Министерство науки и высшего образования Российской Федерации

федеральное государственное бюджетное образовательное учреждение

высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

МОРДОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

им. Н.П. ОГАРЁВА»

(ФГБОУ ВО «МГУ им. Н.П. Огарёва»)

Факультет математики и информационных технологий

Кафедра фундаментальной информатики

КУРСОВАЯ РАБОТА

по теме:

ВЕРОЯТНОСТНО-СТАТИСТИЧЕСКИЕ МЕТОДЫ В ЗАДАЧАХ РАСПОЗНАВАНИЯ

Автор курсовой работы  А. Е. Конышев

подпись, дата

Обозначение курсовой работы КР–02069964–02.03.02–хх–23

Направление подготовки 02.03.02 Фундаментальная информатика и информационные технологии

Руководитель работы

канд. физ.-мат. наук, доц.  Н. М. Куляшова

подпись, дата

Оценка

Саранск 2023

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc135760422)

[1 ОСНОВНЫЕ СВЕДЕНИЯ О ВЕРОЯТНОСТНО-СТАТИСТИЧЕСКИХ МЕТОДАХ 5](#_Toc135760423)

[2 ОБЗОР ЧАСТО ПРИМЕНЯЕМЫХ ВЕРОЯТНОСТНО-СТАТИСТИЧЕСКИХ МЕТОДОВ 8](#_Toc135760424)

[2.1 Наивный байесовский классификатор 8](#_Toc135760425)

[2.2 Метод опорных векторов (SVM) 17](#_Toc135760426)

[2.3 Метод главных компонент 22](#_Toc135760427)

[2.3.3 Объяснение работы метода главных компонент на двумерном примере 23](#_Toc135760428)

[2.3.3 Демонстрация работы метода главных компонент 25](#_Toc135760429)

[2.4 Линейная регрессия 31](#_Toc135760430)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 32](#_Toc135760431)

ВВЕДЕНИЕ

Задачи распознавания — это одна из важнейших областей, связанных с обработкой и анализом данных. Они находят применение в различных областях, таких как медицина, банковское дело, компьютерное зрение, робототехника и многих других. Решение этих задач может быть достаточно сложным, и часто требует использования вероятностно-статистических методов.

В настоящее время, вероятностно-статистические методы являются одними из наиболее эффективных и широко используемых методов решения задач распознавания. Они позволяют обрабатывать данные, извлекать из них информацию и принимать решения на основе этой информации.

Существует множество вероятностно-статистических методов, которые используются в задачах распознавания и других областях. Некоторые из них включают в себя:

* наивный байесовский классификатор - простой, но эффективный метод классификации, основанный на теории вероятностей;
* метод опорных векторов (SVM) - метод машинного обучения, который используется для классификации и регрессии. SVM строит гиперплоскость или набор гиперплоскостей в многомерном пространстве, которые максимально разделяют элементы разных классов;
* метод главных компонент (PCA) - метод, используемый для снижения размерности данных путем проекции их на более низкоразмерное пространство. Это помогает устранить шум, улучшить качество данных и ускорить алгоритмы обработки данных;
* линейная регрессия - метод, используемый для анализа отношения между зависимыми и независимыми переменными. Линейная регрессия может быть использована для прогнозирования значений зависимых переменных на основе значений независимых переменных;
* кластерный анализ - метод, используемый для разделения групп объектов на основе сходства между ними. Кластерный анализ может быть использован для классификации данных или для поиска скрытых структур в данных.

Это только некоторые из методов, используемых в задачах распознавания и других областях. В зависимости от конкретной задачи может быть использован один или несколько из этих методов, а также другие методы машинного обучения и статистические методы.

В рамках данной курсовой работы будут рассмотрены различные вероятностно-статистические методы и их применение в задачах распознавания, таких как классификация изображений, распознавание речи, анализ текстов и другие. Будут рассмотрены основные принципы байесовской статистики, метод опорных векторов, метод главных компонент и другие методы, которые широко применяются в задачах распознавания.

Также будет проведен анализ применимости этих методов на реальных данных с использованием языка программирования Python и библиотек для машинного обучения. Будут использованы различные наборы данных, такие как MNIST для распознавания рукописных цифр, CIFAR-10 для классификации изображений и другие. Анализ результатов позволит оценить эффективность и применимость различных методов в реальных задачах.

