных из Интернета. Среди них 1680 изображенных людей имеют две и более различных фотографий.
- 16 Cityscape Dataset (<https://www.cityscapes-dataset.com/>) — dataset, содержащий аннотированные 3D записи более сотни уличных сцен в 50 различных городах.
- 17 The Boston Housing Dataset (<https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html>) — dataset содержит информацию о домах в Бостоне: количество квартир, стоимость аренды, индекс преступлений.
- 18 Wine quality (<https://archive.ics.uci.edu/ml/datasets/wine+quality>) — dataset — содержит информацию о вине: 4898 записей с 14 параметрами.
- 19 SOCR Data Dinov (http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_Dinov_020108_HeightsWeights) — dataset содержит 25000 записей о росте и весе молодых людей.
- 20 UCI Spambase Dataset (<https://archive.ics.uci.edu/ml/datasets/Spambase>) — dataset для тренировки для обнаружения спама. Содержит 4601 писем с 57 параметрами.

- 11 LSUN (<https://www.yf.io/p/lsun>) — dataset изображений, разбитых по сценам и категориям с частичной разметкой данных.
- 12 The Data Zoo: Center for Coastal Studies (<http://zoo.ucsd.edu/>). Различные dataset о морях и их биологических характеристиках составе.
- 13 Stanford Dogs Dataset (<http://vision.stanford.edu/aditya86/ImageNetDogs/>). Набор данных содержит более 20000 аннотированных изображений 120 пород собак со всего мира.
- 14 Amazon Reviews (<https://snap.stanford.edu/data/web-Amazon.html>). Dataset содержит около 35 млн. отзывов с Amazon за 18 лет. Данные включают информацию о продукте и пользователе, оценки и сам текст отзыва.
- 15 Labelled Faces in the Wild (<http://vis-www.cs.umass.edu/lfw/>). Набор данных фотографий содержит более 13000 аннотированных изображений лиц, собранных из Интернета. Среди них 1680 изображенных людей имеют две и более различных фотографий.
- 16 Cityscape Dataset (<https://www.cityscapes-dataset.com/>) — dataset, содержащий аннотированные 3D записи более сотни уличных сцен в 50 различных городах.
- 17 The Boston Housing Dataset (<https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html>) — dataset содержит информацию о домах в Бостоне: количество квартир, стоимость аренды, индекс преступлений.
- 18 Wine quality (<https://archive.ics.uci.edu/ml/datasets/wine+quality>) — dataset — содержит информацию о вине: 4898 записей с 14 параметрами.
- 19 SOCR Data Dinov (http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_Dinov_020108_HeightsWeights) — dataset содержит 25000 записей о росте и весе молодых людей.
- 20 UCI Spambase Dataset (<https://archive.ics.uci.edu/ml/datasets/Spambase>) — dataset для тренировки для обнаружения спама. Содержит 4601 писем с 57 параметрами.
- 21 IMDB reviews (<http://ai.stanford.edu/~amaas/data/sentiment/>) — 25000 отзывов на фильмы в тренировочном наборе и 25000 — в тестовом.




22 German traffic sign recognition benchmark (<https://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset>) — 50000 изображений 43 дорожных знаков.

- 22 German traffic sign recognition benchmark (<https://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset>) — 50000 изображений 43 дорожных знаков.
- 23 London Datastore Portal (<https://data.london.gov.uk/>) — dataset о жизни людей в Лондоне.

- 22 German traffic sign recognition benchmark (<https://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset>) — 50000 изображений 43 дорожных знаков.
- 23 London Datastore Portal (<https://data.london.gov.uk/>) — dataset о жизни людей в Лондоне.
- 24 Librispeech Dataset (<http://www.openslr.org/12>) — содержит 1000 часов английской речи с разными акцентами.

- 22 German traffic sign recognition benchmark (<https://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset>) — 50000 изображений 43 дорожных знаков.
- 23 London Datastore Portal (<https://data.london.gov.uk/>) — dataset о жизни людей в Лондоне.
- 24 Librispeech Dataset (<http://www.openslr.org/12>) — содержит 1000 часов английской речи с разными акцентами.
- 25 Youtube 8M Dataset (<https://research.google.com/youtube8m/>) — размеченный набор данных видео, который содержит более 6 миллионов идентификаторов видео Youtube

- 22 German traffic sign recognition benchmark (<https://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset>) — 50000 изображений 43 дорожных знаков.
- 23 London Datastore Portal (<https://data.london.gov.uk/>) — dataset о жизни людей в Лондоне.
- 24 Librispeech Dataset (<http://www.openslr.org/12>) — содержит 1000 часов английской речи с разными акцентами.
- 25 Youtube 8M Dataset (<https://research.google.com/youtube8m/>) — размеченный набор данных видео, который содержит более 6 миллионов идентификаторов видео Youtube
- 26 MPII human pose dataset (<http://human-pose.mpi-inf.mpg.de/#>) — содержит 25000 изображений человеческих поз с аннотацией по суставам.

- 22 German traffic sign recognition benchmark (<https://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset>) — 50000 изображений 43 дорожных знаков.
- 23 London Datastore Portal (<https://data.london.gov.uk/>) — dataset о жизни людей в Лондоне.
- 24 Librispeech Dataset (<http://www.openslr.org/12>) — содержит 1000 часов английской речи с разными акцентами.
- 25 Youtube 8M Dataset (<https://research.google.com/youtube8m/>) — размеченный набор данных видео, который содержит более 6 миллионов идентификаторов видео Youtube
- 26 MPII human pose dataset (<http://human-pose.mpi-inf.mpg.de/#>) — содержит 25000 изображений человеческих поз с аннотацией по суставам.
-  Ирисы Фишера (<https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data>) — классический набор данных для задачи классификации. Состоит из 50 образцов каждого из трех видов ириса (*Iris Setosa*, *Iris Virginica* и *Iris Versicolor*). Для каждого образца есть 4 характеристики: длина и ширина чашелистиков и лепестков в сантиметрах (рисунок 2). Набор получил известность благодаря британскому статистiku и биологу Рональду Фишеру, в его работе 1936 г. «Использование многочисленных измерений в таксономических вопросах как пример линейного дискриминантного анализа».



- 22 German traffic sign recognition benchmark (<https://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset>) — 50000 изображений 43 дорожных знаков.
- 23 London Datastore Portal (<https://data.london.gov.uk/>) — dataset о жизни людей в Лондоне.
- 24 Librispeech Dataset (<http://www.openslr.org/12>) — содержит 1000 часов английской речи с разными акцентами.
- 25 Youtube 8M Dataset (<https://research.google.com/youtube8m/>) — размеченный набор данных видео, который содержит более 6 миллионов идентификаторов видео Youtube
- 26 MPII human pose dataset (<http://human-pose.mpi-inf.mpg.de/#>) — содержит 25000 изображений человеческих поз с аннотацией по суставам.
-  Ирисы Фишера (<https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data>) — классический набор данных для задачи классификации. Состоит из 50 образцов каждого из трех видов ириса (*Iris Setosa*, *Iris Virginica* и *Iris Versicolor*). Для каждого образца есть 4 характеристики: длина и ширина чашелистиков и лепестков в сантиметрах (рисунок 2). Набор получил известность благодаря британскому статистiku и биологу Рональду Фишеру, в его работе 1936 г. «Использование многочисленных измерений в таксономических вопросах как пример линейного дискриминантного анализа».




Рис. 2

Несколько примеров небольших тестовых датасетов, предоставляемых пакетом **sklearn.datasets**.

Несколько примеров небольших тестовых датасетов, предоставляемых пакетом **`sklearn.datasets`**.


 **Ирисы Фишера.** Загрузка: `sklearn.datasets.load_iris()`.

Несколько примеров небольших тестовых датасетов, предоставляемых пакетом **sklearn.datasets**.


 **Ирисы Фишера.** Загрузка: `sklearn.datasets.load_iris()`.

Документация: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html#sklearn.datasets.load_iris.


Несколько примеров небольших тестовых датасетов, предоставляемых пакетом **sklearn.datasets**.

 **Ирисы Фишера**. Загрузка: `sklearn.datasets.load_iris()`.


Документация: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html#sklearn.datasets.load_iris.

 **Диабет** — набор данных о диабете. Загрузка: `sklearn.datasets.load_diabetes()`.

Несколько примеров небольших тестовых датасетов, предоставляемых пакетом **sklearn.datasets**.


 **Ирисы Фишера**. Загрузка: `sklearn.datasets.load_iris()`.

Документация: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html#sklearn.datasets.load_iris.


 **Диабет** — набор данных о диабете. Загрузка: `sklearn.datasets.load_diabetes()`.

Документация: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_diabetes.html#sklearn.datasets.load_diabetes.


Несколько примеров небольших тестовых датасетов, предоставляемых пакетом **sklearn.datasets**.

 **Ирисы Фишера**. Загрузка: `sklearn.datasets.load_iris()`.


Документация: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html#sklearn.datasets.load_iris.

 **Диабет** — набор данных о диабете. Загрузка: `sklearn.datasets.load_diabetes()`.


Документация: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_diabetes.html#sklearn.datasets.load_diabetes.

 **Цифры** — набор данных цифр. Загрузка: `sklearn.datasets.load_digits()`.


Несколько примеров небольших тестовых датасетов, предоставляемых пакетом **sklearn.datasets**.

 **Ирисы Фишера**. Загрузка: `sklearn.datasets.load_iris()`.

Документация: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html#sklearn.datasets.load_iris.


 **Диабет** — набор данных о диабете. Загрузка: `sklearn.datasets.load_diabetes()`.

Документация: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_diabetes.html#sklearn.datasets.load_diabetes.


 **Цифры** — набор данных цифр. Загрузка: `sklearn.datasets.load_digits()`.

Документация: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html#sklearn.datasets.load_digits.


Несколько примеров небольших тестовых датасетов, предоставляемых пакетом **sklearn.datasets**.

 **Ирисы Фишера**. Загрузка: `sklearn.datasets.load_iris()`.


Документация: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html#sklearn.datasets.load_iris.

 **Диабет** — набор данных о диабете. Загрузка: `sklearn.datasets.load_diabetes()`.


Документация: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_diabetes.html#sklearn.datasets.load_diabetes.

 **Цифры** — набор данных цифр. Загрузка: `sklearn.datasets.load_digits()`.


Документация: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html#sklearn.datasets.load_digits.

 **Медицина** — набор данных о раке молочной железы в штате Висконсин. Загрузка: `sklearn.datasets.load_breast_cancer()`.


Несколько примеров небольших тестовых датасетов, предоставляемых пакетом **sklearn.datasets**.

 **Ирисы Фишера**. Загрузка: `sklearn.datasets.load_iris()`.


Документация: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html#sklearn.datasets.load_iris.

 **Диабет** — набор данных о диабете. Загрузка: `sklearn.datasets.load_diabetes()`.

Документация: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_diabetes.html#sklearn.datasets.load_diabetes.


 **Цифры** — набор данных цифр. Загрузка: `sklearn.datasets.load_digits()`.

Документация: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html#sklearn.datasets.load_digits.


 **Медицина** — набор данных о раке молочной железы в штате Висконсин. Загрузка: `sklearn.datasets.load_breast_cancer()`.

Документация: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html#sklearn.datasets.load_breast_cancer.


Несколько примеров небольших тестовых датасетов, предоставляемых пакетом **sklearn.datasets**.

 **Ирисы Фишера**. Загрузка: `sklearn.datasets.load_iris()`.


Документация: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html#sklearn.datasets.load_iris.

 **Диабет** — набор данных о диабете. Загрузка: `sklearn.datasets.load_diabetes()`.


Документация: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_diabetes.html#sklearn.datasets.load_diabetes.

 **Цифры** — набор данных цифр. Загрузка: `sklearn.datasets.load_digits()`.


Документация: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html#sklearn.datasets.load_digits.

 **Медицина** — набор данных о раке молочной железы в штате Висконсин. Загрузка: `sklearn.datasets.load_breast_cancer()`.


Документация: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html#sklearn.datasets.load_breast_cancer.

 **Вино** — набор данных о вине. Загрузка: `sklearn.datasets.load_wine()`.


Несколько примеров небольших тестовых датасетов, предоставляемых пакетом **sklearn.datasets**.

 **Ирисы Фишера**. Загрузка: `sklearn.datasets.load_iris()`.


Документация: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html#sklearn.datasets.load_iris.

 **Диабет** — набор данных о диабете. Загрузка: `sklearn.datasets.load_diabetes()`.


Документация: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_diabetes.html#sklearn.datasets.load_diabetes.

 **Цифры** — набор данных цифр. Загрузка: `sklearn.datasets.load_digits()`.

Документация: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html#sklearn.datasets.load_digits.

 **Медицина** — набор данных о раке молочной железы в штате Висконсин. Загрузка: `sklearn.datasets.load_breast_cancer()`.

Документация: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html#sklearn.datasets.load_breast_cancer.

 **Вино** — набор данных о вине. Загрузка: `sklearn.datasets.load_wine()`.

Документация: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_wine.html#sklearn.datasets.load_wine.

Ресурсы для поиска

- 1 *Google Dataset Search* (<https://datasetsearch.research.google.com/>) — поиск dataset по всей сети интернет. Дополняет Google Scholar, поисковую систему для академических исследований и отчетов. При желании можно добавить свои наборы данных.

Ресурсы для поиска

- 1 *Google Dataset Search* (<https://datasetsearch.research.google.com/>) — поиск dataset по всей сети интернет. Дополняет Google Scholar, поисковую систему для академических исследований и отчетов. При желании можно добавить свои наборы данных.
- 2 *Платформа Kaggle* (<https://www.kaggle.com/datasets>) — место встречи всех любителей соревнований по машинному обучению.

Ресурсы для поиска

- 1 *Google Dataset Search* (<https://datasetsearch.research.google.com/>) — поиск dataset по всей сети интернет. Дополняет Google Scholar, поисковую систему для академических исследований и отчетов. При желании можно добавить свои наборы данных.
- 2 *Платформа Kaggle* (<https://www.kaggle.com/datasets>) — место встречи всех любителей соревнований по машинному обучению.
- 3 *VisualData* (<https://www.visualdata.io/>) — поиск dataset для задач машинного зрения, с удобной классификацией по категориям.

Ресурсы для поиска


- 1 *Google Dataset Search* (<https://datasetsearch.research.google.com/>) — поиск dataset по всей сети интернет. Дополняет Google Scholar, поисковую систему для академических исследований и отчетов. При желании можно добавить свои наборы данных.
- 2 *Платформа Kaggle* (<https://www.kaggle.com/datasets>) — место встречи всех любителей соревнований по машинному обучению.
- 3 *VisualData* (<https://www.visualdata.io/>) — поиск dataset для задач машинного зрения, с удобной классификацией по категориям.
- 4 *DATA USA* (<https://datausa.io/>) — полный набор по общедоступным данным США с визуализацией, описанием и инфографикой.


Ресурсы для поиска


- 1 *Google Dataset Search* (<https://datasetsearch.research.google.com/>) — поиск dataset по всей сети интернет. Дополняет Google Scholar, поисковую систему для академических исследований и отчетов. При желании можно добавить свои наборы данных.
- 2 *Платформа Kaggle* (<https://www.kaggle.com/datasets>) — место встречи всех любителей соревнований по машинному обучению.
- 3 *VisualData* (<https://www.visualdata.io/>) — поиск dataset для задач машинного зрения, с удобной классификацией по категориям.
- 4 *DATA USA* (<https://datausa.io/>) — полный набор по общедоступным данным США с визуализацией, описанием и инфографикой.


Хорошая обзорная статья по ресурсам: <https://habr.com/ru/post/452392/>

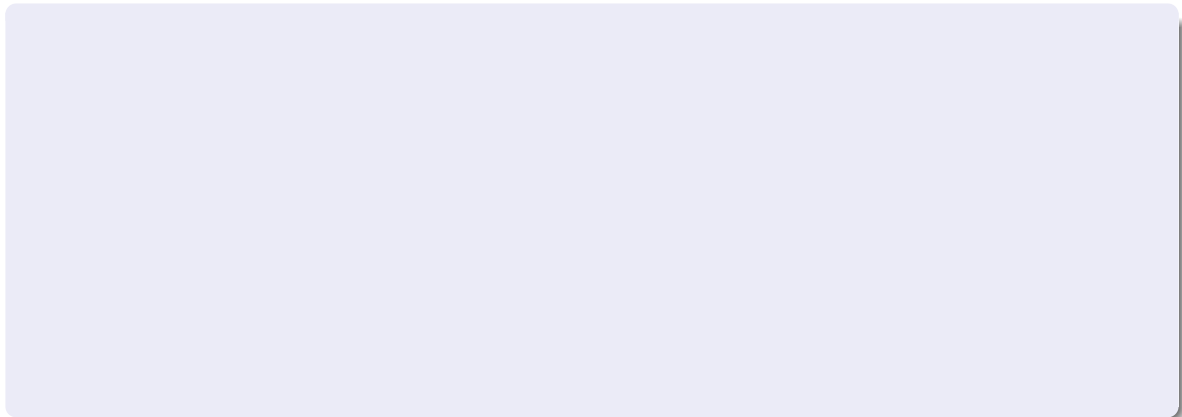
- 1 MachineLearning.ru (www.MachineLearning.ru) — крупнейший русскоязычный информационно-аналитический ресурс, посвященный машинному обучению и интеллектуальному анализу данных. Работает с 2008 года.

- 1 *MachineLearning.ru* (www.MachineLearning.ru) — крупнейший русскоязычный информационно-аналитический ресурс, посвященный машинному обучению и интеллектуальному анализу данных. Работает с 2008 года.
 - 2 *Kaggle* (<https://www.kaggle.com/>) — англоязычная платформа организации соревнований по исследованию данных, а также социальная сеть специалистов по обработке данных и машинному обучению.
-  Далее про Kaggle будет более подробно.

- 1 *MachineLearning.ru* (www.MachineLearning.ru) — крупнейший русскоязычный информационно-аналитический ресурс, посвященный машинному обучению и интеллектуальному анализу данных. Работает с 2008 года.
- 2 *Kaggle* (<https://www.kaggle.com/>) — англоязычная платформа организации соревнований по исследованию данных, а также социальная сеть специалистов по обработке данных и машинному обучению.
 Далее про Kaggle будет более подробно.
- 3 *ML Boot Camp* (<https://mlbootcamp.ru/article/tutorial/>) — русскоязычная платформа, где любой желающий может попробовать себя в Machine Learning (аналог Kaggle).

- 1 *MachineLearning.ru* (www.MachineLearning.ru) — крупнейший русскоязычный информационно-аналитический ресурс, посвященный машинному обучению и интеллектуальному анализу данных. Работает с 2008 года.
- 2 *Kaggle* (<https://www.kaggle.com/>) — англоязычная платформа организации соревнований по исследованию данных, а также социальная сеть специалистов по обработке данных и машинному обучению.
 Далее про Kaggle будет более подробно.
- 3 *ML Boot Camp* (<https://mlbootcamp.ru/article/tutorial/>) — русскоязычная платформа, где любой желающий может попробовать себя в Machine Learning (аналог Kaggle).
- 4 *OpenDataScience RU* (https://t.me/ods_ru) — Telegram канал крупнейшего русскоязычного сообщества специалистов по Data Science и Machine Learning. Новости отрасли, новые разработки и анонсы мероприятий.

- 1 *MachineLearning.ru* (www.MachineLearning.ru) — крупнейший русскоязычный информационно-аналитический ресурс, посвященный машинному обучению и интеллектуальному анализу данных. Работает с 2008 года.
- 2 *Kaggle* (<https://www.kaggle.com/>) — англоязычная платформа организации соревнований по исследованию данных, а также социальная сеть специалистов по обработке данных и машинному обучению.
 Далее про Kaggle будет более подробно.
- 3 *ML Boot Camp* (<https://mlbootcamp.ru/article/tutorial/>) — русскоязычная платформа, где любой желающий может попробовать себя в Machine Learning (аналог Kaggle).
- 4 *OpenDataScience RU* (https://t.me/ods_ru) — Telegram канал крупнейшего русскоязычного сообщества специалистов по Data Science и Machine Learning. Новости отрасли, новые разработки и анонсы мероприятий.
- 5 *Хабр* (<https://habr.com>) — крупнейший в Европе ресурс для IT-специалистов. Здесь можно найти интересные статьи и обсуждения по машинному обучению для разработчиков любого уровня.



- 1 Kaggle — одна из наиболее известных платформ для проведения соревнований по Data Science.

- 1 Kaggle — одна из наиболее известных платформ для проведения соревнований по Data Science.
- 2 В каждом соревновании организаторы выкладывают описание задачи, данные для решения этой задачи, метрику, по которой будет оцениваться решение и устанавливают сроки и призы.

- 1 Kaggle — одна из наиболее известных платформ для проведения соревнований по Data Science.
- 2 В каждом соревновании организаторы выкладывают описание задачи, данные для решения этой задачи, метрику, по которой будет оцениваться решение и устанавливают сроки и призы.
- 3 Участникам дается от 3 до 5 попыток в день на отправку своего варианта решения.

- 1 Kaggle — одна из наиболее известных платформ для проведения соревнований по Data Science.
- 2 В каждом соревновании организаторы выкладывают описание задачи, данные для решения этой задачи, метрику, по которой будет оцениваться решение и устанавливают сроки и призы.
- 3 Участникам дается от 3 до 5 попыток в день на отправку своего варианта решения.
- 4 Данные делятся на тренировочную выборку (**train**) и тестовую (**test**). Для тренировочной части известно значение целевой переменной (**target**), для тестовой — нет.

- 1 Kaggle — одна из наиболее известных платформ для проведения соревнований по Data Science.
- 2 В каждом соревновании организаторы выкладывают описание задачи, данные для решения этой задачи, метрику, по которой будет оцениваться решение и устанавливают сроки и призы.
- 3 Участникам дается от 3 до 5 попыток в день на отправку своего варианта решения.
- 4 Данные делятся на тренировочную выборку (**train**) и тестовую (**test**). Для тренировочной части известно значение целевой переменной (**target**), для тестовой — нет.
- 5 Задача участника создать модель, которая, будучи обучена на тренировочной части данных выдаст максимальный результат на тестовой выборке.

- 1 Kaggle — одна из наиболее известных платформ для проведения соревнований по Data Science.
- 2 В каждом соревновании организаторы выкладывают описание задачи, данные для решения этой задачи, метрику, по которой будет оцениваться решение и устанавливают сроки и призы.
- 3 Участникам дается от 3 до 5 попыток в день на отправку своего варианта решения.
- 4 Данные делятся на тренировочную выборку (**train**) и тестовую (**test**). Для тренировочной части известно значение целевой переменной (**target**), для тестовой — нет.
- 5 Задача участника создать модель, которая, будучи обучена на тренировочной части данных выдаст максимальный результат на тестовой выборке.
- 6 Каждый участник делает предсказания для тестовой выборки и отправляет результат на Kaggle, далее робот (которому известна целевая переменная для теста) оценивает присланный результат, который отображается в разделе **Leaderboard**.

- 1 Kaggle — одна из наиболее известных платформ для проведения соревнований по Data Science.
- 2 В каждом соревновании организаторы выкладывают описание задачи, данные для решения этой задачи, метрику, по которой будет оцениваться решение и устанавливают сроки и призы.
- 3 Участникам дается от 3 до 5 попыток в день на отправку своего варианта решения.
- 4 Данные делятся на тренировочную выборку (**train**) и тестовую (**test**). Для тренировочной части известно значение целевой переменной (**target**), для тестовой — нет.
- 5 Задача участника создать модель, которая, будучи обучена на тренировочной части данных выдаст максимальный результат на тестовой выборке.
- 6 Каждый участник делает предсказания для тестовой выборки и отправляет результат на Kaggle, далее робот (которому известна целевая переменная для теста) оценивает присланный результат, который отображается в разделе **Leaderboard**.
- 7 Тестовые данные делятся в определенной пропорции на публичную (**public**) и приватную (**private**) части.

- 1 Kaggle — одна из наиболее известных платформ для проведения соревнований по Data Science.
- 2 В каждом соревновании организаторы выкладывают описание задачи, данные для решения этой задачи, метрику, по которой будет оцениваться решение и устанавливают сроки и призы.
- 3 Участникам дается от 3 до 5 попыток в день на отправку своего варианта решения.
- 4 Данные делятся на тренировочную выборку (**train**) и тестовую (**test**). Для тренировочной части известно значение целевой переменной (**target**), для тестовой — нет.
- 5 Задача участника создать модель, которая, будучи обучена на тренировочной части данных выдаст максимальный результат на тестовой выборке.
- 6 Каждый участник делает предсказания для тестовой выборки и отправляет результат на Kaggle, далее робот (которому известна целевая переменная для теста) оценивает присланный результат, который отображается в разделе **Leaderboard**.
- 7 Тестовые данные делятся в определенной пропорции на публичную (**public**) и приватную (**private**) части.
- 8 В течение соревнования присланное решение оценивается, согласно установленной организаторами метрике, на публичной части данных и выкладывается на Leaderboard — по которому участники могут оценивать качество своих моделей.

Что такое Kaggle

- 1 Kaggle — одна из наиболее известных платформ для проведения соревнований по Data Science.
- 2 В каждом соревновании организаторы выкладывают описание задачи, данные для решения этой задачи, метрику, по которой будет оцениваться решение и устанавливают сроки и призы.
- 3 Участникам дается от 3 до 5 попыток в день на отправку своего варианта решения.
- 4 Данные делятся на тренировочную выборку (**train**) и тестовую (**test**). Для тренировочной части известно значение целевой переменной (**target**), для тестовой — нет.
- 5 Задача участника создать модель, которая, будучи обучена на тренировочной части данных выдаст максимальный результат на тестовой выборке.
- 6 Каждый участник делает предсказания для тестовой выборки и отправляет результат на Kaggle, далее робот (которому известна целевая переменная для теста) оценивает присланный результат, который отображается в разделе **Leaderboard**.
- 7 Тестовые данные делятся в определенной пропорции на публичную (**public**) и приватную (**private**) части.
- 8 В течение соревнования присланное решение оценивается, согласно установленной организаторами метрике, на публичной части данных и выкладывается на Leaderboard — по которому участники могут оценивать качество своих моделей.
- 9 Окончательное решение оценивается на приватной части тестовых данных и результат попадает на приватный Leaderboard, который доступен только после окончания соревнования.

Интерфейс

Интерфейс

- 1 **Overview.** Краткое описание проблемы, метрики, оценки, призы и временная шкала.

Интерфейс

- 1 **Overview.** Краткое описание проблемы, метрики, оценки, призы и временная шкала.
- 2 **Data.** Все данные, необходимые для участия в конкурсе, другие данные не допускаются.

Интерфейс

- 1 **Overview.** Краткое описание проблемы, метрики, оценки, призы и временная шкала.
- 2 **Data.** Все данные, необходимые для участия в конкурсе, другие данные не допускаются.
- 3 **Kernels.** Предыдущие работы, сделанные вами или другими участниками. Очень важный элемент для соревнования. Можно посмотреть другие скрипты и notebooks, а потом скопировать (называется «Forking»), чтобы изменить их и запустить.

Интерфейс

- 1 **Overview.** Краткое описание проблемы, метрики, оценки, призы и временная шкала.
- 2 **Data.** Все данные, необходимые для участия в конкурсе, другие данные не допускаются.
- 3 **Kernels.** Предыдущие работы, сделанные вами или другими участниками. Очень важный элемент для соревнования. Можно посмотреть другие скрипты и notebooks, а потом скопировать (называется «Forking»), чтобы изменить их и запустить.
- 4 **Discussion.** Обсуждения, в которых участвуют организаторы соревнования и участники. Здесь можно задать уточняющие вопросы или почерпнуть новую информацию из ответов другим участникам.

Интерфейс

- 1 **Overview.** Краткое описание проблемы, метрики, оценки, призы и временная шкала.
- 2 **Data.** Все данные, необходимые для участия в конкурсе, другие данные не допускаются.
- 3 **Kernels.** Предыдущие работы, сделанные вами или другими участниками. Очень важный элемент для соревнования. Можно посмотреть другие скрипты и notebooks, а потом скопировать (называется «Forking»), чтобы изменить их и запустить.
- 4 **Discussion.** Обсуждения, в которых участвуют организаторы соревнования и участники. Здесь можно задать уточняющие вопросы или почерпнуть новую информацию из ответов другим участникам.
- 5 **Leaderboard.** Можно посмотреть, кто в топе, и каков ваш рейтинг.

Интерфейс

- 1 **Overview.** Краткое описание проблемы, метрики, оценки, призы и временная шкала.
- 2 **Data.** Все данные, необходимые для участия в конкурсе, другие данные не допускаются.
- 3 **Kernels.** Предыдущие работы, сделанные вами или другими участниками. Очень важный элемент для соревнования. Можно посмотреть другие скрипты и notebooks, а потом скопировать (называется «Forking»), чтобы изменить их и запустить.
- 4 **Discussion.** Обсуждения, в которых участвуют организаторы соревнования и участники. Здесь можно задать уточняющие вопросы или почерпнуть новую информацию из ответов другим участникам.
- 5 **Leaderboard.** Можно посмотреть, кто в топе, и каков ваш рейтинг.
- 6 **Rules.** Здесь находятся правила — их нужно знать.

Интерфейс

- 1 **Overview.** Краткое описание проблемы, метрики, оценки, призы и временная шкала.
- 2 **Data.** Все данные, необходимые для участия в конкурсе, другие данные не допускаются.
- 3 **Kernels.** Предыдущие работы, сделанные вами или другими участниками. Очень важный элемент для соревнования. Можно посмотреть другие скрипты и notebooks, а потом скопировать (называется «Forking»), чтобы изменить их и запустить.
- 4 **Discussion.** Обсуждения, в которых участвуют организаторы соревнования и участники. Здесь можно задать уточняющие вопросы или почерпнуть новую информацию из ответов другим участникам.
- 5 **Leaderboard.** Можно посмотреть, кто в топе, и каков ваш рейтинг.
- 6 **Rules.** Здесь находятся правила — их нужно знать.
- 7 **Team.** Управление своей командой.

Интерфейс

- 1 **Overview.** Краткое описание проблемы, метрики, оценки, призы и временная шкала.
- 2 **Data.** Все данные, необходимые для участия в конкурсе, другие данные не допускаются.
- 3 **Kernels.** Предыдущие работы, сделанные вами или другими участниками. Очень важный элемент для соревнования. Можно посмотреть другие скрипты и notebooks, а потом скопировать (называется «Forking»), чтобы изменить их и запустить.
- 4 **Discussion.** Обсуждения, в которых участвуют организаторы соревнования и участники. Здесь можно задать уточняющие вопросы или почерпнуть новую информацию из ответов другим участникам.
- 5 **Leaderboard.** Можно посмотреть, кто в топе, и каков ваш рейтинг.
- 6 **Rules.** Здесь находятся правила — их нужно знать.
- 7 **Team.** Управление своей командой.
- 8 **My Submissions.** Здесь ваши предыдущие материалы, можно выбрать финальный вариант, который будет участвовать в соревновании.

Интерфейс

- 1 **Overview.** Краткое описание проблемы, метрики, оценки, призы и временная шкала.
- 2 **Data.** Все данные, необходимые для участия в конкурсе, другие данные не допускаются.
- 3 **Kernels.** Предыдущие работы, сделанные вами или другими участниками. Очень важный элемент для соревнования. Можно посмотреть другие скрипты и notebooks, а потом скопировать (называется «Forking»), чтобы изменить их и запустить.
- 4 **Discussion.** Обсуждения, в которых участвуют организаторы соревнования и участники. Здесь можно задать уточняющие вопросы или почерпнуть новую информацию из ответов другим участникам.
- 5 **Leaderboard.** Можно посмотреть, кто в топе, и каков ваш рейтинг.
- 6 **Rules.** Здесь находятся правила — их нужно знать.
- 7 **Team.** Управление своей командой.
- 8 **My Submissions.** Здесь ваши предыдущие материалы, можно выбрать финальный вариант, который будет участвовать в соревновании.

Возможности

Интерфейс

- 1 **Overview.** Краткое описание проблемы, метрики, оценки, призы и временная шкала.
- 2 **Data.** Все данные, необходимые для участия в конкурсе, другие данные не допускаются.
- 3 **Kernels.** Предыдущие работы, сделанные вами или другими участниками. Очень важный элемент для соревнования. Можно посмотреть другие скрипты и notebooks, а потом скопировать (называется «Forking»), чтобы изменить их и запустить.
- 4 **Discussion.** Обсуждения, в которых участвуют организаторы соревнования и участники. Здесь можно задать уточняющие вопросы или почерпнуть новую информацию из ответов другим участникам.
- 5 **Leaderboard.** Можно посмотреть, кто в топе, и каков ваш рейтинг.
- 6 **Rules.** Здесь находятся правила — их нужно знать.
- 7 **Team.** Управление своей командой.
- 8 **My Submissions.** Здесь ваши предыдущие материалы, можно выбрать финальный вариант, который будет участвовать в соревновании.

Возможности

- D **Datasets.** Различные наборы данных, которые можно бесплатно загрузить. Здесь можно найти интересные данные для изучения или тестирования своих навыков машинного обучения.

Интерфейс

- 1 **Overview.** Краткое описание проблемы, метрики, оценки, призы и временная шкала.
- 2 **Data.** Все данные, необходимые для участия в конкурсе, другие данные не допускаются.
- 3 **Kernels.** Предыдущие работы, сделанные вами или другими участниками. Очень важный элемент для соревнования. Можно посмотреть другие скрипты и notebooks, а потом скопировать (называется «Forking»), чтобы изменить их и запустить.
- 4 **Discussion.** Обсуждения, в которых участвуют организаторы соревнования и участники. Здесь можно задать уточняющие вопросы или почерпнуть новую информацию из ответов другим участникам.
- 5 **Leaderboard.** Можно посмотреть, кто в топе, и каков ваш рейтинг.
- 6 **Rules.** Здесь находятся правила — их нужно знать.
- 7 **Team.** Управление своей командой.
- 8 **My Submissions.** Здесь ваши предыдущие материалы, можно выбрать финальный вариант, который будет участвовать в соревновании.

Возможности

- D **Datasets.** Различные наборы данных, которые можно бесплатно загрузить. Здесь можно найти интересные данные для изучения или тестирования своих навыков машинного обучения.
- C **Machine Learning Competitions.** Соревнования по машинному обучению) — хороший способ изучить новые алгоритмы и проверить свои способности с помощью интересных проблем, основанных на реальных данных.

Интерфейс

- 1 **Overview.** Краткое описание проблемы, метрики, оценки, призы и временная шкала.
- 2 **Data.** Все данные, необходимые для участия в конкурсе, другие данные не допускаются.
- 3 **Kernels.** Предыдущие работы, сделанные вами или другими участниками. Очень важный элемент для соревнования. Можно посмотреть другие скрипты и notebooks, а потом скопировать (называется «Forking»), чтобы изменить их и запустить.
- 4 **Discussion.** Обсуждения, в которых участвуют организаторы соревнования и участники. Здесь можно задать уточняющие вопросы или почерпнуть новую информацию из ответов другим участникам.
- 5 **Leaderboard.** Можно посмотреть, кто в топе, и каков ваш рейтинг.
- 6 **Rules.** Здесь находятся правила — их нужно знать.
- 7 **Team.** Управление своей командой.
- 8 **My Submissions.** Здесь ваши предыдущие материалы, можно выбрать финальный вариант, который будет участвовать в соревновании.

Возможности

- D **Datasets.** Различные наборы данных, которые можно бесплатно загрузить. Здесь можно найти интересные данные для изучения или тестирования своих навыков машинного обучения.
- C **Machine Learning Competitions.** Соревнования по машинному обучению) — хороший способ изучить новые алгоритмы и проверить свои способности с помощью интересных проблем, основанных на реальных данных.
- L **Learn.** Обучающие материалы по ML в Jupyter Notebooks.

- 1 SOCR Data Dinov (http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_Dinov_020108_HeightsWeights) — dataset содержит 25000 записей о росте и весе молодых людей.

- 1 SOCR Data Dinov (http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_Dinov_020108_HeightsWeights) — dataset содержит 25000 записей о росте и весе молодых людей.
- 2 Young People Survey (<https://www.kaggle.com/datasets/miroslavsabo/young-people-survey>). Набор данных о предпочтениях, интересах, привычках, мнениях молодежи.

- 1 SOCR Data Dinov (http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_Dinov_020108_HeightsWeights) — dataset содержит 25000 записей о росте и весе молодых людей.
- 2 Young People Survey (<https://www.kaggle.com/datasets/miroslavsabo/young-people-survey>). Набор данных о предпочтениях, интересах, привычках, мнениях молодежи.
- 3 World University Rankings (<https://www.kaggle.com/datasets/mylesoneill/world-university-rankings>). Данные о лучших университетах мира по нескольким рейтингам:

- ❶ SOCR Data Dinov (http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_Dinov_020108_HeightsWeights) — dataset содержит 25000 записей о росте и весе молодых людей.
- ❷ Young People Survey (<https://www.kaggle.com/datasets/miroslavsabo/young-people-survey>). Набор данных о предпочтениях, интересах, привычках, мнениях молодежи.
- ❸ World University Rankings (<https://www.kaggle.com/datasets/mylesoneill/world-university-rankings>). Данные о лучших университетах мира по нескольким рейтингам:
 - Times Higher Education World University Ranking (<https://www.timeshighereducation.com/world-university-rankings>),

- ❶ SOCR Data Dinov (http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_Dinov_020108_HeightsWeights) — dataset содержит 25000 записей о росте и весе молодых людей.
- ❷ Young People Survey (<https://www.kaggle.com/datasets/miroslavsabo/young-people-survey>). Набор данных о предпочтениях, интересах, привычках, мнениях молодежи.
- ❸ World University Rankings (<https://www.kaggle.com/datasets/mylesoneill/world-university-rankings>). Данные о лучших университетах мира по нескольким рейтингам:
 - Times Higher Education World University Ranking (<https://www.timeshighereducation.com/world-university-rankings>),
 - Shanghai Ranking (<http://www.shanghairanking.com/>) — в основном технический,

- ❶ SOCR Data Dinov (http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_Dinov_020108_HeightsWeights) — dataset содержит 25000 записей о росте и весе молодых людей.
- ❷ Young People Survey (<https://www.kaggle.com/datasets/miroslavsabo/young-people-survey>). Набор данных о предпочтениях, интересах, привычках, мнениях молодежи.
- ❸ World University Rankings (<https://www.kaggle.com/datasets/mylesoneill/world-university-rankings>). Данные о лучших университетах мира по нескольким рейтингам:
 - Times Higher Education World University Ranking (<https://www.timeshighereducation.com/world-university-rankings>),
 - Shanghai Ranking (<http://www.shanghairanking.com/>) — в основном технический,
 - Center for World University Rankings (<https://cwur.org/>) — арабский.

- ❶ SOCR Data Dinov (http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_Dinov_020108_HeightsWeights) — dataset содержит 25000 записей о росте и весе молодых людей.
- ❷ Young People Survey (<https://www.kaggle.com/datasets/miroslavsabo/young-people-survey>). Набор данных о предпочтениях, интересах, привычках, мнениях молодежи.
- ❸ World University Rankings (<https://www.kaggle.com/datasets/mylesoneill/world-university-rankings>). Данные о лучших университетах мира по нескольким рейтингам:
 - Times Higher Education World University Ranking (<https://www.timeshighereducation.com/world-university-rankings>),
 - Shanghai Ranking (<http://www.shanghairanking.com/>) — в основном технический,
 - Center for World University Rankings (<https://cwur.org/>) — арабский.
- ❹ Kaggle Machine Learning and Data Science Survey 2017 (<https://www.kaggle.com/datasets/kaggle/kaggle-survey-2017>) — большое описание текущего состояния данных и машинного обучения.

- ❶ SOCR Data Dinov (http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_Dinov_020108_HeightsWeights) — dataset содержит 25000 записей о росте и весе молодых людей.
- ❷ Young People Survey (<https://www.kaggle.com/datasets/miroslavsabo/young-people-survey>). Набор данных о предпочтениях, интересах, привычках, мнениях молодежи.
- ❸ World University Rankings (<https://www.kaggle.com/datasets/mylesoneill/world-university-rankings>). Данные о лучших университетах мира по нескольким рейтингам:
 - Times Higher Education World University Ranking (<https://www.timeshighereducation.com/world-university-rankings>),
 - Shanghai Ranking (<http://www.shanghairanking.com/>) — в основном технический,
 - Center for World University Rankings (<https://cwur.org/>) — арабский.
- ❹ Kaggle Machine Learning and Data Science Survey 2017 (<https://www.kaggle.com/datasets/kaggle/kaggle-survey-2017>) — большое описание текущего состояния данных и машинного обучения.
- ❺ Gender Recognition by Voice (<https://www.kaggle.com/datasets/primaryobjects/voicegender>). База данных для идентификации пола по голосу на основе акустических свойств речи. Набор данных состоит из 3168 записей мужчин и женщин.

- 1 SOCR Data Dinov (http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_Dinov_020108_HeightsWeights) — dataset содержит 25000 записей о росте и весе молодых людей.
- 2 Young People Survey (<https://www.kaggle.com/datasets/miroslavsabo/young-people-survey>). Набор данных о предпочтениях, интересах, привычках, мнениях молодежи.
- 3 World University Rankings (<https://www.kaggle.com/datasets/mylesoneill/world-university-rankings>). Данные о лучших университетах мира по нескольким рейтингам:
 - Times Higher Education World University Ranking (<https://www.timeshighereducation.com/world-university-rankings>),
 - Shanghai Ranking (<http://www.shanghairanking.com/>) — в основном технический,
 - Center for World University Rankings (<https://cwur.org/>) — арабский.
- 4 Kaggle Machine Learning and Data Science Survey 2017 (<https://www.kaggle.com/datasets/kaggle/kaggle-survey-2017>) — большое описание текущего состояния данных и машинного обучения.
- 5 Gender Recognition by Voice (<https://www.kaggle.com/datasets/primaryobjects/voicegender>). База данных для идентификации пола по голосу на основе акустических свойств речи. Набор данных состоит из 3168 записей мужчин и женщин.
- 6 Fashion MNIST (<https://www.kaggle.com/datasets/zalando>). Набор состоит из 70000 черно-белых изображений одежды по 10 классам: футболки, брюки, свитеры, платья, кроссовки и т.д. Каждая картинка имеет размер 28×28 пикселей.

- ❶ SOCR Data Dinov (http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_Dinov_020108_HeightsWeights) — dataset содержит 25000 записей о росте и весе молодых людей.
- ❷ Young People Survey (<https://www.kaggle.com/datasets/miroslavsabo/young-people-survey>). Набор данных о предпочтениях, интересах, привычках, мнениях молодежи.
- ❸ World University Rankings (<https://www.kaggle.com/datasets/mylesoneill/world-university-rankings>). Данные о лучших университетах мира по нескольким рейтингам:
 - Times Higher Education World University Ranking (<https://www.timeshighereducation.com/world-university-rankings>),
 - Shanghai Ranking (<http://www.shanghairanking.com/>) — в основном технический,
 - Center for World University Rankings (<https://cwur.org/>) — арабский.
- ❹ Kaggle Machine Learning and Data Science Survey 2017 (<https://www.kaggle.com/datasets/kaggle/kaggle-survey-2017>) — большое описание текущего состояния данных и машинного обучения.
- ❺ Gender Recognition by Voice (<https://www.kaggle.com/datasets/primaryobjects/voicegender>). База данных для идентификации пола по голосу на основе акустических свойств речи. Набор данных состоит из 3168 записей мужчин и женщин.
- ❻ Fashion MNIST (<https://www.kaggle.com/datasets/zalando>). Набор состоит из 70000 черно-белых изображений одежды по 10 классам: футболки, брюки, свитеры, платья, кроссовки и т.д. Каждая картинка имеет размер 28×28 пикселей.
- ❼ Mall Customers Dataset (<https://www.kaggle.com/datasets/shwetabh123/mall-customers>) — данные посетителей магазина: id, пол, возраст, доход, рейтинг трат.

- ❶ SOCR Data Dinov (http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_Dinov_020108_HeightsWeights) — dataset содержит 25000 записей о росте и весе молодых людей.
- ❷ Young People Survey (<https://www.kaggle.com/datasets/miroslavsabo/young-people-survey>). Набор данных о предпочтениях, интересах, привычках, мнениях молодежи.
- ❸ World University Rankings (<https://www.kaggle.com/datasets/mylesoneill/world-university-rankings>). Данные о лучших университетах мира по нескольким рейтингам:
 - Times Higher Education World University Ranking (<https://www.timeshighereducation.com/world-university-rankings>),
 - Shanghai Ranking (<http://www.shanghairanking.com/>) — в основном технический,
 - Center for World University Rankings (<https://cwur.org/>) — арабский.
- ❹ Kaggle Machine Learning and Data Science Survey 2017 (<https://www.kaggle.com/datasets/kaggle/kaggle-survey-2017>) — большое описание текущего состояния данных и машинного обучения.
- ❺ Gender Recognition by Voice (<https://www.kaggle.com/datasets/primaryobjects/voicegender>). База данных для идентификации пола по голосу на основе акустических свойств речи. Набор данных состоит из 3168 записей мужчин и женщин.
- ❻ Fashion MNIST (<https://www.kaggle.com/datasets/zalando>). Набор состоит из 70000 черно-белых изображений одежды по 10 классам: футболки, брюки, свитеры, платья, кроссовки и т.д. Каждая картинка имеет размер 28×28 пикселей.
- ❼ Mall Customers Dataset (<https://www.kaggle.com/datasets/shwetabh123/mall-customers>) — данные посетителей магазина: id, пол, возраст, доход, рейтинг трат.
- ❽ Satellite Photograph Order (<https://www.kaggle.com/c/draper-satellite-image-chronology>) — набор данных спутниковых фотографий Земли. Задача: предсказать, какие фотографии были сделаны раньше других.

- ❶ SOCR Data Dinov (http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_Dinov_020108_HeightsWeights) — dataset содержит 25000 записей о росте и весе молодых людей.
- ❷ Young People Survey (<https://www.kaggle.com/datasets/miroslavsabo/young-people-survey>). Набор данных о предпочтениях, интересах, привычках, мнениях молодежи.
- ❸ World University Rankings (<https://www.kaggle.com/datasets/mylesoneill/world-university-rankings>). Данные о лучших университетах мира по нескольким рейтингам:
 - Times Higher Education World University Ranking (<https://www.timeshighereducation.com/world-university-rankings>),
 - Shanghai Ranking (<http://www.shanghairanking.com/>) — в основном технический,
 - Center for World University Rankings (<https://cwur.org/>) — арабский.
- ❹ Kaggle Machine Learning and Data Science Survey 2017 (<https://www.kaggle.com/datasets/kaggle/kaggle-survey-2017>) — большое описание текущего состояния данных и машинного обучения.
- ❺ Gender Recognition by Voice (<https://www.kaggle.com/datasets/primaryobjects/voicegender>). База данных для идентификации пола по голосу на основе акустических свойств речи. Набор данных состоит из 3168 записей мужчин и женщин.
- ❻ Fashion MNIST (<https://www.kaggle.com/datasets/zalando>). Набор состоит из 70000 черно-белых изображений одежды по 10 классам: футболки, брюки, свитеры, платья, кроссовки и т.д. Каждая картинка имеет размер 28×28 пикселей.
- ❼ Mall Customers Dataset (<https://www.kaggle.com/datasets/shwetabh123/mall-customers>) — данные посетителей магазина: id, пол, возраст, доход, рейтинг трат.
- ❽ Satellite Photograph Order (<https://www.kaggle.com/c/draper-satellite-image-chronology>) — набор данных спутниковых фотографий Земли. Задача: предсказать, какие фотографии были сделаны раньше других.
- ❾ Изображения клеток крови (<https://www.kaggle.com/datasets/paultimothymooney/blood-cells>) — 12500 изображений: четыре разных типа клеток.

- 1 SOCR Data Dinov (http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_Dinov_020108_HeightsWeights) — dataset содержит 25000 записей о росте и весе молодых людей.
- 2 Young People Survey (<https://www.kaggle.com/datasets/miroslavsabo/young-people-survey>). Набор данных о предпочтениях, интересах, привычках, мнениях молодежи.
- 3 World University Rankings (<https://www.kaggle.com/datasets/mylesoneill/world-university-rankings>). Данные о лучших университетах мира по нескольким рейтингам:
 - Times Higher Education World University Ranking (<https://www.timeshighereducation.com/world-university-rankings>),
 - Shanghai Ranking (<http://www.shanghairanking.com/>) — в основном технический,
 - Center for World University Rankings (<https://cwur.org/>) — арабский.
- 4 Kaggle Machine Learning and Data Science Survey 2017 (<https://www.kaggle.com/datasets/kaggle/kaggle-survey-2017>) — большое описание текущего состояния данных и машинного обучения.
- 5 Gender Recognition by Voice (<https://www.kaggle.com/datasets/primaryobjects/voicegender>). База данных для идентификации пола по голосу на основе акустических свойств речи. Набор данных состоит из 3168 записей мужчин и женщин.
- 6 Fashion MNIST (<https://www.kaggle.com/datasets/zalando>). Набор состоит из 70000 черно-белых изображений одежды по 10 классам: футболки, брюки, свитеры, платья, кроссовки и т.д. Каждая картинка имеет размер 28×28 пикселей.
- 7 Mall Customers Dataset (<https://www.kaggle.com/datasets/shwetabh123/mall-customers>) — данные посетителей магазина: id, пол, возраст, доход, рейтинг трат.
- 8 Satellite Photograph Order (<https://www.kaggle.com/c/draper-satellite-image-chronology>) — набор данных спутниковых фотографий Земли. Задача: предсказать, какие фотографии были сделаны раньше других.
- 9 Изображения клеток крови (<https://www.kaggle.com/datasets/paultimothymooney/blood-cells>) — 12500 изображений: четыре разных типа клеток.
- 10 Manufacturing Process Failures (<https://www.kaggle.com/c/bosch-production-line-performance>) — набор переменных, которые были измерены в ходе производственного процесса. Цель состоит в том, чтобы предсказать сбой в производстве.

- 11 Multiple Choice Questions (<https://www.kaggle.com/c/the-allen-ai-science-challenge>) — набор данных из вопросов с множественным выбором и соответствующих правильных ответов. Задача: предсказать ответ на любой заданный вопрос.

- 11 Multiple Choice Questions (<https://www.kaggle.com/c/the-allen-ai-science-challenge>) — набор данных из вопросов с множественным выбором и соответствующих правильных ответов. Задача: предсказать ответ на любой заданный вопрос.
- 12 Pokemon Dataset (<https://www.kaggle.com/datasets/abcsds/pokemon>) — статистические данные по 721 покемону: тип, HP, атака, особая атака, особая защита и скорость.

- 11 Multiple Choice Questions (<https://www.kaggle.com/c/the-allen-ai-science-challenge>) — набор данных из вопросов с множественным выбором и соответствующих правильных ответов. Задача: предсказать ответ на любой заданный вопрос.
- 12 Pokemon Dataset (<https://www.kaggle.com/datasets/abcsds/pokemon>) — статистические данные по 721 покемону: тип, HP, атака, особая атака, особая защита и скорость.
- 13 Stack Overflow Questions (<https://www.kaggle.com/datasets/imoore/60k-stack-overflow-questions-with-quality-rate>) — dataset содержит 60000 вопросов на Stack Overflow с 2016 по 2020 годы. Есть 3 типа вопросов: HQ (высококачественные сообщения), LQ-EDIT (низкокачественные сообщения с отрицательной оценкой) и LQ-CLOSE (низкокачественные сообщения, закрытые сообществом).

- 11 Multiple Choice Questions (<https://www.kaggle.com/c/the-allen-ai-science-challenge>) — набор данных из вопросов с множественным выбором и соответствующих правильных ответов. Задача: предсказать ответ на любой заданный вопрос.
- 12 Pokemon Dataset (<https://www.kaggle.com/datasets/abcsds/pokemon>) — статистические данные по 721 покемону: тип, HP, атака, особая атака, особая защита и скорость.
- 13 Stack Overflow Questions (<https://www.kaggle.com/datasets/imoore/60k-stack-overflow-questions-with-quality-rate>) — dataset содержит 60000 вопросов на Stack Overflow с 2016 по 2020 годы. Есть 3 типа вопросов: HQ (высококачественные сообщения), LQ-EDIT (низкокачественные сообщения с отрицательной оценкой) и LQ-CLOSE (низкокачественные сообщения, закрытые сообществом).
- 14 Netflix movies and TV shows (<https://www.kaggle.com/datasets/shivamb/netflix-shows>) — фильмы и сериалы, доступные на Netflix на середину 2021 года: название, режиссер, рейтинг, год выпуска и продолжительность.

- 11 Multiple Choice Questions (<https://www.kaggle.com/c/the-allen-ai-science-challenge>) — набор данных из вопросов с множественным выбором и соответствующих правильных ответов. Задача: предсказать ответ на любой заданный вопрос.
- 12 Pokemon Dataset (<https://www.kaggle.com/datasets/abcsds/pokemon>) — статистические данные по 721 покемону: тип, HP, атака, особая атака, особая защита и скорость.
- 13 Stack Overflow Questions (<https://www.kaggle.com/datasets/imoore/60k-stack-overflow-questions-with-quality-rate>) — dataset содержит 60000 вопросов на Stack Overflow с 2016 по 2020 годы. Есть 3 типа вопросов: HQ (высококачественные сообщения), LQ-EDIT (низкокачественные сообщения с отрицательной оценкой) и LQ-CLOSE (низкокачественные сообщения, закрытые сообществом).
- 14 Netflix movies and TV shows (<https://www.kaggle.com/datasets/shivamb/netflix-shows>) — фильмы и сериалы, доступные на Netflix на середину 2021 года: название, режиссер, рейтинг, год выпуска и продолжительность.
- 15 Fake News Detection Dataset (<https://www.kaggle.com/c/fake-news/data>) — содержит 7796 записей с разметкой новостей: правда или ложь.

- 11 Multiple Choice Questions (<https://www.kaggle.com/c/the-allen-ai-science-challenge>) — набор данных из вопросов с множественным выбором и соответствующих правильных ответов. Задача: предсказать ответ на любой заданный вопрос.
- 12 Pokemon Dataset (<https://www.kaggle.com/datasets/abcsds/pokemon>) — статистические данные по 721 покемону: тип, HP, атака, особая атака, особая защита и скорость.
- 13 Stack Overflow Questions (<https://www.kaggle.com/datasets/imoore/60k-stack-overflow-questions-with-quality-rate>) — dataset содержит 60000 вопросов на Stack Overflow с 2016 по 2020 годы. Есть 3 типа вопросов: HQ (высококачественные сообщения), LQ-EDIT (низкокачественные сообщения с отрицательной оценкой) и LQ-CLOSE (низкокачественные сообщения, закрытые сообществом).
- 14 Netflix movies and TV shows (<https://www.kaggle.com/datasets/shivamb/netflix-shows>) — фильмы и сериалы, доступные на Netflix на середину 2021 года: название, режиссер, рейтинг, год выпуска и продолжительность.
- 15 Fake News Detection Dataset (<https://www.kaggle.com/c/fake-news/data>) — содержит 7796 записей с разметкой новостей: правда или ложь.
- 16 Uber Pickups Dataset (<https://www.kaggle.com/datasets/fivethirtyeight/uber-pickups-in-new-york-city>) — данные о более чем 4 миллионах поездок на Uber 2014 года и 14 миллионах 2015 года.

- 11 Multiple Choice Questions (<https://www.kaggle.com/c/the-allen-ai-science-challenge>) — набор данных из вопросов с множественным выбором и соответствующих правильных ответов. Задача: предсказать ответ на любой заданный вопрос.
- 12 Pokemon Dataset (<https://www.kaggle.com/datasets/abcsds/pokemon>) — статистические данные по 721 покемону: тип, HP, атака, особая атака, особая защита и скорость.
- 13 Stack Overflow Questions (<https://www.kaggle.com/datasets/imoore/60k-stack-overflow-questions-with-quality-rate>) — dataset содержит 60000 вопросов на Stack Overflow с 2016 по 2020 годы. Есть 3 типа вопросов: HQ (высококачественные сообщения), LQ-EDIT (низкокачественные сообщения с отрицательной оценкой) и LQ-CLOSE (низкокачественные сообщения, закрытые сообществом).
- 14 Netflix movies and TV shows (<https://www.kaggle.com/datasets/shivamb/netflix-shows>) — фильмы и сериалы, доступные на Netflix на середину 2021 года: название, режиссер, рейтинг, год выпуска и продолжительность.
- 15 Fake News Detection Dataset (<https://www.kaggle.com/c/fake-news/data>) — содержит 7796 записей с разметкой новостей: правда или ложь.
- 16 Uber Pickups Dataset (<https://www.kaggle.com/datasets/fivethirtyeight/uber-pickups-in-new-york-city>) — данные о более чем 4 миллионах поездок на Uber 2014 года и 14 миллионах 2015 года.
- 17 Flickr 30k Dataset (<https://www.kaggle.com/datasets/hsankesara/flickr-image-dataset>) — более 30000 изображений и подписей к ним.

- 11 Multiple Choice Questions (<https://www.kaggle.com/c/the-allen-ai-science-challenge>) — набор данных из вопросов с множественным выбором и соответствующих правильных ответов. Задача: предсказать ответ на любой заданный вопрос.
- 12 Pokemon Dataset (<https://www.kaggle.com/datasets/abcsds/pokemon>) — статистические данные по 721 покемону: тип, HP, атака, особая атака, особая защита и скорость.
- 13 Stack Overflow Questions (<https://www.kaggle.com/datasets/imoore/60k-stack-overflow-questions-with-quality-rate>) — dataset содержит 60000 вопросов на Stack Overflow с 2016 по 2020 годы. Есть 3 типа вопросов: HQ (высококачественные сообщения), LQ-EDIT (низкокачественные сообщения с отрицательной оценкой) и LQ-CLOSE (низкокачественные сообщения, закрытые сообществом).
- 14 Netflix movies and TV shows (<https://www.kaggle.com/datasets/shivamb/netflix-shows>) — фильмы и сериалы, доступные на Netflix на середину 2021 года: название, режиссер, рейтинг, год выпуска и продолжительность.
- 15 Fake News Detection Dataset (<https://www.kaggle.com/c/fake-news/data>) — содержит 7796 записей с разметкой новостей: правда или ложь.
- 16 Uber Pickups Dataset (<https://www.kaggle.com/datasets/fivethirtyeight/uber-pickups-in-new-york-city>) — данные о более чем 4 миллионах поездок на Uber 2014 года и 14 миллионах 2015 года.
- 17 Flickr 30k Dataset (<https://www.kaggle.com/datasets/hsankesara/flickr-image-dataset>) — более 30000 изображений и подписей к ним.
- 18 Шоколадный рейтинг (<https://www.kaggle.com/datasets/ratatman/chocolate-bar-ratings>) — экспертный рейтинг более 1700 шоколадных батончиков.

- 11 Multiple Choice Questions (<https://www.kaggle.com/c/the-allen-ai-science-challenge>) — набор данных из вопросов с множественным выбором и соответствующих правильных ответов. Задача: предсказать ответ на любой заданный вопрос.
- 12 Pokemon Dataset (<https://www.kaggle.com/datasets/abcsds/pokemon>) — статистические данные по 721 покемону: тип, HP, атака, особая атака, особая защита и скорость.
- 13 Stack Overflow Questions (<https://www.kaggle.com/datasets/imoore/60k-stack-overflow-questions-with-quality-rate>) — dataset содержит 60000 вопросов на Stack Overflow с 2016 по 2020 годы. Есть 3 типа вопросов: HQ (высококачественные сообщения), LQ-EDIT (низкокачественные сообщения с отрицательной оценкой) и LQ-CLOSE (низкокачественные сообщения, закрытые сообществом).
- 14 Netflix movies and TV shows (<https://www.kaggle.com/datasets/shivamb/netflix-shows>) — фильмы и сериалы, доступные на Netflix на середину 2021 года: название, режиссер, рейтинг, год выпуска и продолжительность.
- 15 Fake News Detection Dataset (<https://www.kaggle.com/c/fake-news/data>) — содержит 7796 записей с разметкой новостей: правда или ложь.
- 16 Uber Pickups Dataset (<https://www.kaggle.com/datasets/fivethirtyeight/uber-pickups-in-new-york-city>) — данные о более чем 4 миллионах поездок на Uber 2014 года и 14 миллионах 2015 года.
- 17 Flickr 30k Dataset (<https://www.kaggle.com/datasets/hsankesara/flickr-image-dataset>) — более 30000 изображений и подписей к ним.
- 18 Шоколадный рейтинг (<https://www.kaggle.com/datasets/ratatman/chocolate-bar-ratings>) — экспертный рейтинг более 1700 шоколадных батончиков.
- 19 Звуки сердцебиения (<https://www.kaggle.com/datasets/kinguistics/heartbeat-sounds>) — для классификации аномалий сердцебиения по стетоскопу.

- 11 Multiple Choice Questions (<https://www.kaggle.com/c/the-allen-ai-science-challenge>) — набор данных из вопросов с множественным выбором и соответствующих правильных ответов. Задача: предсказать ответ на любой заданный вопрос.
- 12 Pokemon Dataset (<https://www.kaggle.com/datasets/abcsds/pokemon>) — статистические данные по 721 покемону: тип, HP, атака, особая атака, особая защита и скорость.
- 13 Stack Overflow Questions (<https://www.kaggle.com/datasets/imoore/60k-stack-overflow-questions-with-quality-rate>) — dataset содержит 60000 вопросов на Stack Overflow с 2016 по 2020 годы. Есть 3 типа вопросов: HQ (высококачественные сообщения), LQ-EDIT (низкокачественные сообщения с отрицательной оценкой) и LQ-CLOSE (низкокачественные сообщения, закрытые сообществом).
- 14 Netflix movies and TV shows (<https://www.kaggle.com/datasets/shivamb/netflix-shows>) — фильмы и сериалы, доступные на Netflix на середину 2021 года: название, режиссер, рейтинг, год выпуска и продолжительность.
- 15 Fake News Detection Dataset (<https://www.kaggle.com/c/fake-news/data>) — содержит 7796 записей с разметкой новостей: правда или ложь.
- 16 Uber Pickups Dataset (<https://www.kaggle.com/datasets/fivethirtyeight/uber-pickups-in-new-york-city>) — данные о более чем 4 миллионах поездок на Uber 2014 года и 14 миллионах 2015 года.
- 17 Flickr 30k Dataset (<https://www.kaggle.com/datasets/hsankesara/flickr-image-dataset>) — более 30000 изображений и подписей к ним.
- 18 Шоколадный рейтинг (<https://www.kaggle.com/datasets/ratatman/chocolate-bar-ratings>) — экспертный рейтинг более 1700 шоколадных батончиков.
- 19 Звуки сердцебиения (<https://www.kaggle.com/datasets/kinguistics/heartbeat-sounds>) — для классификации аномалий сердцебиения по стетоскопу.
- 20 Качество красного вина (<https://www.kaggle.com/datasets/uciml/red-wine-quality-cortez-et-al-2009>) — набор данных для регрессии или классификации.

❶ *NumPy* (Numerical Python extensions) — библиотека для математических вычислений.

Англоязычная информация — на официальном сайте (<https://numpy.org/doc/stable/reference/>).

Подробной документации NumPy на русском языке нет, есть многочисленные выжимки (например, <https://highload.today/numpy-python/> или <https://habr.com/ru/post/352678/>).

❶ *NumPy* (Numerical Python extensions) — библиотека для математических вычислений.

Англоязычная информация — на официальном сайте (<https://numpy.org/doc/stable/reference/>).

Подробной документации NumPy на русском языке нет, есть многочисленные выжимки (например, <https://highload.today/numpy-python/> или <https://habr.com/ru/post/352678/>).

❷ *SciPy* — библиотека, основанная на расширении NumPy, но для более глубоких и сложных научных вычислений, анализа данных и построения графиков. Включает модули для статистики, оптимизации, интеграции, линейной алгебры, преобразований Фурье, обработки сигналов и изображений и многое другое. В SciPy можно работать с теми же данными, что и в NumPy.

Англоязычная документация на официальном сайте (<https://docs.scipy.org/doc/>).

❶ *NumPy* (Numerical Python extensions) — библиотека для математических вычислений.

Англоязычная информация — на официальном сайте (<https://numpy.org/doc/stable/reference/>).

Подробной документации NumPy на русском языке нет, есть многочисленные выжимки (например, <https://highload.today/numpy-python/> или <https://habr.com/ru/post/352678/>).

❷ *SciPy* — библиотека, основанная на расширении NumPy, но для более глубоких и сложных научных вычислений, анализа данных и построения графиков. Включает модули для статистики, оптимизации, интеграции, линейной алгебры, преобразований Фурье, обработки сигналов и изображений и многое другое. В SciPy можно работать с теми же данными, что и в NumPy.

Англоязычная документация на официальном сайте (<https://docs.scipy.org/doc/>).

❸ *Pandas* — одна из лучших библиотек для исследования данных. Она основана на NumPy, в частности, предоставляет быстрые и гибкие структуры данных, поддерживает преобразование данных NumPy в собственные структуры и обратно. Включает простое построение гистограмм, удобные встроенные функции описательной статистики.

Англоязычная документация на официальном сайте (<https://pandas.pydata.org/docs/>). Неплохой обзор на русском можно посмотреть здесь: <https://russianblogs.com/article/3063300256/>.

❶ *NumPy* (Numerical Python extensions) — библиотека для математических вычислений.

Англоязычная информация — на официальном сайте (<https://numpy.org/doc/stable/reference/>).

Подробной документации NumPy на русском языке нет, есть многочисленные выжимки (например, <https://highload.today/numpy-python/> или <https://habr.com/ru/post/352678/>).

❷ *SciPy* — библиотека, основанная на расширении NumPy, но для более глубоких и сложных научных вычислений, анализа данных и построения графиков. Включает модули для статистики, оптимизации, интеграции, линейной алгебры, преобразований Фурье, обработки сигналов и изображений и многое другое. В SciPy можно работать с теми же данными, что и в NumPy.

Англоязычная документация на официальном сайте (<https://docs.scipy.org/doc/>).

❸ *Pandas* — одна из лучших библиотек для исследования данных. Она основана на NumPy, в частности, предоставляет быстрые и гибкие структуры данных, поддерживает преобразование данных NumPy в собственные структуры и обратно. Включает простое построение гистограмм, удобные встроенные функции описательной статистики.

Англоязычная документация на официальном сайте (<https://pandas.pydata.org/docs/>). Неплохой обзор на русском можно посмотреть здесь: <https://russianblogs.com/article/3063300256/>.

❹ *Scikit-learn* — библиотека машинного обучения Python, с широким спектром алгоритмов кластеризации, регрессии и классификации. Она отлично взаимодействует с NumPy и SciPy. Часто ее одной хватает для полной реализации модели.

Англоязычная документация на официальном сайте (<https://scikit-learn.org/stable/tutorial/index.html/>). Есть хорошие переводы документации в виде статей на Хабр, например, <https://habr.com/ru/post/264241/>.

❶ **NumPy** (Numerical Python extensions) — библиотека для математических вычислений.

Англоязычная информация — на официальном сайте (<https://numpy.org/doc/stable/reference/>).

Подробной документации NumPy на русском языке нет, есть многочисленные выжимки (например, <https://highload.today/numpy-python/> или <https://habr.com/ru/post/352678/>).

❷ **SciPy** — библиотека, основанная на расширении NumPy, но для более глубоких и сложных научных вычислений, анализа данных и построения графиков. Включает модули для статистики, оптимизации, интеграции, линейной алгебры, преобразований Фурье, обработки сигналов и изображений и многое другое. В SciPy можно работать с теми же данными, что и в NumPy.

Англоязычная документация на официальном сайте (<https://docs.scipy.org/doc/>).

❸ **Pandas** — одна из лучших библиотек для исследования данных. Она основана на NumPy, в частности, предоставляет быстрые и гибкие структуры данных, поддерживает преобразование данных NumPy в собственные структуры и обратно. Включает простое построение гистограмм, удобные встроенные функции описательной статистики.

Англоязычная документация на официальном сайте (<https://pandas.pydata.org/docs/>). Неплохой обзор на русском можно посмотреть здесь: <https://russianblogs.com/article/3063300256/>.

❹ **Scikit-learn** — библиотека машинного обучения Python, с широким спектром алгоритмов кластеризации, регрессии и классификации. Она отлично взаимодействует с NumPy и SciPy. Часто ее одной хватает для полной реализации модели.

Англоязычная документация на официальном сайте (<https://scikit-learn.org/stable/tutorial/index.html/>). Есть хорошие переводы документации в виде статей на Хабр, например, <https://habr.com/ru/post/264241/>.

❺ **Matplotlib** — пожалуй, самая известная библиотека для создания статических, анимированных и интерактивных визуализаций данных. Поддерживает множество стандартных средств для визуализации данных, представленных различными графиками и диаграммами. Также может работать вместе с NumPy и SciPy. Интерфейс Matplotlib похож на MATLAB.

Англоязычная документация на официальном сайте (<https://matplotlib.org/>).

Есть краткое руководство на русском — https://pyprog.pro/mpl/mpl_short_guide.html.

6 *Plotly* — интерактивная библиотека построения графиков с открытым исходным кодом на основе браузера для Python. Plotly можно использовать для создания красивых и интерактивных визуализаций. Это хороший инструмент для выявления закономерностей в наборах данных. Проект на GitHub — <https://github.com/plotly/plotly.py>.

6 *Plotly* — интерактивная библиотека построения графиков с открытым исходным кодом на основе браузера для Python. Plotly можно использовать для создания красивых и интерактивных визуализаций. Это хороший инструмент для выявления закономерностей в наборах данных.

Проект на GitHub — <https://github.com/plotly/plotly.py>.

7 *Seaborn* — библиотека визуализации на Python, основанная на Matplotlib. Предоставляет собой высокоуровневый интерфейс для рисования привлекательных статистических графиков на Python, тесно интегрирована со структурами данных Pandas.

Проект на GitHub — <https://github.com/mwaskom/seaborn>.

Мотивация



Первым шагом машинного обучения является сбор данных.

Мотивация

- 1. Первым шагом машинного обучения является сбор данных.
- 1. Мы собрали данные, что с ними делать?

Мотивация

- 1 Первый шагом машинного обучения является сбор данных.
- 1 Мы собрали данные, что с ними делать?
- 1 Данные нужно проанализировать в зависимости от поставленной задачи.

Мотивация

- 1 Первый шагом машинного обучения является сбор данных.
- 1 Мы собрали данные, что с ними делать?
- 1 Данные нужно проанализировать в зависимости от поставленной задачи.
- 1 Для анализа данных, мы будем использовать статистические методы, чтобы прийти к выводам.

Мотивация

- 1 Первый шагом машинного обучения является сбор данных.
- 1 Мы собрали данные, что с ними делать?
- 1 Данные нужно проанализировать в зависимости от поставленной задачи.
- 1 Для анализа данных, мы будем использовать статистические методы, чтобы прийти к выводам.
- 1 Решение задачи на основе данных будет зависеть от того, насколько эффективно используются эти методы.

Мотивация

- 1 Первый шагом машинного обучения является сбор данных.
- 1 Мы собрали данные, что с ними делать?
- 1 Данные нужно проанализировать в зависимости от поставленной задачи.
- 1 Для анализа данных, мы будем использовать статистические методы, чтобы прийти к выводам.
- 1 Решение задачи на основе данных будет зависеть от того, насколько эффективно используются эти методы.

Описательная статистика

- 1 **Описательная** или дескриптивная статистика занимается обработкой эмпирических данных, их систематизацией, наглядным представлением (визуализацией с помощью графиков, гистограмм, диаграмм рассеяния и т. п.), а также их количественным описанием с помощью основных статистических показателей.

Мотивация






- 1 Первый шагом машинного обучения является сбор данных.
- 1 Мы собрали данные, что с ними делать?
- 1 Данные нужно проанализировать в зависимости от поставленной задачи.
- 1 Для анализа данных, мы будем использовать статистические методы, чтобы прийти к выводам.
- 1 Решение задачи на основе данных будет зависеть от того, насколько эффективно используются эти методы.

Описательная статистика

- 1 **Описательная** или дескриптивная статистика занимается обработкой эмпирических данных, их систематизацией, наглядным представлением (визуализацией с помощью графиков, гистограмм, диаграмм рассеяния и т. п.), а также их количественным описанием с помощью основных статистических показателей.
- 2 Описательная статистика помогает обобщать ключевые характеристики набора данных, способствуя лучшему пониманию данных.

Собрали данные, что дальше?

Мотивация

-  Первым шагом машинного обучения является сбор данных.
-  Мы собрали данные, что с ними делать?
-  Данные нужно проанализировать в зависимости от поставленной задачи.
-  Для анализа данных, мы будем использовать статистические методы, чтобы прийти к выводам.
-  Решение задачи на основе данных будет зависеть от того, насколько эффективно используются эти методы.

Описательная статистика

- 1 Описательная** или дескриптивная статистика занимается обработкой эмпирических данных, их систематизацией, наглядным представлением (визуализацией с помощью графиков, гистограмм, диаграмм рассеяния и т. п.), а также их количественным описанием с помощью основных статистических показателей.
- 2** Описательная статистика помогает обобщать ключевые характеристики набора данных, способствуя лучшему пониманию данных.
- 3** Основная цель описательной статистики — предоставить четкое и краткое изложение данных, позволяющее понять закономерности, тенденции и распределения в наборе данных.

В качестве примера рассмотрим элементы описательной статистики для набора **Ирисы Фишера**

В качестве примера рассмотрим элементы описательной статистики для набора **Ирисы Фишера**

1 Посмотрим среднее (mean), среднеквадратичное отклонение (std), а также 25% — первый, median — второй и 75% — соответственно третий квартили распределения набора данных x_1, \dots, x_n .

$$\text{mean} = \frac{1}{n} \sum_{i=1}^n x_i, \quad \text{std} = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \text{mean})^2},$$

В качестве примера рассмотрим элементы описательной статистики для набора **Ирисы Фишера**

1 Посмотрим среднее (mean), среднеквадратичное отклонение (std), а также 25% — первый, median — второй и 75% — соответственно третий квартили распределения набора данных x_1, \dots, x_n .

$$\text{mean} = \frac{1}{n} \sum_{i=1}^n x_i, \quad \text{std} = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \text{mean})^2},$$

2 Рассчитаем корреляционную матрицу признаков для Ирисов Фишера. Каждая ячейка матрицы показывает корреляцию между двумя признаками.

В качестве примера рассмотрим элементы описательной статистики для набора **Ирисы Фишера**

- 1 Посмотрим среднее (mean), среднеквадратичное отклонение (std), а также 25% — первый, median — второй и 75% — соответственно третий квартили распределения набора данных x_1, \dots, x_n .

$$\text{mean} = \frac{1}{n} \sum_{i=1}^n x_i, \quad \text{std} = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \text{mean})^2},$$

- 2 Рассчитаем корреляционную матрицу признаков для Ирисов Фишера. Каждая ячейка матрицы показывает корреляцию между двумя признаками.
- 3 Построим диаграммы рассеивания для пар признаков. Диаграмма представляет значения двух признаков в виде точек на декартовой плоскости, значения первого по горизонтальной оси, а значения второго — по вертикальной. Диаграммы рассеивания позволяют визуализировать наличие или отсутствие корреляции между двумя признаками.

В качестве примера рассмотрим элементы описательной статистики для набора **Ирисы Фишера**

- 1 Посмотрим среднее (mean), среднеквадратичное отклонение (std), а также 25% — первый, median — второй и 75% — соответственно третий квартили распределения набора данных x_1, \dots, x_n .

$$\text{mean} = \frac{1}{n} \sum_{i=1}^n x_i, \quad \text{std} = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \text{mean})^2},$$

- 2 Рассчитаем корреляционную матрицу признаков для Ирисов Фишера. Каждая ячейка матрицы показывает корреляцию между двумя признаками.
- 3 Построим диаграммы рассеивания для пар признаков. Диаграмма представляет значения двух признаков в виде точек на декартовой плоскости, значения первого по горизонтальной оси, а значения второго — по вертикальной. Диаграммы рассеивания позволяют визуализировать наличие или отсутствие корреляции между двумя признаками.
- 4 Построим гистограммы и графики плотности распределений признаков для Ирисов Фишера.

В качестве примера рассмотрим элементы описательной статистики для набора **Ирисы Фишера**

1 Посмотрим среднее (mean), среднеквадратичное отклонение (std), а также 25% — первый, median — второй и 75% — соответственно третий квартили распределения набора данных x_1, \dots, x_n .

$$\text{mean} = \frac{1}{n} \sum_{i=1}^n x_i, \quad \text{std} = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \text{mean})^2},$$

- 2 Рассчитаем корреляционную матрицу признаков для Ирисов Фишера. Каждая ячейка матрицы показывает корреляцию между двумя признаками.
- 3 Построим диаграммы рассеивания для пар признаков. Диаграмма представляет значения двух признаков в виде точек на декартовой плоскости, значения первого по горизонтальной оси, а значения второго — по вертикальной. Диаграммы рассеивания позволяют визуализировать наличие или отсутствие корреляции между двумя признаками.
- 4 Построим гистограммы и графики плотности распределений признаков для Ирисов Фишера.
- 5 Проверим, имеют ли признаки для Ирисов Фишера нормальное распределение.

1 Статистики для **Ирисов Фишера** — таблица 5.

1 Статистики для Ирисов Фишера — таблица 5.

Таблица 5. Статистики

	min	max	mean	std	25%	median	75%
sepal width	2.0	4.4	3.054	0.434	2.8	3.0	3.3
sepal length	4.3	7.9	5.84	0.828	5.1	5.8	6.4
petal width	0.1	6	1.198	0.763	0.3	1.3	1.8
petal length	1.0	6.9	3.758	1.764	1.6	4.35	5.1

1 Статистики для Ирисов Фишера — таблица 5.

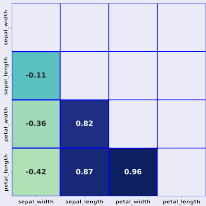
Таблица 5. Статистики

	min	max	mean	std	25%	median	75%
sepal width	2.0	4.4	3.054	0.434	2.8	3.0	3.3
sepal length	4.3	7.9	5.84	0.828	5.1	5.8	6.4
petal width	0.1	6	1.198	0.763	0.3	1.3	1.8
petal length	1.0	6.9	3.758	1.764	1.6	4.35	5.1

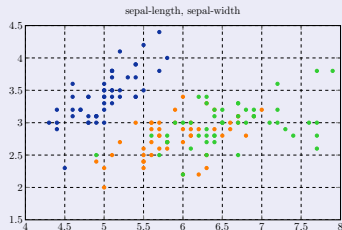
2 Корреляционная матрица признаков (таблица 6) и тепловая карта зависимости признаков для Ирисов Фишера.

Таблица 6. Корреляционная матрица признаков

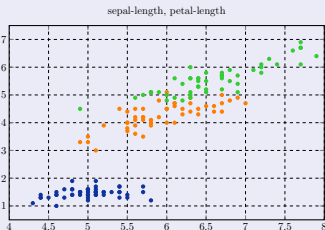
	sepal width	sepal length	petal width	petal length
sepal width	1.0	−0.109	−0.357	−0.421
sepal length	−0.109	1.0	0.818	0.872
petal width	−0.357	0.818	1.0	0.963
petal length	−0.421	0.872	0.963	1.0



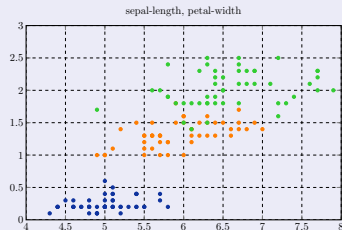
Значения коэффициентов корреляции интерпретируются следующим образом: до 0.2 — очень слабая корреляция, до 0.5 — слабая, до 0.7 — средняя, до 0.9 — высокая, больше 0.9 — очень высокая. Видно, что между признаками *petal length* и *petal width* выявлена очень высокая зависимость — 0.96.



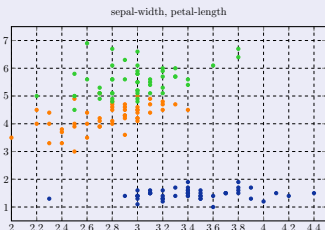
(a)



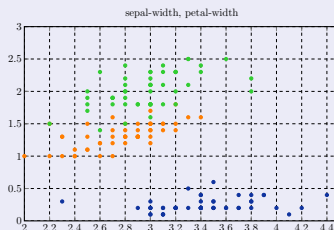
(b)



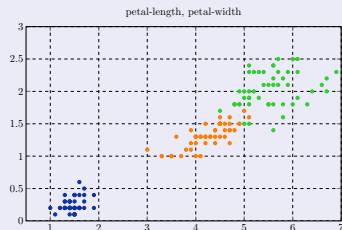
(c)



(d)



(e)



(f)

Рис. 3. Диаграммы рассеивания: *setosa* — синий, *virginica* — зеленый, *versicolor* — оранжевый

Для оценки плотности вероятности применяем **ядерное сглаживание**.

Для оценки плотности вероятности применяем **ядерное сглаживание**.

Пусть (x_1, \dots, x_n) — независимые выборки, взятые из одномерного распределения с неизвестной плотностью f . Ее ядерная плотность $\hat{f}_h(x)$ оценивается следующим образом:

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right),$$

где K — ядро, а $h > 0$ — параметр сглаживания.

Для оценки плотности вероятности применяем **ядерное сглаживание**.

Пусть (x_1, \dots, x_n) — независимые выборки, взятые из одномерного распределения с неизвестной плотностью f . Ее ядерная плотность $\hat{f}_h(x)$ оценивается следующим образом:

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right),$$

где K — ядро, а $h > 0$ — параметр сглаживания.

Строим гистограммы и графики плотности вероятности признаков для Ирисов Фишера — рисунок 4.

Для оценки плотности вероятности применяем **ядерное сглаживание**.

Пусть (x_1, \dots, x_n) — независимые выборки, взятые из одномерного распределения с неизвестной плотностью f . Ее ядерная плотность $\hat{f}_h(x)$ оценивается следующим образом:

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right),$$

где K — ядро, а $h > 0$ — параметр сглаживания.

Строим гистограммы и графики плотности вероятности признаков для Ирисов Фишера — рисунок 4.

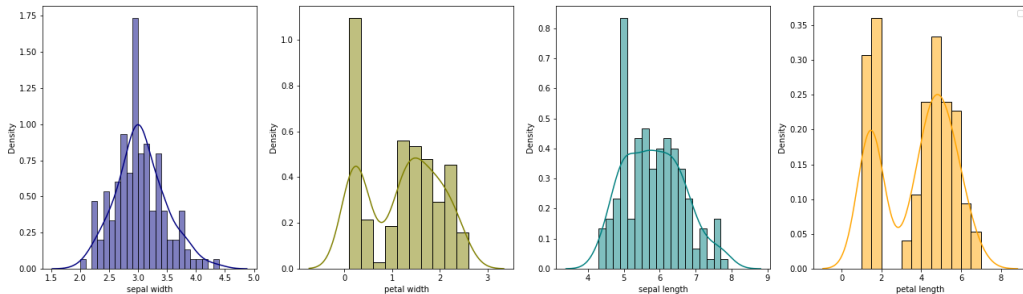


Рис. 4. Гистограммы и графики плотности вероятности признаков



- 1 Проверим, имеют ли признаки для Ирисов Фишера нормальное распределение.

- 1 Проверим, имеют ли признаки для Ирисов Фишера нормальное распределение.
- 2 Для анализа рассмотрим две гипотезы. Гипотеза H_0 — данные распределены нормально, альтернативная гипотеза H_1 — данные не имеют нормального распределения.

- 1 Проверим, имеют ли признаки для Ирисов Фишера нормальное распределение.
- 2 Для анализа рассмотрим две гипотезы. Гипотеза H_0 — данные распределены нормально, альтернативная гипотеза H_1 — данные не имеют нормального распределения.
- 3 Для исследования используем критерии Шапиро–Уилка, Д’Агостино–Пирсона и Жарка–Бера.

- 1 Проверим, имеют ли признаки для Ирисов Фишера нормальное распределение.
- 2 Для анализа рассмотрим две гипотезы. Гипотеза H_0 — данные распределены нормально, альтернативная гипотеза H_1 — данные не имеют нормального распределения.
- 3 Для исследования используем критерии Шапиро–Уилка, Д’Агостино–Пирсона и Жарка–Бера.
- 4 Поскольку объем датасета небольшой, будем применять критерии нормального распределения со стандартным пороговым значением $\alpha = 0.05$. При значении меньше порога гипотеза H_0 отклоняется.

- 1 Проверим, имеют ли признаки для Ирисов Фишера нормальное распределение.
- 2 Для анализа рассмотрим две гипотезы. Гипотеза H_0 — данные распределены нормально, альтернативная гипотеза H_1 — данные не имеют нормального распределения.
- 3 Для исследования используем критерии Шапиро–Уилка, Д’Агостино–Пирсона и Жарка–Бера.
- 4 Поскольку объем датасета небольшой, будем применять критерии нормального распределения со стандартным пороговым значением $\alpha = 0.05$. При значении меньше порога гипотеза H_0 отклоняется.
- 5 Результаты тестов приведены в таблице 7.

Таблица 7. Проверка на нормальное распределение

	Шапиро–Уилка	Д’Агостино–Пирсона	Жарка–Бера
sepal width	0.07518	0.1672	0.2124
sepal length	$1.864 \cdot 10^{-8}$	$1.991 \cdot 10^{-30}$	0.0033
petal width	0.01018	0.05682	0.1061
petal length	$7.545 \cdot 10^{-10}$	$8.677 \cdot 10^{-49}$	0.000905

- 1
- Проверим, имеют ли признаки для Ирисов Фишера нормальное распределение.
- 2
- Для анализа рассмотрим две гипотезы. Гипотеза H_0 — данные распределены нормально, альтернативная гипотеза H_1 — данные не имеют нормального распределения.
- 3
- Для исследования используем критерии Шапиро–Уилка, Д’Агостино–Пирсона и Жарка–Бера.
- 4
- Поскольку объем датасета небольшой, будем применять критерии нормального распределения со стандартным пороговым значением $\alpha = 0.05$. При значении меньше порога гипотеза H_0 отклоняется.
- 5
- Результаты тестов приведены в таблице 7.

Таблица 7. Проверка на нормальное распределение

	Шапиро–Уилка	Д’Агостино–Пирсона	Жарка–Бера
sepal width	0.07518	0.1672	0.2124
sepal length	$1.864 \cdot 10^{-8}$	$1.991 \cdot 10^{-30}$	0.0033
petal width	0.01018	0.05682	0.1061
petal length	$7.545 \cdot 10^{-10}$	$8.677 \cdot 10^{-49}$	0.000905

- 6
- По всем трем тестам нормальное распределение имеет один анализируемый признак: **sepal width**.

- 1 В машинном обучении выбор модели как правило не занимает много времени.

- 1 В машинном обучении выбор модели как правило не занимает много времени.
- 2 При этом предварительная обработка данных может занимать до 90% работы.

- 1 В машинном обучении выбор модели как правило не занимает много времени.
- 2 При этом предварительная обработка данных может занимать до 90% работы.
- 3 Например, высокая размерность данных при работе с текстом или изображениями.

- 1 В машинном обучении выбор модели как правило не занимает много времени.
- 2 При этом предварительная обработка данных может занимать до 90% работы.
- 3 Например, высокая размерность данных при работе с текстом или изображениями.
- 4 Как ускорить процесс обучения?

- 1 В машинном обучении выбор модели как правило не занимает много времени.
- 2 При этом предварительная обработка данных может занимать до 90% работы.
- 3 Например, высокая размерность данных при работе с текстом или изображениями.
- 4 Как ускорить процесс обучения?
- 5 Как выяснить, какой набор параметров хорошо описывает наши данные, но при этом имеет небольшую избыточность? То есть, как уменьшить размерность пространства данных, пожертвовав при этом минимальной частью информации?


- 1 В машинном обучении выбор модели как правило не занимает много времени.
- 2 При этом предварительная обработка данных может занимать до 90% работы.
- 3 Например, высокая размерность данных при работе с текстом или изображениями.
- 4 Как ускорить процесс обучения?
- 5 Как выяснить, какой набор параметров хорошо описывает наши данные, но при этом имеет небольшую избыточность? То есть, как уменьшить размерность пространства данных, пожертвовав при этом минимальной частью информации?
- 6 Как преобразовать данные, чтобы они были удобны для визуализации?

- 1 В машинном обучении выбор модели как правило не занимает много времени.
- 2 При этом предварительная обработка данных может занимать до 90% работы.
- 3 Например, высокая размерность данных при работе с текстом или изображениями.
- 4 Как ускорить процесс обучения?
- 5 Как выяснить, какой набор параметров хорошо описывает наши данные, но при этом имеет небольшую избыточность? То есть, как уменьшить размерность пространства данных, пожертвовав при этом минимальной частью информации?
- 6 Как преобразовать данные, чтобы они были удобны для визуализации?
- 7 Как найти закономерности в данных, чтобы упростить модель?

- 1 В машинном обучении выбор модели как правило не занимает много времени.
- 2 При этом предварительная обработка данных может занимать до 90% работы.
- 3 Например, высокая размерность данных при работе с текстом или изображениями.
- 4 Как ускорить процесс обучения?
- 5 Как выяснить, какой набор параметров хорошо описывает наши данные, но при этом имеет небольшую избыточность? То есть, как уменьшить размерность пространства данных, пожертвовав при этом минимальной частью информации?
- 6 Как преобразовать данные, чтобы они были удобны для визуализации?
- 7 Как найти закономерности в данных, чтобы упростить модель?
- 8 Способы решения этой задачи, называются методами уменьшения размерности (dimensionality reduction).

- 1 В машинном обучении выбор модели как правило не занимает много времени.
- 2 При этом предварительная обработка данных может занимать до 90% работы.
- 3 Например, высокая размерность данных при работе с текстом или изображениями.
- 4 Как ускорить процесс обучения?
- 5 Как выяснить, какой набор параметров хорошо описывает наши данные, но при этом имеет небольшую избыточность? То есть, как уменьшить размерность пространства данных, пожертвовав при этом минимальной частью информации?
- 6 Как преобразовать данные, чтобы они были удобны для визуализации?
- 7 Как найти закономерности в данных, чтобы упростить модель?
- 8 Способы решения этой задачи, называются методами уменьшения размерности (dimensionality reduction).
- 9 Самый популярный и простой — **метод главных компонент** (Principal Components Analysis, PCA).

Понижение размерности данных. Мотивация

- 1 В машинном обучении выбор модели как правило не занимает много времени.
 - 2 При этом предварительная обработка данных может занимать до 90% работы.
 - 3 Например, высокая размерность данных при работе с текстом или изображениями.
 - 4 Как ускорить процесс обучения?
 - 5 Как выяснить, какой набор параметров хорошо описывает наши данные, но при этом имеет небольшую избыточность? То есть, как уменьшить размерность пространства данных, пожертвовав при этом минимальной частью информации?
 - 6 Как преобразовать данные, чтобы они были удобны для визуализации?
 - 7 Как найти закономерности в данных, чтобы упростить модель?
 - 8 Способы решения этой задачи, называются методами уменьшения размерности (dimensionality reduction).
 - 9 Самый популярный и простой — **метод главных компонент** (Principal Components Analysis, PCA).
-  Метод был предложен **Карлом Пирсоном** в 1901 году, затем доработан Гарольдом Хотеллингом. Иногда PCA называют преобразованием **Карунена-Лоэва** (Кари Карунен и Мишель Лоэв), реже преобразованием **Хотеллинга**.

Идея

Основная идея метода главных компонент состоит в том, чтобы уменьшить размерность набора данных, состоящего из множества переменных, сильно или слабо коррелирующих друг с другом.

Идея

Основная идея метода главных компонент состоит в том, чтобы уменьшить размерность набора данных, состоящего из множества переменных, сильно или слабо коррелирующих друг с другом.

Представляем наши данные, как некоторый эллипсоид в пространстве признаков и базис в этом пространстве совпадает с осями этого эллипсоида (рисунок 5, размерность признаков — 2).

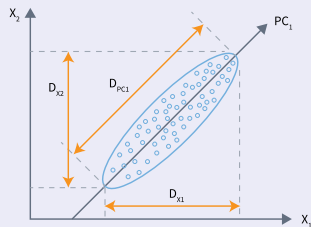


Рис. 5. Иллюстрация идеи PCA

Идея

Основная идея метода главных компонент состоит в том, чтобы уменьшить размерность набора данных, состоящего из множества переменных, сильно или слабо коррелирующих друг с другом.

Представляем наши данные, как некоторый эллипсоид в пространстве признаков и базис в этом пространстве совпадает с осями этого эллипсоида (рисунок 5, размерность признаков — 2).

Наша задача — избавиться от наиболее коррелированных признаков.

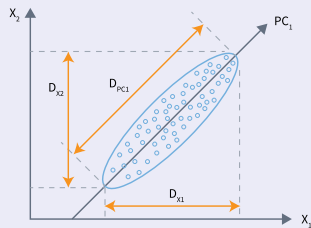


Рис. 5. Иллюстрация идеи PCA

Идея

Основная идея метода главных компонент состоит в том, чтобы уменьшить размерность набора данных, состоящего из множества переменных, сильно или слабо коррелирующих друг с другом.

Представляем наши данные, как некоторый эллипсоид в пространстве признаков и базис в этом пространстве совпадает с осями этого эллипсоида (рисунок 5, размерность признаков — 2).

Наша задача — избавиться от наиболее коррелированных признаков.

Взаимосвязь между признаками покажет ковариационная матрица.

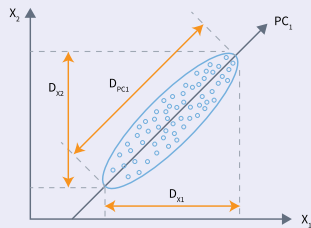


Рис. 5. Иллюстрация идеи PCA

Идея

Основная идея метода главных компонент состоит в том, чтобы уменьшить размерность набора данных, состоящего из множества переменных, сильно или слабо коррелирующих друг с другом.

Представляем наши данные, как некоторый эллипсоид в пространстве признаков и базис в этом пространстве совпадает с осями этого эллипсоида (рисунок 5, размерность признаков — 2).

Наша задача — избавиться от наиболее коррелированных признаков.

Взаимосвязь между признаками покажет ковариационная матрица.

Мы будем отбрасывать «лишнее» подпространство вдоль осей которого эллипсоид будет *наименее растянут*.

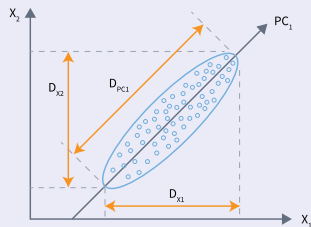


Рис. 5. Иллюстрация идеи PCA

Идея

Основная идея метода главных компонент состоит в том, чтобы уменьшить размерность набора данных, состоящего из множества переменных, сильно или слабо коррелирующих друг с другом.

Представляем наши данные, как некоторый эллипсоид в пространстве признаков и базис в этом пространстве совпадает с осями этого эллипсоида (рисунок 5, размерность признаков — 2).

Наша задача — избавиться от наиболее коррелированных признаков.

Взаимосвязь между признаками покажет ковариационная матрица.

Мы будем отбрасывать «лишнее» подпространство вдоль осей которого эллипсоид будет *наименее растянут*.

Наш алгоритм «жадный» — будем выбирать последовательно оси эллипсоида из оставшихся, вдоль которой дисперсия будет максимальной.

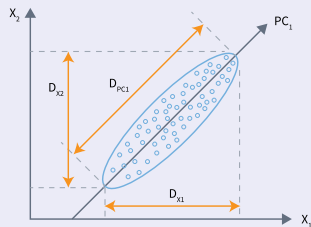


Рис. 5. Иллюстрация идеи PCA

Идея

Основная идея метода главных компонент состоит в том, чтобы уменьшить размерность набора данных, состоящего из множества переменных, сильно или слабо коррелирующих друг с другом.

Представляем наши данные, как некоторый эллипсоид в пространстве признаков и базис в этом пространстве совпадает с осями этого эллипсоида (рисунок 5, размерность признаков — 2).

Наша задача — избавиться от наиболее коррелированных признаков.

Взаимосвязь между признаками покажет ковариационная матрица.

Мы будем отбрасывать «лишнее» подпространство вдоль осей которого эллипсоид будет *наименее растянут*.

Наш алгоритм «жадный» — будем выбирать последовательно оси эллипсоида из оставшихся, вдоль которой дисперсия будет максимальной.

Направления, вдоль которых исходные признаки имеют максимальную дисперсию покажут собственные векторы (компоненты) ковариационной матрицы признаков.

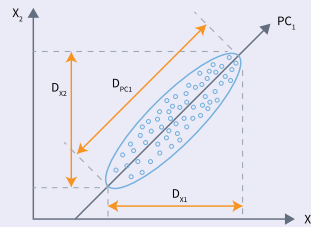


Рис. 5. Иллюстрация идеи PCA

Идея

Основная идея метода главных компонент состоит в том, чтобы уменьшить размерность набора данных, состоящего из множества переменных, сильно или слабо коррелирующих друг с другом.

Представляем наши данные, как некоторый эллипсоид в пространстве признаков и базис в этом пространстве совпадает с осями этого эллипсоида (рисунок 5, размерность признаков — 2).

Наша задача — избавиться от наиболее коррелированных признаков.

Взаимосвязь между признаками покажет ковариационная матрица.

Мы будем отбрасывать «лишнее» подпространство вдоль осей которого эллипсоид будет *наименее растянут*.

Наш алгоритм «жадный» — будем выбирать последовательно оси эллипсоида из оставшихся, вдоль которой дисперсия будет максимальной.

Направления, вдоль которых исходные признаки имеют максимальную дисперсию покажут собственные векторы (компоненты) ковариационной матрицы признаков.

При этом первая ось новой системы координат строится таким образом, чтобы дисперсия данных вдоль нее была бы максимальна (на рисунке 5 — PC_1 с дисперсией D_{PC_1}). Вторая ось строится ортогонально первой так, чтобы дисперсия данных вдоль нее, была бы максимальной из оставшихся возможных и т.д.

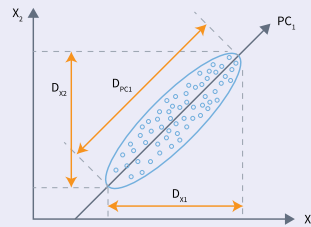


Рис. 5. Иллюстрация идеи PCA

Идея

Основная идея метода главных компонент состоит в том, чтобы уменьшить размерность набора данных, состоящего из множества переменных, сильно или слабо коррелирующих друг с другом.

Представляем наши данные, как некоторый эллипсоид в пространстве признаков и базис в этом пространстве совпадает с осями этого эллипсоида (рисунок 5, размерность признаков — 2).

Наша задача — избавиться от наиболее коррелированных признаков.

Взаимосвязь между признаками покажет ковариационная матрица.

Мы будем отбрасывать «лишнее» подпространство вдоль осей которого эллипсоид будет *наименее растянут*.

Наш алгоритм «жадный» — будем выбирать последовательно оси эллипсоида из оставшихся, вдоль которой дисперсия будет максимальной.

Направления, вдоль которых исходные признаки имеют максимальную дисперсию покажут собственные векторы (компоненты) ковариационной матрицы признаков.

При этом первая ось новой системы координат строится таким образом, чтобы дисперсия данных вдоль нее была бы максимальна (на рисунке 5 — PC_1 с дисперсией D_{PC_1}). Вторая ось строится ортогонально первой так, чтобы дисперсия данных вдоль нее, была бы максимальной из оставшихся возможных и т.д.

Как только мы преобразуем данные по новым координатам, мы сможем отказаться от переменных, которые не имеют дисперсии. Это даст возможность уменьшить размеры данных и сосредоточиться на тех признаках, которые имеют большую дисперсию.

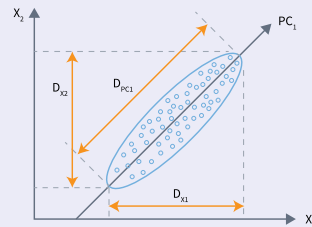


Рис. 5. Иллюстрация идеи PCA

Идея

Основная идея метода главных компонент состоит в том, чтобы уменьшить размерность набора данных, состоящего из множества переменных, сильно или слабо коррелирующих друг с другом.

Представляем наши данные, как некоторый эллипсоид в пространстве признаков и базис в этом пространстве совпадает с осями этого эллипсоида (рисунок 5, размерность признаков — 2).

Наша задача — избавиться от наиболее коррелированных признаков.

Взаимосвязь между признаками покажет ковариационная матрица.

Мы будем отбрасывать «лишнее» подпространство вдоль осей которого эллипсоид будет *наименее растянут*.

Наш алгоритм «жадный» — будем выбирать последовательно оси эллипсоида из оставшихся, вдоль которой дисперсия будет максимальной.

Направления, вдоль которых исходные признаки имеют максимальную дисперсию покажут собственные векторы (компоненты) ковариационной матрицы признаков.

При этом первая ось новой системы координат строится таким образом, чтобы дисперсия данных вдоль нее была бы максимальна (на рисунке 5 — PC_1 с дисперсией D_{PC_1}). Вторая ось строится ортогонально первой так, чтобы дисперсия данных вдоль нее, была бы максимальной из оставшихся возможных и т.д.

Как только мы преобразуем данные по новым координатам, мы сможем отказаться от переменных, которые не имеют дисперсии. Это даст возможность уменьшить размеры данных и сосредоточиться на тех признаках, которые имеют большую дисперсию.

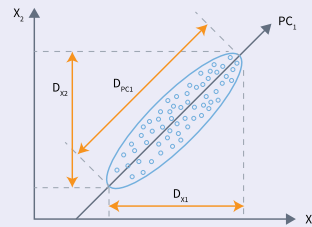


Рис. 5. Иллюстрация идеи PCA

⚠ Жадный алгоритм (англ. *Greedy algorithm*) — это алгоритм, заключающийся в принятии локально оптимальных решений на каждом этапе, допуская, что конечное решение также окажется оптимальным.

Подготовка данных

- 1 Проводим стандартизацию/нормализацию значений признаков.

Подготовка данных

- 1 Проводим стандартизацию/нормализацию значений признаков.

Собственные вектора и собственные числа

- 2 Находим собственные числа $\{\lambda_1, \dots, \lambda_k\}$ и соответствующие им собственные вектора $\{\mathbf{V}_1, \dots, \mathbf{V}_k\}$ ковариационной матрицы Σ .
Ранжируем в порядке убывания собственных чисел:

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_k. \quad (6)$$

Подготовка данных

- 1 Проводим стандартизацию/нормализацию значений признаков.

Собственные вектора и собственные числа

- 2 Находим собственные числа $\{\lambda_1, \dots, \lambda_k\}$ и соответствующие им собственные вектора $\{\mathbf{V}_1, \dots, \mathbf{V}_k\}$ ковариационной матрицы Σ .
Ранжируем в порядке убывания собственных чисел:

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_k. \quad (6)$$

- 3 Ковариационная матрица Σ — симметрична, ее собственные векторы, отвечающие различным собственным значениям, образуют ортогональный базис. Проводим нормирование, получаем ортонормированный базис $\{\tilde{\mathbf{V}}_1, \dots, \tilde{\mathbf{V}}_k\}$.

Подготовка данных

- 1 Проводим стандартизацию/нормализацию значений признаков.

Собственные вектора и собственные числа

- 2 Находим собственные числа $\{\lambda_1, \dots, \lambda_k\}$ и соответствующие им собственные вектора $\{\mathbf{V}_1, \dots, \mathbf{V}_k\}$ ковариационной матрицы Σ . Ранжируем в порядке убывания собственных чисел:

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_k. \quad (6)$$

- 3 Ковариационная матрица Σ — симметрична, ее собственные векторы, отвечающие различным собственным значениям, образуют ортогональный базис. Проводим нормирование, получаем ортонормированный базис $\{\tilde{\mathbf{V}}_1, \dots, \tilde{\mathbf{V}}_k\}$.

Главные компоненты

- 4 Максимальная дисперсия матрицы признаков \mathbf{X} достигается вдоль собственных векторов, соответствующего максимальным собственным значениям. Зафиксируем первые p векторов. Эти вектора называются *главными компонентами* — это оси эллипсоида, на которые мы хотели спроецировать наши данные.

Подготовка данных

- 1 Проводим стандартизацию/нормализацию значений признаков.

Собственные вектора и собственные числа

- 2 Находим собственные числа $\{\lambda_1, \dots, \lambda_k\}$ и соответствующие им собственные вектора $\{\mathbf{V}_1, \dots, \mathbf{V}_k\}$ ковариационной матрицы Σ . Ранжируем в порядке убывания собственных чисел:

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_k. \quad (6)$$

- 3 Ковариационная матрица Σ — симметрична, ее собственные векторы, отвечающие различным собственным значениям, образуют ортогональный базис. Проводим нормирование, получаем ортонормированный базис $\{\tilde{\mathbf{V}}_1, \dots, \tilde{\mathbf{V}}_k\}$.

Главные компоненты

- 4 Максимальная дисперсия матрицы признаков \mathbf{X} достигается вдоль собственных векторов, соответствующего максимальным собственным значениям. Зафиксируем первые p векторов. Эти вектора называются *главными компонентами* — это оси эллипсоида, на которые мы хотели спроецировать наши данные.
- 5 Строим матрицу проекции на новое пространство:

$$\mathbf{V} = (\tilde{\mathbf{V}}_1 \mid \dots \mid \tilde{\mathbf{V}}_p). \quad (7)$$

Подготовка данных

- 1 Проводим стандартизацию/нормализацию значений признаков.

Собственные вектора и собственные числа

- 2 Находим собственные числа $\{\lambda_1, \dots, \lambda_k\}$ и соответствующие им собственные вектора $\{\mathbf{V}_1, \dots, \mathbf{V}_k\}$ ковариационной матрицы Σ . Ранжируем в порядке убывания собственных чисел:

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_k. \quad (6)$$

- 3 Ковариационная матрица Σ — симметрична, ее собственные векторы, отвечающие различным собственным значениям, образуют ортогональный базис. Проводим нормирование, получаем ортонормированный базис $\{\tilde{\mathbf{V}}_1, \dots, \tilde{\mathbf{V}}_k\}$.

Главные компоненты

- 4 Максимальная дисперсия матрицы признаков \mathbf{X} достигается вдоль собственных векторов, соответствующего максимальным собственным значениям. Зафиксируем первые p векторов. Эти вектора называются *главными компонентами* — это оси эллипсоида, на которые мы хотели спроецировать наши данные.
- 5 Строим матрицу проекции на новое пространство:

$$\mathbf{V} = (\tilde{\mathbf{V}}_1 \mid \dots \mid \tilde{\mathbf{V}}_p). \quad (7)$$

Редукция: уменьшение признакового пространства

- 6 Проецируем признаки на выбранные главные компоненты:

$$\tilde{\mathbf{X}} = \mathbf{X} \mathbf{V}. \quad (8)$$

Размерность уменьшилась до заданного значения p .

Дисперсия очень чувствительна к масштабированию. Поэтому предварительно проводим *стандартизацию* — избавляемся большого разброса значений признаков.

Дисперсия очень чувствительна к масштабированию. Поэтому предварительно проводим *стандартизацию* — избавляемся большого разброса значений признаков.

Ковариационная матрица $\text{cov}(\mathbf{X}_i, \mathbf{X}_j)$ — матрица, у которой элемент (i, j) есть корреляция векторов признаков $(\mathbf{X}_i, \mathbf{X}_j)$. Тогда

$$\text{cov}(\mathbf{X}_i, \mathbf{X}_j) = E[(\mathbf{X}_i - \mu_i)(\mathbf{X}_j - \mu_j)] = E[\mathbf{X}_i \mathbf{X}_j] - \mu_i \mu_j, \quad (9)$$

μ_i — математическое ожидание i -ого признака.

Дисперсия очень чувствительна к масштабированию. Поэтому предварительно проводим *стандартизацию* — избавляемся большого разброса значений признаков.

Ковариационная матрица $\text{cov}(\mathbf{X}_i, \mathbf{X}_j)$ — матрица, у которой элемент (i, j) есть корреляция векторов признаков $(\mathbf{X}_i, \mathbf{X}_j)$. Тогда

$$\text{cov}(\mathbf{X}_i, \mathbf{X}_j) = E[(\mathbf{X}_i - \mu_i)(\mathbf{X}_j - \mu_j)] = E[\mathbf{X}_i \mathbf{X}_j] - \mu_i \mu_j, \quad (9)$$

μ_i — математическое ожидание i -ого признака.

Из (9) следует, что ковариационная матрица есть симметричная матрица, на диагонали которой расположены дисперсии соответствующих признаков, а вне диагонали — ковариации соответствующих пар векторов признаков.

Дисперсия очень чувствительна к масштабированию. Поэтому предварительно проводим *стандартизацию* — избавляемся большого разброса значений признаков.

Ковариационная матрица $\text{cov}(\mathbf{X}_i, \mathbf{X}_j)$ — матрица, у которой элемент (i, j) есть корреляция векторов признаков $(\mathbf{X}_i, \mathbf{X}_j)$. Тогда

$$\text{cov}(\mathbf{X}_i, \mathbf{X}_j) = E[(\mathbf{X}_i - \mu_i)(\mathbf{X}_j - \mu_j)] = E[\mathbf{X}_i \mathbf{X}_j] - \mu_i \mu_j, \quad (9)$$

μ_i — математическое ожидание i -ого признака.

Из (9) следует, что ковариационная матрица есть симметричная матрица, на диагонали которой расположены дисперсии соответствующих признаков, а вне диагонали — ковариации соответствующих пар векторов признаков.

Обозначив через \mathbf{X} матрицу признаков, выражение (9) можно переписать в матричном виде:

$$\Sigma = E[(\mathbf{X} - E(\mathbf{X}))(\mathbf{X} - E(\mathbf{X}))^T] \quad (10)$$

Дисперсия очень чувствительна к масштабированию. Поэтому предварительно проводим *стандартизацию* — избавляемся большого разброса значений признаков.

Ковариационная матрица $\text{cov}(\mathbf{X}_i, \mathbf{X}_j)$ — матрица, у которой элемент (i, j) есть корреляция векторов признаков $(\mathbf{X}_i, \mathbf{X}_j)$. Тогда

$$\text{cov}(\mathbf{X}_i, \mathbf{X}_j) = E[(\mathbf{X}_i - \mu_i)(\mathbf{X}_j - \mu_j)] = E[\mathbf{X}_i \mathbf{X}_j] - \mu_i \mu_j, \quad (9)$$

μ_i — математическое ожидание i -ого признака.

Из (9) следует, что ковариационная матрица есть симметричная матрица, на диагонали которой расположены дисперсии соответствующих признаков, а вне диагонали — ковариации соответствующих пар векторов признаков.

Обозначив через \mathbf{X} матрицу признаков, выражение (9) можно переписать в матричном виде:

$$\Sigma = E[(\mathbf{X} - E(\mathbf{X}))(\mathbf{X} - E(\mathbf{X}))^T] \quad (10)$$

Ковариационная матрица легко вычисляется в библиотеке *NumPy*. Пример вычисления:

```
import numpy as np
dataset = np.array([[85,72,80], [64, 35, 26], [67, 48, 29], [100, 11, 102], [130, 14, 151]])
cov = np.cov(dataset)
```

¹Источник: <https://www.turing.com/kb/guide-to-principal-component-analysis>

Входные данные¹. Набор наблюдений — (X_1, \dots, X_5) , признаки — y_1, \dots, y_4 . Исходная матрица признаков \mathbf{X} в таблице 8а.

Таблица 8

	y_1	y_2	y_3	y_4
X_1	1	5	3	1
X_2	4	2	6	3
X_3	1	4	3	2
X_4	4	4	1	1
X_5	5	5	2	3

(а) Матрица исходных признаков \mathbf{X}

¹Источник: <https://www.turing.com/kb/guide-to-principal-component-analysis>

Входные данные¹. Набор наблюдений — (X_1, \dots, X_5) , признаки — y_1, \dots, y_4 . Исходная матрица признаков \mathbf{X} в таблице 8а.

Таблица 8

	y_1	y_2	y_3	y_4
X_1	1	5	3	1
X_2	4	2	6	3
X_3	1	4	3	2
X_4	4	4	1	1
X_5	5	5	2	3

(а) Матрица исходных признаков \mathbf{X}

1 Стандартизация признаков.

¹Источник: <https://www.turing.com/kb/guide-to-principal-component-analysis>

Входные данные¹. Набор наблюдений — (X_1, \dots, X_5) , признаки — y_1, \dots, y_4 . Исходная матрица признаков \mathbf{X} в таблице 8а.

Таблица 8

	y_1	y_2	y_3	y_4
X_1	1	5	3	1
X_2	4	2	6	3
X_3	1	4	3	2
X_4	4	4	1	1
X_5	5	5	2	3

(а) Матрица исходных признаков \mathbf{X}

1 Стандартизация признаков.

✓ Вычисляем средние и стандартные отклонения по каждому признаку (28):

$$\mu_i = (3, 4, 3, 2), \sigma_i = (1.87, 1.223, 1.87, 1.0), i \in 1 : 4.$$

(11)

¹Источник: <https://www.turing.com/kb/guide-to-principal-component-analysis>

Входные данные¹. Набор наблюдений — (X_1, \dots, X_5) , признаки — y_1, \dots, y_4 . Исходная матрица признаков \mathbf{X} в таблице 8а.

Таблица 8

	y_1	y_2	y_3	y_4
X_1	1	5	3	1
X_2	4	2	6	3
X_3	1	4	3	2
X_4	4	4	1	1
X_5	5	5	2	3

(а) Матрица исходных признаков \mathbf{X}

1 Стандартизация признаков.

- ✓ Вычисляем средние и стандартные отклонения по каждому признаку (28):

$$\mu_i = (3, 4, 3, 2), \sigma_i = (1.87, 1.223, 1.87, 1.0), i \in 1 : 4. \quad (11)$$

- ✓ Нормализуем данные (12):

$$\hat{y}_i = \frac{y_i - \mu_i}{\sigma_i}, i \in 1 : 4. \quad (12)$$

¹Источник: <https://www.turing.com/kb/guide-to-principal-component-analysis>

Входные данные¹. Набор наблюдений — (X_1, \dots, X_5) , признаки — y_1, \dots, y_4 . Исходная матрица признаков X в таблице 8a.

Таблица 8

	y_1	y_2	y_3	y_4
X_1	1	5	3	1
X_2	4	2	6	3
X_3	1	4	3	2
X_4	4	4	1	1
X_5	5	5	2	3

(a) Матрица исходных признаков X

	\hat{y}_1	\hat{y}_2	\hat{y}_3	\hat{y}_4
\hat{X}_1	-1.0695	0.8196	0.0	-1.0
\hat{X}_2	0.5347	-1.6393	1.6042	1.0
\hat{X}_3	-1.0695	0.0	0.0	0.0
\hat{X}_4	0.5347	0.0	-1.0695	-1.0
\hat{X}_5	1.0695	0.8196	-0.5347	1.0

(b) Матрица стандартизированных признаков \hat{X}

1 Стандартизация признаков.

✓ Вычисляем средние и стандартные отклонения по каждому признаку (28):

$$\mu_i = (3, 4, 3, 2), \sigma_i = (1.87, 1.223, 1.87, 1.0), i \in 1 : 4. \tag{11}$$

✓ Нормализуем данные (12):

$$\hat{y}_i = \frac{y_i - \mu_i}{\sigma_i}, i \in 1 : 4. \tag{12}$$

✓ Матрица стандартизированных признаков \hat{X} в таблице 8b.

¹Источник: <https://www.turing.com/kb/guide-to-principal-component-analysis>



2 Расчет ковариационной матрицы.

$$\Sigma = \begin{pmatrix} 0.78 & -0.8586 & -0.055 & 0.424 \\ -0.8586 & 0.78 & -0.607 & -0.326 \\ -0.055 & -0.607 & 0.78 & 0.426 \\ 0.424 & -0.326 & 0.426 & 0.78 \end{pmatrix}, \quad (13)$$

где

$$\Sigma_{i,j} = \frac{1}{5} \sum_{k=1}^5 \hat{\mathbf{X}}_{k,i} \hat{\mathbf{X}}_{k,j}, \quad i, j \in 1:4. \quad (14)$$

2 Расчет ковариационной матрицы.

$$\Sigma = \begin{pmatrix} 0.78 & -0.8586 & -0.055 & 0.424 \\ -0.8586 & 0.78 & -0.607 & -0.326 \\ -0.055 & -0.607 & 0.78 & 0.426 \\ 0.424 & -0.326 & 0.426 & 0.78 \end{pmatrix}, \quad (13)$$

где

$$\Sigma_{i,j} = \frac{1}{5} \sum_{k=1}^5 \hat{\mathbf{X}}_{k,i} \hat{\mathbf{X}}_{k,j}, \quad i, j \in 1:4. \quad (14)$$

Для примера, используя таблицу 8 b, получаем:

$$\Sigma_{1,2} = \Sigma_{2,1} = (-1.0695 \times 0.8196 + 0.5347 \times (-1.6393) + (-1.0695) \times 0.0 + 0.5347 \times 0.0 + 1.0695 \times 0.8196)/5 = -0.8586 \quad (15)$$

2 Расчет ковариационной матрицы.

$$\Sigma = \begin{pmatrix} 0.78 & -0.8586 & -0.055 & 0.424 \\ -0.8586 & 0.78 & -0.607 & -0.326 \\ -0.055 & -0.607 & 0.78 & 0.426 \\ 0.424 & -0.326 & 0.426 & 0.78 \end{pmatrix}, \quad (13)$$

где

$$\Sigma_{i,j} = \frac{1}{5} \sum_{k=1}^5 \hat{\mathbf{X}}_{k,i} \hat{\mathbf{X}}_{k,j}, \quad i, j \in 1:4. \quad (14)$$

Для примера, используя таблицу 8 b, получаем:

$$\Sigma_{1,2} = \Sigma_{2,1} = (-1.0695 \times 0.8196 + 0.5347 \times (-1.6393) + (-1.0695) \times 0.0 + 0.5347 \times 0.0 + 1.0695 \times 0.8196)/5 = -0.8586 \quad (15)$$

3 Вычисление собственных чисел и векторов.

2 Расчет ковариационной матрицы.

$$\Sigma = \begin{pmatrix} 0.78 & -0.8586 & -0.055 & 0.424 \\ -0.8586 & 0.78 & -0.607 & -0.326 \\ -0.055 & -0.607 & 0.78 & 0.426 \\ 0.424 & -0.326 & 0.426 & 0.78 \end{pmatrix}, \quad (13)$$

где

$$\Sigma_{i,j} = \frac{1}{5} \sum_{k=1}^5 \hat{\mathbf{X}}_{k,i} \hat{\mathbf{X}}_{k,j}, \quad i, j \in 1:4. \quad (14)$$

Для примера, используя таблицу 8 б, получаем:

$$\Sigma_{1,2} = \Sigma_{2,1} = (-1.0695 \times 0.8196 + 0.5347 \times (-1.6393) + (-1.0695) \times 0.0 + 0.5347 \times 0.0 + 1.0695 \times 0.8196)/5 = -0.8586 \quad (15)$$

3 Вычисление собственных чисел и векторов.

✓ Собственные числа (16):

$$\lambda_1 = 2.1169, \lambda_2 = 0.8554, \lambda_3 = 0.4817, \lambda_4 = 0.3340. \quad (16)$$

2 Расчет ковариационной матрицы.

$$\Sigma = \begin{pmatrix} 0.78 & -0.8586 & -0.055 & 0.424 \\ -0.8586 & 0.78 & -0.607 & -0.326 \\ -0.055 & -0.607 & 0.78 & 0.426 \\ 0.424 & -0.326 & 0.426 & 0.78 \end{pmatrix}, \quad (13)$$

где

$$\Sigma_{i,j} = \frac{1}{5} \sum_{k=1}^5 \hat{\mathbf{X}}_{k,i} \hat{\mathbf{X}}_{k,j}, \quad i, j \in 1:4. \quad (14)$$

Для примера, используя таблицу 8 b, получаем:

$$\Sigma_{1,2} = \Sigma_{2,1} = (-1.0695 \times 0.8196 + 0.5347 \times (-1.6393) + (-1.0695) \times 0.0 + 0.5347 \times 0.0 + 1.0695 \times 0.8196)/5 = -0.8586 \quad (15)$$

3 Вычисление собственных чисел и векторов.

✓ Собственные числа (16):

$$\lambda_1 = 2.1169, \lambda_2 = 0.8554, \lambda_3 = 0.4817, \lambda_4 = 0.3340. \quad (16)$$

✓ Собственные вектора (17):

$$\begin{aligned} \mathbf{V}_1 &= (0.5155, -0.6166, 0.3993, 0.4411), & \mathbf{V}_2 &= (-0.6230, 0.1131, 0.7443, 0.2125), \\ \mathbf{V}_3 &= (0.0349, 0.4523, -0.2809, 0.8457), & \mathbf{V}_4 &= (-0.5872, -0.6343, -0.4558, 0.2122). \end{aligned} \quad (17)$$

2 Расчет ковариационной матрицы.

$$\Sigma = \begin{pmatrix} 0.78 & -0.8586 & -0.055 & 0.424 \\ -0.8586 & 0.78 & -0.607 & -0.326 \\ -0.055 & -0.607 & 0.78 & 0.426 \\ 0.424 & -0.326 & 0.426 & 0.78 \end{pmatrix}, \quad (13)$$

где

$$\Sigma_{i,j} = \frac{1}{5} \sum_{k=1}^5 \hat{\mathbf{X}}_{k,i} \hat{\mathbf{X}}_{k,j}, \quad i, j \in 1:4. \quad (14)$$

Для примера, используя таблицу 8 b, получаем:

$$\Sigma_{1,2} = \Sigma_{2,1} = (-1.0695 \times 0.8196 + 0.5347 \times (-1.6393) + (-1.0695) \times 0.0 + 0.5347 \times 0.0 + 1.0695 \times 0.8196)/5 = -0.8586 \quad (15)$$

3 Вычисление собственных чисел и векторов.

✓ Собственные числа (16):

$$\lambda_1 = 2.1169, \lambda_2 = 0.8554, \lambda_3 = 0.4817, \lambda_4 = 0.3340. \quad (16)$$

✓ Собственные вектора (17):

$$\begin{aligned} \mathbf{V}_1 &= (0.5155, -0.6166, 0.3993, 0.4411), & \mathbf{V}_2 &= (-0.6230, 0.1131, 0.7443, 0.2125), \\ \mathbf{V}_3 &= (0.0349, 0.4523, -0.2809, 0.8457), & \mathbf{V}_4 &= (-0.5872, -0.6343, -0.4558, 0.2122). \end{aligned} \quad (17)$$

4 Собственные числа и вектора легко вычисляются в библиотеке *NumPy*:

```
from numpy.linalg import eig
```



4 **Редукция признаков.** Полагаем $p = 2$, проецируем признаки на первые две главные компоненты: $\tilde{\mathbf{X}} = \hat{\mathbf{X}} \times (\mathbf{V}_1 \mid \mathbf{V}_2)$. Получаем новые признаки для объектов: $\tilde{\mathbf{X}}_1, \dots, \tilde{\mathbf{X}}_5$:

$$\begin{pmatrix} \hat{\mathbf{X}}_1 & -1.0695 & 0.8196 & 0.0000 & -1.0000 \\ \hat{\mathbf{X}}_2 & 0.5347 & -1.6393 & 1.6042 & 1.0000 \\ \hat{\mathbf{X}}_3 & -1.0695 & 0.0000 & 0.0000 & 0.0000 \\ \hat{\mathbf{X}}_4 & 0.5347 & 0.0000 & -1.0695 & 1.0000 \\ \hat{\mathbf{X}}_5 & 1.0695 & 0.8196 & -0.5347 & 1.0000 \end{pmatrix} \times \begin{pmatrix} \mathbf{V}_1 & \mathbf{V}_2 \\ \hline 0.5155 & -0.6230 \\ -0.6166 & 0.1131 \\ 0.3993 & 0.7443 \\ 0.4411 & 0.2125 \end{pmatrix} = \begin{pmatrix} \tilde{\mathbf{X}}_1 & 0.4268 & 0.00116 \\ \tilde{\mathbf{X}}_2 & -0.4235 & -0.3918 \\ \tilde{\mathbf{X}}_3 & -0.5513 & -0.7959 \\ \tilde{\mathbf{X}}_4 & 0.4652 & 0.8999 \\ \tilde{\mathbf{X}}_5 & 0.08273 & 0.2865 \end{pmatrix} \quad (18)$$

4 **Редукция признаков.** Полагаем $p = 2$, проецируем признаки на первые две главные компоненты: $\tilde{\mathbf{X}} = \hat{\mathbf{X}} \times (\mathbf{V}_1 \mid \mathbf{V}_2)$. Получаем новые признаки для объектов: $\tilde{\mathbf{X}}_1, \dots, \tilde{\mathbf{X}}_5$:

$$\begin{pmatrix} \hat{\mathbf{X}}_1 & -1.0695 & 0.8196 & 0.0000 & -1.0000 \\ \hat{\mathbf{X}}_2 & 0.5347 & -1.6393 & 1.6042 & 1.0000 \\ \hat{\mathbf{X}}_3 & -1.0695 & 0.0000 & 0.0000 & 0.0000 \\ \hat{\mathbf{X}}_4 & 0.5347 & 0.0000 & -1.0695 & 1.0000 \\ \hat{\mathbf{X}}_5 & 1.0695 & 0.8196 & -0.5347 & 1.0000 \end{pmatrix} \times \begin{pmatrix} \mathbf{V}_1 & \mathbf{V}_2 \\ 0.5155 & -0.6230 \\ -0.6166 & 0.1131 \\ 0.3993 & 0.7443 \\ 0.4411 & 0.2125 \end{pmatrix} = \begin{pmatrix} \tilde{\mathbf{X}}_1 & 0.4268 & 0.00116 \\ \tilde{\mathbf{X}}_2 & -0.4235 & -0.3918 \\ \tilde{\mathbf{X}}_3 & -0.5513 & -0.7959 \\ \tilde{\mathbf{X}}_4 & 0.4652 & 0.8999 \\ \tilde{\mathbf{X}}_5 & 0.08273 & 0.2865 \end{pmatrix} \quad (18)$$

i На рисунке 6 визуализация нового **двумерного** пространства признаков: $\tilde{\mathbf{X}}_1, \dots, \tilde{\mathbf{X}}_5$ согласно (18). Зеленая линия — первая главная компонента.

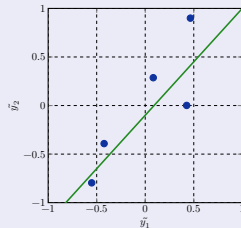


Рис. 6



Как выбрать число главных компонент?

Эффективной размерностью задачи называют число главных компонент p . Как выбрать подходящее значение?

Как выбрать число главных компонент?

Эффективной размерностью задачи называют число главных компонент p . Как выбрать подходящее значение?

- 1 Зададим некоторый порог $\epsilon \in (0, 1)$. Значение p определим, как наименьшее число, при котором величина $\text{Егг}(p)$ не превышает ϵ :

$$\text{Егг}(p) = \frac{\lambda_{p+1} + \dots + \lambda_k}{\lambda_1 + \dots + \lambda_k} \leq \epsilon. \quad (19)$$

Как выбрать число главных компонент?

Эффективной размерностью задачи называют число главных компонент p . Как выбрать подходящее значение?

- 1 Зададим некоторый порог $\epsilon \in (0, 1)$. Значение p определим, как наименьшее число, при котором величина $\text{Егг}(p)$ не превышает ϵ :

$$\text{Егг}(p) = \frac{\lambda_{p+1} + \dots + \lambda_k}{\lambda_1 + \dots + \lambda_k} \leq \epsilon. \quad (19)$$

Значение $\text{Егг}(p)$ в (19) показывает, какая доля информации теряется при замене исходных признаков длины n на редуцированные, длины p .

Как выбрать число главных компонент?

Эффективной размерностью задачи называют число главных компонент p . Как выбрать подходящее значение?

- 1 Зададим некоторый порог $\epsilon \in (0, 1)$. Значение p определим, как наименьшее число, при котором величина $\text{Егг}(p)$ не превышает ϵ :

$$\text{Егг}(p) = \frac{\lambda_{p+1} + \dots + \lambda_k}{\lambda_1 + \dots + \lambda_k} \leq \epsilon. \quad (19)$$

Значение $\text{Егг}(p)$ в (19) показывает, какая доля информации теряется при замене исходных признаков длины n на редуцированные, длины p .

Пусть получен набор собственных чисел

$$\Lambda = \{10212, 7152, 6394, 5803, 5209, 4818, 1282, 1135, 955, 704, 662, 491, 466, 398, 315, 274, 229, 150, 32, 9\} \quad (20)$$

Как выбрать число главных компонент?

Эффективной размерностью задачи называют число главных компонент p . Как выбрать подходящее значение?

- 1 Зададим некоторый порог $\epsilon \in (0, 1)$. Значение p определим, как наименьшее число, при котором величина $\text{Егг}(p)$ не превышает ϵ :

$$\text{Егг}(p) = \frac{\lambda_{p+1} + \dots + \lambda_k}{\lambda_1 + \dots + \lambda_k} \leq \epsilon. \quad (19)$$

Значение $\text{Егг}(p)$ в (19) показывает, какая доля информации теряется при замене исходных признаков длины n на редуцированные, длины p .

Пусть получен набор собственных чисел

$$\Lambda = \{10212, 7152, 6394, 5803, 5209, 4818, 1282, 1135, 955, 704, 662, 491, 466, 398, 315, 274, 229, 150, 32, 9\} \quad (20)$$

Зададим $\epsilon = 0.05$. На рисунке 7 график зависимости $\text{Егг}(p)$ от p для (20). Тогда можно взять $p = 7$.

Как выбрать число главных компонент?

Эффективной размерностью задачи называют число главных компонент p . Как выбрать подходящее значение?

- 1 Зададим некоторый порог $\epsilon \in (0, 1)$. Значение p определим, как наименьшее число, при котором величина $\text{Err}(p)$ не превышает ϵ :

$$\text{Err}(p) = \frac{\lambda_{p+1} + \dots + \lambda_k}{\lambda_1 + \dots + \lambda_k} \leq \epsilon. \quad (19)$$

Значение $\text{Err}(p)$ в (19) показывает, какая доля информации теряется при замене исходных признаков длины n на редуцированные, длины p .

Пусть получен набор собственных чисел

$$\Lambda = \{10212, 7152, 6394, 5803, 5209, 4818, 1282, 1135, 955, 704, 662, 491, 466, 398, 315, 274, 229, 150, 32, 9\} \quad (20)$$

Зададим $\epsilon = 0.05$. На рисунке 7 график зависимости $\text{Err}(p)$ от p для (20). Тогда можно взять $p = 7$.

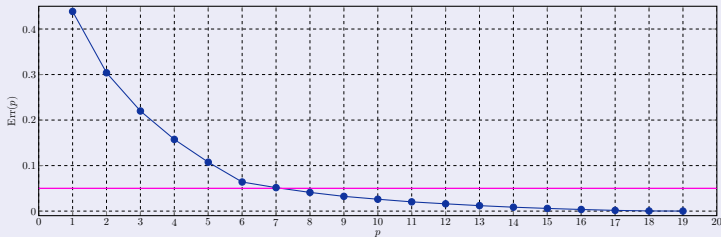


Рис. 7. Выбор эффективной размерности

Как выбрать число главных компонент?

Эффективной размерностью задачи называют число главных компонент p . Как выбрать подходящее значение?

- 1 Зададим некоторый порог $\epsilon \in (0, 1)$. Значение p определим, как наименьшее число, при котором величина $\text{Егг}(p)$ не превышает ϵ :

$$\text{Егг}(p) = \frac{\lambda_{p+1} + \dots + \lambda_k}{\lambda_1 + \dots + \lambda_k} \leq \epsilon. \quad (19)$$

Значение $\text{Егг}(p)$ в (19) показывает, какая доля информации теряется при замене исходных признаков длины n на редуцированные, длины p .

Пусть получен набор собственных чисел

$$\Lambda = \{10212, 7152, 6394, 5803, 5209, 4818, 1282, 1135, 955, 704, 662, 491, 466, 398, 315, 274, 229, 150, 32, 9\} \quad (20)$$

Зададим $\epsilon = 0.05$. На рисунке 7 график зависимости $\text{Егг}(p)$ от p для (20). Тогда можно взять $p = 7$.

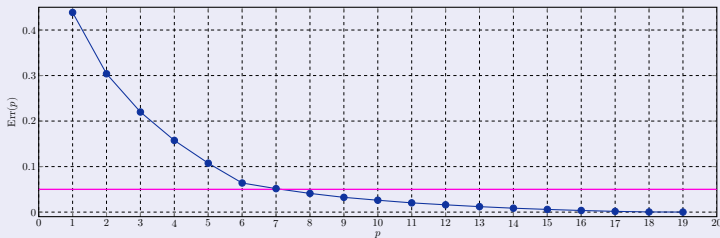


Рис. 7. Выбор эффективной размерности

- 2 Альтернативный к (19) подход (**критерий крутого склона**): выбираем такое p , при котором $\text{Егг}(p-1) \gg \text{Егг}(p)$, и $\text{Егг}(p)$ близко к нулю.

Популярный эвристический подход к оценке числа необходимых главных компонент p .

Популярный эвристический подход к оценке числа необходимых главных компонент p .

- 1 Рассматриваем нормированный набор собственных чисел из (6):

$$\frac{\lambda_1}{\lambda}, \dots, \frac{\lambda_k}{\lambda}, \quad \lambda = \sum_{i=1}^k \lambda_i. \quad (21)$$

Популярный эвристический подход к оценке числа необходимых главных компонент p .

- 1 Рассматриваем нормированный набор собственных чисел из (6):

$$\frac{\lambda_1}{\lambda}, \dots, \frac{\lambda_k}{\lambda}, \quad \lambda = \sum_{i=1}^k \lambda_i. \quad (21)$$

- 2 Находим распределение длин обломков трости единичной длины, сломанной в $k - 1$ точке. Места разлома выбираются независимо и распределены равномерно по длине трости.

Обозначим $L_i, i \in 1 : k$ — длины полученных кусков трости, ранжированных в порядке уменьшения длины:

$$L_1 \geq L_2 \geq \dots \geq L_k. \quad (22)$$

Популярный эвристический подход к оценке числа необходимых главных компонент p .

- 1 Рассматриваем нормированный набор собственных чисел из (6):

$$\frac{\lambda_1}{\lambda}, \dots, \frac{\lambda_k}{\lambda}, \quad \lambda = \sum_{i=1}^k \lambda_i. \quad (21)$$

- 2 Находим распределение длин обломков трости единичной длины, сломанной в $k - 1$ точке. Места разлома выбираются независимо и распределены равномерно по длине трости.

Обозначим $L_i, i \in 1 : k$ — длины полученных кусков трости, ранжированных в порядке уменьшения длины:

$$L_1 \geq L_2 \geq \dots \geq L_k. \quad (22)$$

- 3 Вычисляем математическое ожидание длин L_i :

$$l_i = E(L_i) = \frac{1}{k} \sum_{j=i}^k \frac{1}{j}. \quad (23)$$

Популярный эвристический подход к оценке числа необходимых главных компонент p .

- 1 Рассматриваем нормированный набор собственных чисел из (6):

$$\frac{\lambda_1}{\lambda}, \dots, \frac{\lambda_k}{\lambda}, \quad \lambda = \sum_{i=1}^k \lambda_i. \quad (21)$$

- 2 Находим распределение длин обломков трости единичной длины, сломанной в $k - 1$ точке. Места разлома выбираются независимо и распределены равномерно по длине трости.

Обозначим $L_i, i \in 1 : k$ — длины полученных кусков трости, ранжированных в порядке уменьшения длины:

$$L_1 \geq L_2 \geq \dots \geq L_k. \quad (22)$$

- 3 Вычисляем математическое ожидание длин L_i :

$$l_i = E(L_i) = \frac{1}{k} \sum_{j=i}^k \frac{1}{j}. \quad (23)$$

Например, для $k = 3$:

$$l_1 = \frac{1}{3} \left(1 + \frac{1}{2} + \frac{1}{3} \right) \approx 0.611, \quad l_2 = \frac{1}{3} \left(\frac{1}{2} + \frac{1}{3} \right) \approx 0.278, \quad l_3 = \frac{1}{3} \left(\frac{1}{3} \right) \approx 0.111.$$

Популярный эвристический подход к оценке числа необходимых главных компонент p .

- 1 Рассматриваем нормированный набор собственных чисел из (6):

$$\frac{\lambda_1}{\lambda}, \dots, \frac{\lambda_k}{\lambda}, \quad \lambda = \sum_{i=1}^k \lambda_i. \quad (21)$$

- 2 Находим распределение длин обломков трости единичной длины, сломанной в $k - 1$ точке. Места разлома выбираются независимо и распределены равномерно по длине трости.

Обозначим $L_i, i \in 1 : k$ — длины полученных кусков трости, ранжированных в порядке уменьшения длины:

$$L_1 \geq L_2 \geq \dots \geq L_k. \quad (22)$$

- 3 Вычисляем математическое ожидание длин L_i :

$$l_i = E(L_i) = \frac{1}{k} \sum_{j=i}^k \frac{1}{j}. \quad (23)$$

Например, для $k = 3$:

$$l_1 = \frac{1}{3} \left(1 + \frac{1}{2} + \frac{1}{3} \right) \approx 0.611, \quad l_2 = \frac{1}{3} \left(\frac{1}{2} + \frac{1}{3} \right) \approx 0.278, \quad l_3 = \frac{1}{3} \left(\frac{1}{3} \right) \approx 0.111.$$

- 4 **Правило сломанной трости:** n -й собственный вектор сохраняем в списке главных компонент, если

$$\frac{\lambda_1}{\lambda} > l_1, \dots, \frac{\lambda_n}{\lambda} > l_n \quad (24)$$

Пусть получен набор собственных чисел:

$$\Lambda = \{10568, 6840, 5928, 4654, 4484, 3827, 3644, 2263, 1947, 1527, 1489, 1357, 1017, 959, 667, 537, 398, 315, 233, 9\} \quad (25)$$

Пусть получен набор собственных чисел:

$$\Lambda = \{10568, 6840, 5928, 4654, 4484, 3827, 3644, 2263, 1947, 1527, 1489, 1357, 1017, 959, 667, 537, 398, 315, 233, 9\} \quad (25)$$

Нормируем его (на рисунке 8 синего цвета) и вычисляем математическое ожидание длин l_i (на рисунке 8 красного цвета):

0.20068, 0.1299, 0.1126, 0.0883, 0.0851, 0.0727, 0.0692, 0.0429, 0.0369, 0.0290, 0.0283, 0.0258, 0.0193,
0.0182, 0.0127, 0.0102, 0.0075, 0.0059, 0.0044, 0.00017

Пусть получен набор собственных чисел:

$$\Lambda = \{10568, 6840, 5928, 4654, 4484, 3827, 3644, 2263, 1947, 1527, 1489, 1357, 1017, 959, 667, 537, 398, 315, 233, 9\} \quad (25)$$

Нормируем его (на рисунке 8 синего цвета) и вычисляем математическое ожидание длин l_i (на рисунке 8 красного цвета):

0.20068, 0.1299, 0.1126, 0.0883, 0.0851, 0.0727, 0.0692, 0.0429, 0.0369, 0.0290, 0.0283, 0.0258, 0.0193,
0.0182, 0.0127, 0.0102, 0.0075, 0.0059, 0.0044, 0.00017

Выше красной линии первых 7 собственных чисел.

Пусть получен набор собственных чисел:

$$\Lambda = \{10568, 6840, 5928, 4654, 4484, 3827, 3644, 2263, 1947, 1527, 1489, 1357, 1017, 959, 667, 537, 398, 315, 233, 9\} \quad (25)$$

Нормируем его (на рисунке 8 синего цвета) и вычисляем математическое ожидание длин l_i (на рисунке 8 красного цвета):

$$0.20068, 0.1299, 0.1126, 0.0883, 0.0851, 0.0727, 0.0692, 0.0429, 0.0369, 0.0290, 0.0283, 0.0258, 0.0193, \\ 0.0182, 0.0127, 0.0102, 0.0075, 0.0059, 0.0044, 0.00017$$

Выше красной линии первых 7 собственных чисел.

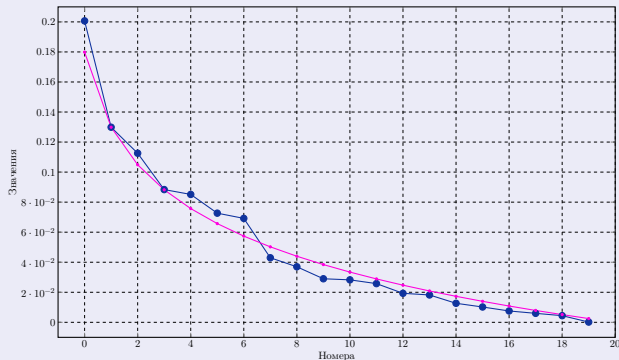


Рис. 8

Правило Кайзера — простейший и старейший метод отбора главных компонент. Иногда в литературе: **критерий Кайзера-Гутмана**.

Правило Кайзера — простейший и старейший метод отбора главных компонент. Иногда в литературе: **критерий Кайзера-Гутмана**.


Сохраняем в списке главных компонент n -й собственный вектор, если

$$\lambda_n > \frac{1}{k} \sum_{i=1}^k \lambda_i. \quad (26)$$

Правило Кайзера — простейший и старейший метод отбора главных компонент. Иногда в литературе: **критерий Кайзера-Гутмана**.

Сохраняем в списке главных компонент n -й собственный вектор, если


$$\lambda_n > \frac{1}{k} \sum_{i=1}^k \lambda_i. \quad (26)$$


 Правило Кайзера хорошо работает в простых случаях, когда есть несколько главных компонент с λ_i , намного превосходящими среднее значение, а остальные собственные числа меньше него.

Правило Кайзера — простейший и старейший метод отбора главных компонент. Иногда в литературе: **критерий Кайзера-Гутмана**.

Сохраняем в списке главных компонент n -й собственный вектор, если

$$\lambda_n > \frac{1}{k} \sum_{i=1}^k \lambda_i. \quad (26)$$

 Правило Кайзера хорошо работает в простых случаях, когда есть несколько главных компонент с λ_i , намного превосходящими среднее значение, а остальные собственные числа меньше него.

 На рисунке 9 иллюстрация правила Кайзера для предыдущего примера (26). По горизонтали номера главных компонент, по вертикали — их нормированные значения. Среднее изображено красной линией. Выше него первых 7 собственных чисел.

Правило Кайзера — простейший и старейший метод отбора главных компонент. Иногда в литературе: **критерий Кайзера-Гутмана**.

Сохраняем в списке главных компонент n -й собственный вектор, если

$$\lambda_n > \frac{1}{k} \sum_{i=1}^k \lambda_i. \quad (26)$$

Правило Кайзера хорошо работает в простых случаях, когда есть несколько главных компонент с λ_i , намного превосходящими среднее значение, а остальные собственные числа меньше него.

На рисунке 9 иллюстрация правила Кайзера для предыдущего примера (26). По горизонтали номера главных компонент, по вертикали — их нормированные значения. Среднее изображено красной линией. Выше него первых 7 собственных чисел.

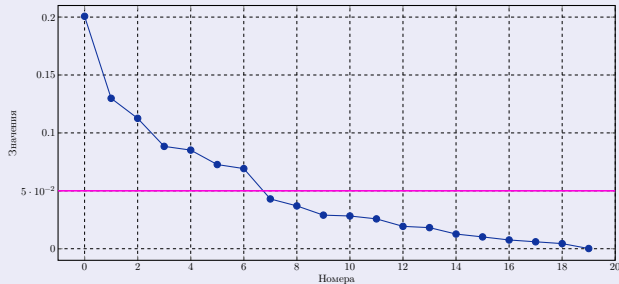


Рис. 9. Иллюстрация правила Кайзера



- 1 Вычисляем число обусловленности μ ковариационной матрицы Σ :

$$\mu(\Sigma) = \frac{\lambda_{\max}}{\lambda_{\min}} = \frac{\lambda_1}{\lambda_k} \geq 1; \quad (27)$$

- 1 Вычисляем число обусловленности μ ковариационной матрицы Σ :

$$\mu(\Sigma) = \frac{\lambda_{\max}}{\lambda_{\min}} = \frac{\lambda_1}{\lambda_k} \geq 1; \quad (27)$$

- 2 Выбираем значение порога $\gamma > 1$.

- 1 Вычисляем число обусловленности μ ковариационной матрицы Σ :

$$\mu(\Sigma) = \frac{\lambda_{\max}}{\lambda_{\min}} = \frac{\lambda_1}{\lambda_k} \geq 1; \quad (27)$$

- 2 Выбираем значение порога $\gamma > 1$.
- 3 Оставляем компоненты, для которых

$$\lambda_i > \frac{\lambda_1}{\gamma}, \quad (28)$$

- 1 Вычисляем число обусловленности μ ковариационной матрицы Σ :

$$\mu(\Sigma) = \frac{\lambda_{\max}}{\lambda_{\min}} = \frac{\lambda_1}{\lambda_k} \geq 1; \quad (27)$$

- 2 Выбираем значение порога $\gamma > 1$.
- 3 Оставляем компоненты, для которых

$$\lambda_i > \frac{\lambda_1}{\gamma}, \quad (28)$$

- 1 Выбор порога γ определяется спецификой задачи. Обычно выбирают $\gamma = 10$.

- 1 Вычисляем число обусловленности μ ковариационной матрицы Σ :

$$\mu(\Sigma) = \frac{\lambda_{\max}}{\lambda_{\min}} = \frac{\lambda_1}{\lambda_k} \geq 1; \quad (27)$$

- 2 Выбираем значение порога $\gamma > 1$.
- 3 Оставляем компоненты, для которых

$$\lambda_i > \frac{\lambda_1}{\gamma}, \quad (28)$$

- 1 Выбор порога γ определяется спецификой задачи. Обычно выбирают $\gamma = 10$.

Рассмотрим собственные числа из ранее рассмотренного примера (16)

$$\lambda_1 = 2.1169, \lambda_2 = 0.8554, \lambda_3 = 0.4817, \lambda_4 = 0.3340.$$

- 1 Вычисляем число обусловленности μ ковариационной матрицы Σ :

$$\mu(\Sigma) = \frac{\lambda_{\max}}{\lambda_{\min}} = \frac{\lambda_1}{\lambda_k} \geq 1; \quad (27)$$

- 2 Выбираем значение порога $\gamma > 1$.
- 3 Оставляем компоненты, для которых

$$\lambda_i > \frac{\lambda_1}{\gamma}, \quad (28)$$

- 1 Выбор порога γ определяется спецификой задачи. Обычно выбирают $\gamma = 10$.

Рассмотрим собственные числа из ранее рассмотренного примера (16)

$$\lambda_1 = 2.1169, \lambda_2 = 0.8554, \lambda_3 = 0.4817, \lambda_4 = 0.3340.$$

Тогда число обусловленности:

$$\mu = \frac{\lambda_1}{\lambda_4} = \frac{2.1169}{0.3340} = 6.338.$$

- 1 Вычисляем число обусловленности μ ковариационной матрицы Σ :

$$\mu(\Sigma) = \frac{\lambda_{\max}}{\lambda_{\min}} = \frac{\lambda_1}{\lambda_k} \geq 1; \quad (27)$$

- 2 Выбираем значение порога $\gamma > 1$.
- 3 Оставляем компоненты, для которых

$$\lambda_i > \frac{\lambda_1}{\gamma}, \quad (28)$$

- 1 Выбор порога γ определяется спецификой задачи. Обычно выбирают $\gamma = 10$.

Рассмотрим собственные числа из ранее рассмотренного примера (16)

$$\lambda_1 = 2.1169, \lambda_2 = 0.8554, \lambda_3 = 0.4817, \lambda_4 = 0.3340.$$

Тогда число обусловленности:

$$\mu = \frac{\lambda_1}{\lambda_4} = \frac{2.1169}{0.3340} = 6.338.$$

При $\gamma = 10$ выше порога $0.1 \times \lambda_1 = 0.21169$ все 4 собственных числа.

- 1 Вычисляем число обусловленности μ ковариационной матрицы Σ :

$$\mu(\Sigma) = \frac{\lambda_{\max}}{\lambda_{\min}} = \frac{\lambda_1}{\lambda_k} \geq 1; \quad (27)$$

- 2 Выбираем значение порога $\gamma > 1$.
- 3 Оставляем компоненты, для которых

$$\lambda_i > \frac{\lambda_1}{\gamma}, \quad (28)$$

- 1 Выбор порога γ определяется спецификой задачи. Обычно выбирают $\gamma = 10$.

Рассмотрим собственные числа из ранее рассмотренного примера (16)

$$\lambda_1 = 2.1169, \lambda_2 = 0.8554, \lambda_3 = 0.4817, \lambda_4 = 0.3340.$$

Тогда число обусловленности:

$$\mu = \frac{\lambda_1}{\lambda_4} = \frac{2.1169}{0.3340} = 6.338.$$

При $\gamma = 10$ выше порога $0.1 \times \lambda_1 = 0.21169$ все 4 собственных числа.

При $\gamma = 5$ выше порога $0.2 \times \lambda_1 = 0.42338$ первые 3 собственных числа.

- 1 Вычисляем число обусловленности μ ковариационной матрицы Σ :

$$\mu(\Sigma) = \frac{\lambda_{\max}}{\lambda_{\min}} = \frac{\lambda_1}{\lambda_k} \geq 1; \quad (27)$$

- 2 Выбираем значение порога $\gamma > 1$.
- 3 Оставляем компоненты, для которых

$$\lambda_i > \frac{\lambda_1}{\gamma}, \quad (28)$$

- 1 Выбор порога γ определяется спецификой задачи. Обычно выбирают $\gamma = 10$.

Рассмотрим собственные числа из ранее рассмотренного примера (16)

$$\lambda_1 = 2.1169, \lambda_2 = 0.8554, \lambda_3 = 0.4817, \lambda_4 = 0.3340.$$

Тогда число обусловленности:

$$\mu = \frac{\lambda_1}{\lambda_4} = \frac{2.1169}{0.3340} = 6.338.$$

При $\gamma = 10$ выше порога $0.1 \times \lambda_1 = 0.21169$ все 4 собственных числа.

При $\gamma = 5$ выше порога $0.2 \times \lambda_1 = 0.42338$ первые 3 собственных числа.

При $\gamma = 4$ выше порога $0.25 \times \lambda_1 = 0.5292$ первые 2 собственных числа.

Достоинства

- 1 Нет необходимости вручную устанавливать параметры метода. Окончательный результат зависит только от данных и не зависит от пользователя.

Достоинства

- 1 Нет необходимости вручную устанавливать параметры метода. Окончательный результат зависит только от данных и не зависит от пользователя.
- 2 Ортогональность между главными компонентами уменьшает взаимное влияние между исходными элементами данных.

Достоинства

- 1 Нет необходимости вручную устанавливать параметры метода. Окончательный результат зависит только от данных и не зависит от пользователя.
- 2 Ортогональность между главными компонентами уменьшает взаимное влияние между исходными элементами данных.
- 3 Метод расчета прост и легко реализуем, сокращение вычислительных затрат при обработке данных.

Достоинства

- 1 Нет необходимости вручную устанавливать параметры метода. Окончательный результат зависит только от данных и не зависит от пользователя.
- 2 Ортогональность между главными компонентами уменьшает взаимное влияние между исходными элементами данных.
- 3 Метод расчета прост и легко реализуем, сокращение вычислительных затрат при обработке данных.
- 4 Сжатие данных для более эффективного хранения информации.

Достоинства

- 1 Нет необходимости вручную устанавливать параметры метода. Окончательный результат зависит только от данных и не зависит от пользователя.
- 2 Ортогональность между главными компонентами уменьшает взаимное влияние между исходными элементами данных.
- 3 Метод расчета прост и легко реализуем, сокращение вычислительных затрат при обработке данных.
- 4 Сжатие данных для более эффективного хранения информации.
- 5 Борьба с *переобучением*.

Достоинства

- 1 Нет необходимости вручную устанавливать параметры метода. Окончательный результат зависит только от данных и не зависит от пользователя.
- 2 Ортогональность между главными компонентами уменьшает взаимное влияние между исходными элементами данных.
- 3 Метод расчета прост и легко реализуем, сокращение вычислительных затрат при обработке данных.
- 4 Сжатие данных для более эффективного хранения информации.
- 5 Борьба с *переобучением*.
- 6 Визуализация данных. Проектирование выборки на двух-трехмерное пространство позволяет графически представить выборку.

Достоинства

- 1 Нет необходимости вручную устанавливать параметры метода. Окончательный результат зависит только от данных и не зависит от пользователя.
- 2 Ортогональность между главными компонентами уменьшает взаимное влияние между исходными элементами данных.
- 3 Метод расчета прост и легко реализуем, сокращение вычислительных затрат при обработке данных.
- 4 Сжатие данных для более эффективного хранения информации.
- 5 Борьба с *переобучением*.
- 6 Визуализация данных. Проектирование выборки на двух-трехмерное пространство позволяет графически представить выборку.

Недостатки

- 1 Отбрасываемые компоненты с небольшой дисперсией часто могут содержать важную информацию о различиях выборки.

Достоинства

- 1 Нет необходимости вручную устанавливать параметры метода. Окончательный результат зависит только от данных и не зависит от пользователя.
- 2 Ортогональность между главными компонентами уменьшает взаимное влияние между исходными элементами данных.
- 3 Метод расчета прост и легко реализуем, сокращение вычислительных затрат при обработке данных.
- 4 Сжатие данных для более эффективного хранения информации.
- 5 Борьба с *переобучением*.
- 6 Визуализация данных. Проектирование выборки на двух-трехмерное пространство позволяет графически представить выборку.

Недостатки

- 1 Отбрасываемые компоненты с небольшой дисперсией часто могут содержать важную информацию о различиях выборки.
- 2 Высокая чувствительность к выбросам в данных.

Достоинства

- 1 Нет необходимости вручную устанавливать параметры метода. Окончательный результат зависит только от данных и не зависит от пользователя.
- 2 Ортогональность между главными компонентами уменьшает взаимное влияние между исходными элементами данных.
- 3 Метод расчета прост и легко реализуем, сокращение вычислительных затрат при обработке данных.
- 4 Сжатие данных для более эффективного хранения информации.
- 5 Борьба с *переобучением*.
- 6 Визуализация данных. Проектирование выборки на двух-трехмерное пространство позволяет графически представить выборку.

Недостатки

- 1 Отбрасываемые компоненты с небольшой дисперсией часто могут содержать важную информацию о различиях выборки.
- 2 Высокая чувствительность к выбросам в данных.
- 3 Если существуют одинаковые собственные вектора, то тогда гиперплоскость проекции определяется не однозначно.

Достоинства

- 1 Нет необходимости вручную устанавливать параметры метода. Окончательный результат зависит только от данных и не зависит от пользователя.
- 2 Ортогональность между главными компонентами уменьшает взаимное влияние между исходными элементами данных.
- 3 Метод расчета прост и легко реализуем, сокращение вычислительных затрат при обработке данных.
- 4 Сжатие данных для более эффективного хранения информации.
- 5 Борьба с *переобучением*.
- 6 Визуализация данных. Проектирование выборки на двух-трехмерное пространство позволяет графически представить выборку.

Недостатки

- 1 Отбрасываемые компоненты с небольшой дисперсией часто могут содержать важную информацию о различиях выборки.
- 2 Высокая чувствительность к выбросам в данных.
- 3 Если существуют одинаковые собственные вектора, то тогда гиперплоскость проекции определяется не однозначно.
- 4 Прямые и плоскости не всегда обеспечивают хорошую аппроксимацию. Например, данные могут с хорошей точностью следовать какой-нибудь кривой, а эта кривая может быть сложно расположена в пространстве данных.

 Самое распространенное использование РСА — решение **проклятия размерности**.

 Самое распространенное использование РСА — решение **проклятия размерности**.

Это проблема, связанная с экспоненциальным возрастанием количества данных из-за увеличения размерности пространства.

 Самое распространенное использование РСА — решение **проклятия размерности**.

Это проблема, связанная с экспоненциальным возрастанием количества данных из-за увеличения размерности пространства.

Термин был введен Ричардом Беллманом в 1961 г.

 Самое распространенное использование РСА — решение **проклятия размерности**.

Это проблема, связанная с экспоненциальным возрастанием количества данных из-за увеличения размерности пространства.

Термин был введен Ричардом Беллманом в 1961 г.


РСА позволяет уменьшить размер признаков без потери слишком большого количества информации.

 Самое распространенное использование РСА — решение **проклятия размерности**.

Это проблема, связанная с экспоненциальным возрастанием количества данных из-за увеличения размерности пространства.

Термин был введен Ричардом Беллманом в 1961 г.

РСА позволяет уменьшить размер признаков без потери слишком большого количества информации.

 **Сжатие изображений.** РСА можно использовать для уменьшения размера изображения без существенного влияния на качество. Помимо простого уменьшения размера, это полезно для алгоритмов классификации изображений.

 Самое распространенное использование РСА — решение **проклятия размерности**.

Это проблема, связанная с экспоненциальным возрастанием количества данных из-за увеличения размерности пространства.

Термин был введен Ричардом Беллманом в 1961 г.

РСА позволяет уменьшить размер признаков без потери слишком большого количества информации.

1 Сжатие изображений. РСА можно использовать для уменьшения размера изображения без существенного влияния на качество. Помимо простого уменьшения размера, это полезно для алгоритмов классификации изображений.

2 Визуализация многомерных данных. РСА можно использовать для визуализации многомерных данных в двух или трех измерениях, что упрощает их понимание и интерпретацию.

i Самое распространенное использование РСА — решение **проклятия размерности**.

Это проблема, связанная с экспоненциальным возрастанием количества данных из-за увеличения размерности пространства.

Термин был введен Ричардом Беллманом в 1961 г.

РСА позволяет уменьшить размер признаков без потери слишком большого количества информации.

1 Сжатие изображений. РСА можно использовать для уменьшения размера изображения без существенного влияния на качество. Помимо простого уменьшения размера, это полезно для алгоритмов классификации изображений.

2 Визуализация многомерных данных. РСА можно использовать для визуализации многомерных данных в двух или трех измерениях, что упрощает их понимание и интерпретацию.

3 Финансы и экономика. Многие модели прогнозирования цен на акции основаны на оценке ковариационных матриц. Однако это может быть затруднительно с многомерными данными. РСА можно использовать для сокращения данных, чтобы решить эту проблему.

i Самое распространенное использование РСА — решение **проклятия размерности**.

Это проблема, связанная с экспоненциальным возрастанием количества данных из-за увеличения размерности пространства.

Термин был введен Ричардом Беллманом в 1961 г.

РСА позволяет уменьшить размер признаков без потери слишком большого количества информации.

1 Сжатие изображений. РСА можно использовать для уменьшения размера изображения без существенного влияния на качество. Помимо простого уменьшения размера, это полезно для алгоритмов классификации изображений.

2 Визуализация многомерных данных. РСА можно использовать для визуализации многомерных данных в двух или трех измерениях, что упрощает их понимание и интерпретацию.

3 Финансы и экономика. Многие модели прогнозирования цен на акции основаны на оценке ковариационных матриц. Однако это может быть затруднительно с многомерными данными. РСА можно использовать для сокращения данных, чтобы решить эту проблему.

4 Биология. РСА используется для анализа данных об экспрессии генов, что позволяет идентифицировать и выделять гены, которые в наибольшей степени способствуют наблюдаемым вариациям.

i Самое распространенное использование РСА — решение **проклятия размерности**.

Это проблема, связанная с экспоненциальным возрастанием количества данных из-за увеличения размерности пространства.

Термин был введен Ричардом Беллманом в 1961 г.

РСА позволяет уменьшить размер признаков без потери слишком большого количества информации.

- 1 Сжатие изображений.** РСА можно использовать для уменьшения размера изображения без существенного влияния на качество. Помимо простого уменьшения размера, это полезно для алгоритмов классификации изображений.
- 2 Визуализация многомерных данных.** РСА можно использовать для визуализации многомерных данных в двух или трех измерениях, что упрощает их понимание и интерпретацию.
- 3 Финансы и экономика.** Многие модели прогнозирования цен на акции основаны на оценке ковариационных матриц. Однако это может быть затруднительно с многомерными данными. РСА можно использовать для сокращения данных, чтобы решить эту проблему.
- 4 Биология.** РСА используется для анализа данных об экспрессии генов, что позволяет идентифицировать и выделять гены, которые в наибольшей степени способствуют наблюдаемым вариациям.
- 5 Медицина.** В моделях здравоохранения используются многомерные наборы данных, поскольку на результаты здравоохранения влияет множество факторов. РСА предоставляет метод уменьшения размерности, сохраняя при этом соответствующую дисперсию.

i Самое распространенное использование РСА — решение **проклятия размерности**.

Это проблема, связанная с экспоненциальным возрастанием количества данных из-за увеличения размерности пространства.

Термин был введен Ричардом Беллманом в 1961 г.

РСА позволяет уменьшить размер признаков без потери слишком большого количества информации.

1 Сжатие изображений. РСА можно использовать для уменьшения размера изображения без существенного влияния на качество. Помимо простого уменьшения размера, это полезно для алгоритмов классификации изображений.

2 Визуализация многомерных данных. РСА можно использовать для визуализации многомерных данных в двух или трех измерениях, что упрощает их понимание и интерпретацию.

3 Финансы и экономика. Многие модели прогнозирования цен на акции основаны на оценке ковариационных матриц. Однако это может быть затруднительно с многомерными данными. РСА можно использовать для сокращения данных, чтобы решить эту проблему.

4 Биология. РСА используется для анализа данных об экспрессии генов, что позволяет идентифицировать и выделять гены, которые в наибольшей степени способствуют наблюдаемым вариациям.

5 Медицина. В моделях здравоохранения используются многомерные наборы данных, поскольку на результаты здравоохранения влияет множество факторов. РСА предоставляет метод уменьшения размерности, сохраняя при этом соответствующую дисперсию.

6 Пищевые продукты: РСА позволяет проводить классификации пищевых продуктов в тех случаях, когда для характеристики их свойств используется одновременно большое число признаков.

РСА. Пример для Ирисов Фишера

```
from sklearn import datasets
iris = datasets.load_iris() # Using the dataset "Irises"
iris_X = iris.data

# Importing the decomposition module
from sklearn import decomposition

# Initializing PCA (two components)
pca = decomposition.PCA(n_components=2)

# Initializing PCA (saving 98% of the information)
#pca = decomposition.PCA(n_components=.98)

iris_X_prime = pca.fit_transform(iris_X)
iris_X_prime.shape

# Graphics
from matplotlib import pyplot as plt
f = plt.figure(figsize=(5, 5))
ax = f.add_subplot(111)
ax.scatter(iris_X_prime[:,0], iris_X_prime[:, 1], c=iris.target)
ax.set_title("PCA: 2 components")

# Percentage of information in 2 new components
print(pca.explained_variance_ratio_.sum())
```

```
from sklearn import datasets
iris = datasets.load_iris() # Using the dataset "Iris"
iris_X = iris.data

# Importing the decomposition module
from sklearn import decomposition

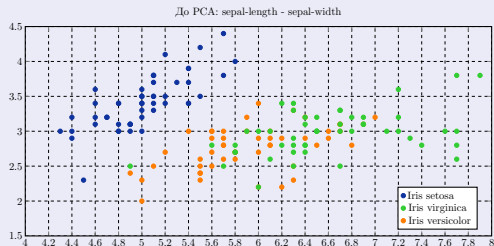
# Initializing PCA (two components)
pca = decomposition.PCA(n_components=2)

# Initializing PCA (saving 98% of the information)
#pca = decomposition.PCA(n_components=.98)

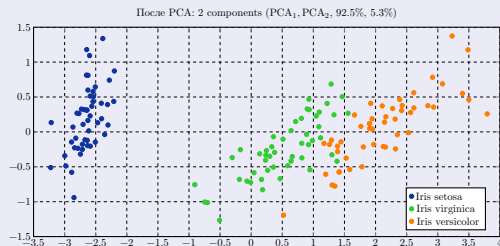
iris_X_prime = pca.fit_transform(iris_X)
iris_X_prime.shape

# Graphics
from matplotlib import pyplot as plt
f = plt.figure(figsize=(5, 5))
ax = f.add_subplot(111)
ax.scatter(iris_X_prime[:,0], iris_X_prime[:, 1], c=iris.target)
ax.set_title("PCA: 2 components")

# Percentage of information in 2 new components
print(pca.explained_variance_ratio_.sum())
```



(a)



(b)



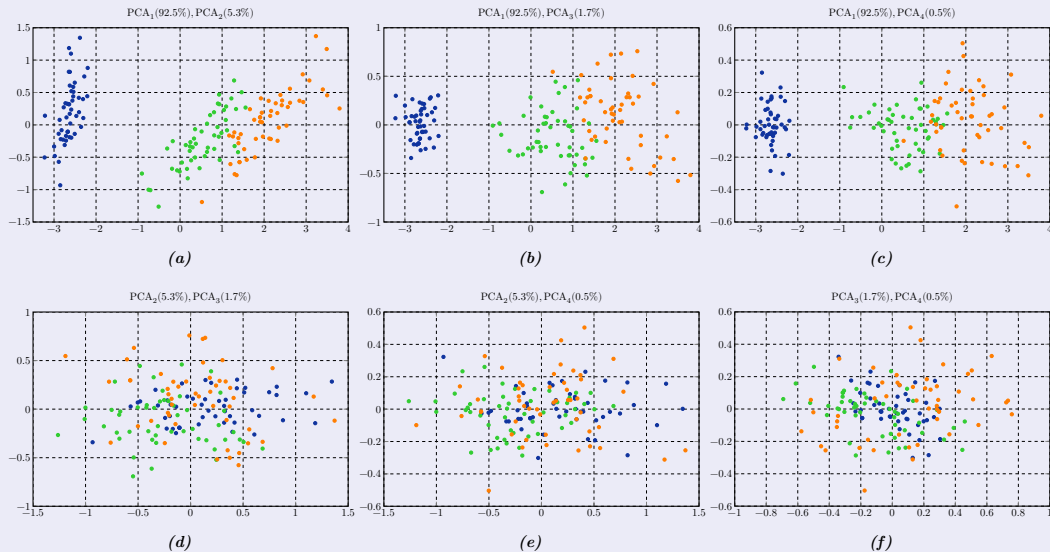


Рис. 11. Проекция на главные компоненты: *setosa* — синий, *virginica* — зеленый, *versicolor* — оранжевый



- 1 Импортируем соответствующие библиотеки, которые помогут нам выполнить анализ и визуализировать результаты.

- 1 Импортируем соответствующие библиотеки, которые помогут нам выполнить анализ и визуализировать результаты.
- 2 Будем использовать набор данных под названием «Диабет» библиотеки *sklearn*.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_diabetes
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
```

1 Импортируем соответствующие библиотеки, которые помогут нам выполнить анализ и визуализировать результаты.

2 Будем использовать набор данных под названием «Диабет» библиотеки *sklearn*.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_diabetes
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
```

3 Преобразуем набор данных в формат **DataFrame Pandas**.

```
diabetes = load_diabetes()
df = pd.DataFrame(data=diabetes.data, columns=diabetes.feature_names)
```


- 1 Необходимо масштабировать переменные перед проведением анализа, чтобы избежать ошибочных результатов РСА из-за различий в единицах измерения.

- 1 Необходимо масштабировать переменные перед проведением анализа, чтобы избежать ошибочных результатов РСА из-за различий в единицах измерения.
- 2 Для масштабирования данных в диапазон $(0, 1)$, будем использовать класс **StandardScaler** из *sklearn*.

- 1 Необходимо масштабировать переменные перед проведением анализа, чтобы избежать ошибочных результатов РСА из-за различий в единицах измерения.
- 2 Для масштабирования данных в диапазон $(0, 1)$, будем использовать класс **StandardScaler** из *sklearn*.
- 3 Сначала создадим объект класса StandardScaler, затем используем его для преобразования данных в новый диапазон.

```
scaler = StandardScaler()  
scaler.fit(df)  
Diabetes_scaled = scaler.transform(df)
```

- 1 Необходимо масштабировать переменные перед проведением анализа, чтобы избежать ошибочных результатов РСА из-за различий в единицах измерения.
- 2 Для масштабирования данных в диапазон $(0, 1)$, будем использовать класс **StandardScaler** из *sklearn*.
- 3 Сначала создадим объект класса StandardScaler, затем используем его для преобразования данных в новый диапазон.

```
scaler = StandardScaler()  
scaler.fit(df)  
Diabetes_scaled = scaler.transform(df)
```
- 4 В результате получаем двумерный массив NumPy (442 строки и 10 столбцов).

1 Необходимо масштабировать переменные перед проведением анализа, чтобы избежать ошибочных результатов PCA из-за различий в единицах измерения.

2 Для масштабирования данных в диапазон $(0, 1)$, будем использовать класс **StandardScaler** из *sklearn*.

3 Сначала создадим объект класса **StandardScaler**, затем используем его для преобразования данных в новый диапазон.

```
scaler = StandardScaler()  
scaler.fit(df)  
Diabetes_scaled = scaler.transform(df)
```

4 В результате получаем двумерный массив NumPy (442 строки и 10 столбцов).

5 Преобразуем его в DataFrame Pandas.

```
dataframe_scaled = pd.DataFrame(data=Diabetes_scaled, columns=diabetes.feature_names)  
dataframe_scaled.head(6)
```


1 Необходимо масштабировать переменные перед проведением анализа, чтобы избежать ошибочных результатов PCA из-за различий в единицах измерения.

2 Для масштабирования данных в диапазон (0, 1), будем использовать класс **StandardScaler** из *sklearn*.

3 Сначала создадим объект класса StandardScaler, затем используем его для преобразования данных в новый диапазон.

```
scaler = StandardScaler()
scaler.fit(df)
Diabetes_scaled = scaler.transform(df)
```

4 В результате получаем двумерный массив NumPy (442 строки и 10 столбцов).

5 Преобразуем его в DataFrame Pandas.

```
dataframe_scaled = pd.DataFrame(data=Diabetes_scaled, columns=diabetes.feature_names)
dataframe_scaled.head(6)
```

6 Первые шесть строк показаны в таблице.

	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6
0	0.800500	1.065488	1.297088	0.459840	-0.92974	-0.73206	-0.91245	-0.05449	0.418530	-0.37098
1	-0.03956	-0.93853	-1.08218	-0.55350	-0.17762	-0.40288	1.564413	-0.83030	-1.43658	-1.93847
2	1.793306	1.065488	0.934533	-0.11921	-0.95867	-0.71889	-0.68024	-0.05449	0.060155	-0.54515
3	-1.87244	-0.93853	-0.24377	-0.77064	0.256292	0.525397	-0.75764	0.721302	0.476982	-0.19682
4	0.113172	-0.93853	-0.76494	0.459840	0.082725	0.327890	0.171177	-0.05449	-0.67250	-0.98056
5	-1.94881	-0.93853	-0.85558	-0.40874	-1.45044	-1.66693	0.867795	-1.60610	-0.86567	-2.02556

1 Необходимо масштабировать переменные перед проведением анализа, чтобы избежать ошибочных результатов PCA из-за различий в единицах измерения.

2 Для масштабирования данных в диапазон (0, 1), будем использовать класс **StandardScaler** из *sklearn*.

3 Сначала создадим объект класса `StandardScaler`, затем используем его для преобразования данных в новый диапазон.

```
scaler = StandardScaler()
scaler.fit(df)
Diabetes_scaled = scaler.transform(df)
```

4 В результате получаем двумерный массив NumPy (442 строки и 10 столбцов).

5 Преобразуем его в DataFrame Pandas.

```
dataframe_scaled = pd.DataFrame(data=Diabetes_scaled, columns=diabetes.feature_names)
dataframe_scaled.head(6)
```

6 Первые шесть строк показаны в таблице.

	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6
0	0.800500	1.065488	1.297088	0.459840	-0.92974	-0.73206	-0.91245	-0.05449	0.418530	-0.37098
1	-0.03956	-0.93853	-1.08218	-0.55350	-0.17762	-0.40288	1.564413	-0.83030	-1.43658	-1.93847
2	1.793306	1.065488	0.934533	-0.11921	-0.95867	-0.71889	-0.68024	-0.05449	0.060155	-0.54515
3	-1.87244	-0.93853	-0.24377	-0.77064	0.256292	0.525397	-0.75764	0.721302	0.476982	-0.19682
4	0.113172	-0.93853	-0.76494	0.459840	0.082725	0.327890	0.171177	-0.05449	-0.67250	-0.98056
5	-1.94881	-0.93853	-0.85558	-0.40874	-1.45044	-1.66693	0.867795	-1.60610	-0.86567	-2.02556

7 Теперь у нас есть те же данные, но стандартизированные. Все готово к запуску PCA.

- 1 Проведем РСА для достаточно большого количества компонентов.

- 1 Проведем РСА для достаточно большого количества компонентов.
- 2 Визуализируем собственные значения для каждой компоненты.

- 1 Проведем РСА для достаточно большого количества компонентов.
- 2 Визуализируем собственные значения для каждой компоненты.
- 3 На основании графика подберем оптимальное число.

- 1 Проведем РСА для достаточно большого количества компонентов.
- 2 Визуализируем собственные значения для каждой компоненты.
- 3 На основании графика подберем оптимальное число.
- 4 Для примера запустим наш РСА для 10 компонент.

```
pca = PCA(n_components=10)  
pca.fit_transform(Diabetes_scaled)
```

- 1 Проведем РСА для достаточно большого количества компонентов.
- 2 Визуализируем собственные значения для каждой компоненты.
- 3 На основании графика подберем оптимальное число.
- 4 Для примера запустим наш РСА для 10 компонент.

```
pca = PCA(n_components=10)  
pca.fit_transform(Diabetes_scaled)
```

- 5 Извлекаем собственные значения и дисперсию
- ```
prop_var = pca.explained_variance_ratio_
eigenvalues = pca.explained_variance_
```



- 1 Проведем РСА для достаточно большого количества компонент.
- 2 Визуализируем собственные значения для каждой компоненты.
- 3 На основании графика подберем оптимальное число.
- 4 Для примера запустим наш РСА для 10 компонент.

```
pca = PCA(n_components=10)
pca.fit_transform(Diabetes_scaled)
```

- 5 Извлекаем собственные значения и дисперсию

```
prop_var = pca.explained_variance_ratio_
eigenvalues = pca.explained_variance_
```

- 6 Строим соответствующий график (рисунок 12).

```
PC_numbers = np.arange(pca.n_components_) + 1
plt.plot(PC_numbers, prop_var, 'bo-')
plt.xlabel('Components')
plt.ylabel('Fractions of variance')
plt.grid()
plt.show()
```

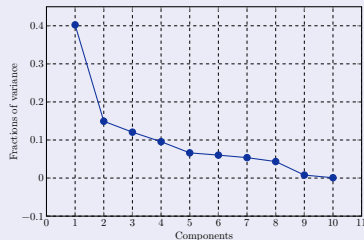


Рис. 12

- 1 Проведем РСА для достаточно большого количества компонент.
- 2 Визуализируем собственные значения для каждой компоненты.
- 3 На основании графика подберем оптимальное число.
- 4 Для примера запустим наш РСА для 10 компонент.

```
pca = PCA(n_components=10)
pca.fit_transform(Diabetes_scaled)
```

- 5 Извлекаем собственные значения и дисперсию

```
prop_var = pca.explained_variance_ratio_
eigenvalues = pca.explained_variance_
```

- 6 Строим соответствующий график (рисунок 12).

```
PC_numbers = np.arange(pca.n_components_) + 1
plt.plot(PC_numbers, prop_var, 'bo-')
plt.xlabel('Components')
plt.ylabel('Fractions of variance')
plt.grid()
plt.show()
```

- 7 По правилу локтя отбираем первые две компоненты.

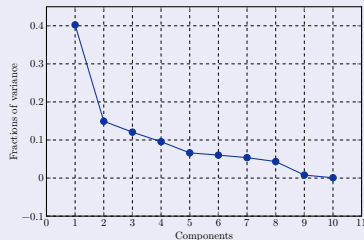


Рис. 12



1 Преобразуем наши переменные в главные компоненты, используя алгоритм РСА.

```
pca = PCA(n_components=2)
PC = pca.fit_transform(Diabetes_scaled)
```

- 1 Преобразуем наши переменные в главные компоненты, используя алгоритм РСА.

```
pca = PCA(n_components=2)
PC = pca.fit_transform(Diabetes_scaled)
```

- 2 Сохраним оценки компонент в DataFrame и проверим их с помощью метода *head()*.

```
pca_diabetes = pd.DataFrame(data = PC, columns = ['PC1', 'PC2'])
pca_diabetes.head(6)
```

❶ Преобразуем наши переменные в главные компоненты, используя алгоритм РСА.

```
pca = PCA(n_components=2)
PC = pca.fit_transform(Diabetes_scaled)
```

❷ Сохраним оценки компонент в DataFrame и проверим их с помощью метода *head()*.

```
pca_diabetes = pd.DataFrame(data = PC, columns = ['PC1', 'PC2'])
pca_diabetes.head(6)
```

|   | PC1      | PC2      |
|---|----------|----------|
| 0 | 0.587199 | -1.94683 |
| 1 | -2.83162 | 1.372081 |
| 2 | 0.272128 | -1.63490 |
| 3 | 0.049281 | 0.382278 |
| 4 | -0.75642 | 0.811960 |
| 5 | -3.96632 | -0.38106 |

1 Преобразуем наши переменные в главные компоненты, используя алгоритм PCA.

```
pca = PCA(n_components=2)
PC = pca.fit_transform(Diabetes_scaled)
```

2 Сохраним оценки компонент в DataFrame и проверим их с помощью метода `head()`.

```
pca_diabetes = pd.DataFrame(data = PC, columns = ['PC1', 'PC2'])
pca_diabetes.head(6)
```

3 Посмотрим новые координаты признаков в двух компонентах: PC1 и PC2.

```
loadings = pd.DataFrame(pca.components_.T, columns=['PC1', 'PC2'], index=diabetes.feature_names)
loadings
```

|   | PC1      | PC2      |
|---|----------|----------|
| 0 | 0.587199 | -1.94683 |
| 1 | -2.83162 | 1.372081 |
| 2 | 0.272128 | -1.63490 |
| 3 | 0.049281 | 0.382278 |
| 4 | -0.75642 | 0.811960 |
| 5 | -3.96632 | -0.38106 |

1 Преобразуем наши переменные в главные компоненты, используя алгоритм PCA.

```
pca = PCA(n_components=2)
PC = pca.fit_transform(Diabetes_scaled)
```

2 Сохраним оценки компонент в DataFrame и проверим их с помощью метода `head()`.

```
pca_diabetes = pd.DataFrame(data = PC, columns = ['PC1', 'PC2'])
pca_diabetes.head(6)
```

3 Посмотрим новые координаты признаков в двух компонентах: PC1 и PC2.

```
loadings = pd.DataFrame(pca.components_.T, columns=['PC1', 'PC2'], index=diabetes.feature_names)
loadings
```

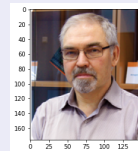
|   | PC1      | PC2      |
|---|----------|----------|
| 0 | 0.587199 | -1.94683 |
| 1 | -2.83162 | 1.372081 |
| 2 | 0.272128 | -1.63490 |
| 3 | 0.049281 | 0.382278 |
| 4 | -0.75642 | 0.811960 |
| 5 | -3.96632 | -0.38106 |

|     | age      | sex      | bmi      | bp       | s1       | s2       | s3       | s4       | s5       | s6       |
|-----|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| PC1 | 0.216430 | 0.186966 | 0.303162 | 0.271737 | 0.343255 | 0.351860 | -0.28243 | 0.428833 | 0.378618 | 0.322182 |
| PC2 | 0.044367 | -0.38654 | -0.15628 | -0.13826 | 0.573026 | 0.455941 | 0.506239 | -0.06818 | -0.02618 | -0.08494 |





- 1 Рассмотрим пример сжатия изображения (рисунок 13:а) с использованием анализа главных компонент.

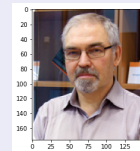


*(а) Исходное изображение*

1 Рассмотрим пример сжатия изображения (рисунок 13:a) с использованием анализа главных компонент.

2 Импортируем необходимые библиотеки

```
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
import numpy as np
from sklearn.decomposition import PCA
```



(a) Исходное изображение

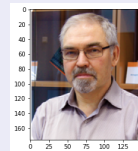
1 Рассмотрим пример сжатия изображения (рисунок 13:а) с использованием анализа главных компонент.

2 Импортируем необходимые библиотеки

```
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
import numpy as np
from sklearn.decomposition import PCA
```

3 Загружаем изображение и выводим его данные:

```
img = mpimg.imread('Rybin_photo.png')
print(img.shape)
```



(а) Исходное изображение

1 Рассмотрим пример сжатия изображения (рисунок 13:a) с использованием анализа главных компонент.

2 Импортируем необходимые библиотеки

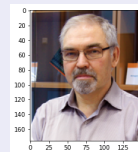
```
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
import numpy as np
from sklearn.decomposition import PCA
```

3 Загружаем изображение и выводим его данные:

```
img = mpimg.imread('Rybin_photo.png')
print(img.shape)
```

4 Изображение имеет форму 176 строк, каждая из которых содержит 148 пикселей и имеет 4 канала. Изменяем размер изображения так, чтобы оно имело формат, необходимый для ввода РСА. Поскольку  $148 * 4 = 592$ , мы изменяем размер изображения до (176, 592):

```
img_r = np.reshape(img, (176, 592))
```



(a) Исходное изображение

1 Рассмотрим пример сжатия изображения (рисунок 13:a) с использованием анализа главных компонент.

2 Импортируем необходимые библиотеки

```
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
import numpy as np
from sklearn.decomposition import PCA
```

3 Загружаем изображение и выводим его данные:

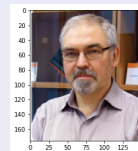
```
img = mpimg.imread('Rybin_photo.png')
print(img.shape)
```

4 Изображение имеет форму 176 строк, каждая из которых содержит 148 пикселей и имеет 4 канала. Изменяем размер изображения так, чтобы оно имело формат, необходимый для ввода PCA. Поскольку  $148 * 4 = 592$ , мы изменяем размер изображения до (176, 592):

```
img_r = np.reshape(img, (176, 592))
```

5 Запускаем PCA с 32 главными компонентами:

```
pca = PCA(32).fit(img_r)
img_transformed = pca.transform(img_r)
print(np.sum(pca.explained_variance_ratio_))
```



(a) Исходное изображение

1 Рассмотрим пример сжатия изображения (рисунок 13:a) с использованием анализа главных компонент.

2 Импортируем необходимые библиотеки

```
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
import numpy as np
from sklearn.decomposition import PCA
```

3 Загружаем изображение и выводим его данные:

```
img = mpimg.imread('Rybin_photo.png')
print(img.shape)
```

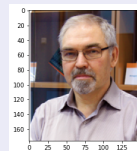
4 Изображение имеет форму 176 строк, каждая из которых содержит 148 пикселей и имеет 4 канала. Изменяем размер изображения так, чтобы оно имело формат, необходимый для ввода РСА. Поскольку  $148 * 4 = 592$ , мы изменяем размер изображения до (176, 592):

```
img_r = np.reshape(img, (176, 592))
```

5 Запускаем PCA с 32 главными компонентами:

```
pca = PCA(32).fit(img_r)
img_transformed = pca.transform(img_r)
print(np.sum(pca.explained_variance_ratio_))
```

6 С помощью этих 32 компонент можно выразить 98.6% дисперсии.



(a) Исходное изображение

1 Рассмотрим пример сжатия изображения (рисунок 13:a) с использованием анализа главных компонент.

2 Импортируем необходимые библиотеки

```
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
import numpy as np
from sklearn.decomposition import PCA
```

3 Загружаем изображение и выводим его данные:

```
img = mpimg.imread('Rybin_photo.png')
print(img.shape)
```

4 Изображение имеет форму 176 строк, каждая из которых содержит 148 пикселей и имеет 4 канала. Изменяем размер изображения так, чтобы оно имело формат, необходимый для ввода РСА. Поскольку  $148 * 4 = 592$ , мы изменяем размер изображения до (176, 592):

```
img_r = np.reshape(img, (176, 592))
```

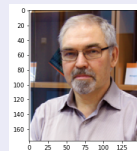
5 Запускаем PCA с 32 главными компонентами:

```
pca = PCA(32).fit(img_r)
img_transformed = pca.transform(img_r)
print(np.sum(pca.explained_variance_ratio_))
```

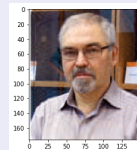
6 С помощью этих 32 компонент можно выразить 98.6% дисперсии.

7 Делаем обратное преобразование изображения и его вывод с помощью *imshow* (рисунок 13:b):

```
temp = pca.inverse_transform(img_transformed)
temp = np.reshape(temp, (176, 148, 4))
plt.imshow(temp)
```



(a) Исходное изображение



(b) 32 компоненты



1 Рассмотрим пример сжатия изображения (рисунок 13:a) с использованием анализа главных компонент.

2 Импортируем необходимые библиотеки

```
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
import numpy as np
from sklearn.decomposition import PCA
```

3 Загружаем изображение и выводим его данные:

```
img = mpimg.imread('Rybin_photo.png')
print(img.shape)
```

4 Изображение имеет форму 176 строк, каждая из которых содержит 148 пикселей и имеет 4 канала. Изменяем размер изображения так, чтобы оно имело формат, необходимый для ввода РСА. Поскольку  $148 * 4 = 592$ , мы изменяем размер изображения до (176, 592):

```
img_r = np.reshape(img, (176, 592))
```

5 Запускаем PCA с 32 главными компонентами:

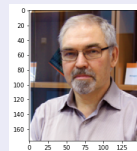
```
pca = PCA(32).fit(img_r)
img_transformed = pca.transform(img_r)
print(np.sum(pca.explained_variance_ratio_))
```

6 С помощью этих 32 компонент можно выразить 98.6% дисперсии.

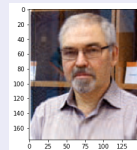
7 Делаем обратное преобразование изображения и его вывод с помощью `imshow` (рисунок 13:b):

```
temp = pca.inverse_transform(img_transformed)
temp = np.reshape(temp, (176, 148, 4))
plt.imshow(temp)
```

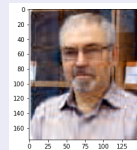
Результат для 16 главных компонент представлен на рисунке 13:c (дисперсия — 95.8%).



(a) Исходное изображение



(b) 32 компоненты



(c) 16 компонент