Итак, данная работа имеет важное практическое значение и может быть полезна для специалистов в области машинного обучения, анализа данных и других смежных областей, которые занимаются задачами распознавания.

# 1 ОСНОВНЫЕ СВЕДЕНИЯ О ВЕРОЯТНОСТНО-СТАТИСТИЧЕСКИХ МЕТОДАХ

Вероятностно-статистические методы — это методы и подходы, основанные на теории вероятностей и математической статистике. Они используются для анализа и обработки данных, а также для решения различных задач, таких как классификация, регрессия, кластеризация и другие.

Вероятностно-статистические методы могут использоваться для прогнозирования будущих значений, моделирования зависимостей между переменными, оценки параметров распределений и других статистических характеристик, а также для поиска скрытых закономерностей в данных.

Некоторые из основных вероятностно-статистических методов, используемых в машинном обучении, включают в себя байесовскую статистику, методы регрессии, методы классификации, методы кластеризации и методы снижения размерности данных.

Байесовская статистика — это метод статистического вывода, который основан на теореме Байеса. Он используется для оценки вероятности гипотезы на основе имеющихся данных. Байесовские методы могут использоваться для классификации, регрессии и других задач.

Методы регрессии — это методы, используемые для анализа отношения между зависимыми и независимыми переменными. Они могут быть использованы для прогнозирования значений зависимых переменных на основе значений независимых переменных.

Методы классификации — это методы, используемые для разделения объектов на группы на основе их свойств. Они могут быть использованы для классификации изображений, распознавания речи, анализа текстов и других задач.

Методы кластеризации — это методы, используемые для разделения объектов на группы на основе сходства между ними. Они могут быть использованы для классификации данных или для поиска скрытых структур в данных.

Методы снижения размерности данных — это методы, используемые для уменьшения размерности данных путем проекции их на более низкоразмерное пространство. Это может помочь устранить шум, улучшить качество данных и ускорить алгоритмы обработки данных.

Вероятностно-статистические методы являются важным инструментом в машинном обучении и анализе данных и широко применяются в различных областях, таких как медицина, банковское дело, компьютерное зрение, робототехника и другие. Важно отметить, что выбор конкретного вероятностно-статистического метода зависит от конкретной задачи и особенностей данных, а также от целей и требований заказчика. Корректный выбор метода и оценка его эффективности являются важными задачами при решении задач распознавания и других задач анализа данных.

Многие вероятностно-статистические методы реализованы в библиотеках для машинного обучения, таких как Scikit-learn и TensorFlow. Использование этих библиотек может значительно упростить процесс анализа данных и решения задач распознавания. Однако, важно иметь понимание основных принципов и методов, чтобы правильно интерпретировать результаты и выбрать наиболее подходящий метод для конкретной задачи.

Задачи распознавания — это задачи, связанные с выделением информации из различных источников данных, таких как изображения, звуковые сигналы, тексты и другие данные. Целью задач распознавания является автоматическое определение характеристик или свойств объектов на основе их представления в виде данных.

Примеры задач распознавания включают в себя:

* распознавание рукописных цифр на изображениях,
* распознавание лиц на изображениях или видео,
* распознавание речи и преобразование речи в текст,
* анализ текстов и определение их темы или настроения,
* классификация изображений на основе их содержания, например, классификация изображений животных, растений или транспортных средств.

Для решения задач распознавания часто используются методы машинного обучения, такие как нейронные сети, метод опорных векторов, байесовские классификаторы, а также вероятностно-статистические методы и другие. Кроме того, для решения задач распознавания также могут быть использованы методы обработки сигналов и изображений, такие как фильтры и преобразования Фурье.

# 2 ОБЗОР ЧАСТО ПРИМЕНЯЕМЫХ ВЕРОЯТНОСТНО-СТАТИСТИЧЕСКИХ МЕТОДОВ

## 2.1 Наивный байесовский классификатор

Наивный байесовский классификатор — это алгоритм машинного обучения, основанный на байесовской статистике и используемый для решения задач классификации. Он основан на предположении о независимости признаков объекта друг от друга.

Алгоритм наивного байесовского классификатора состоит из следующих шагов:

1. подготовка обучающей выборки;
2. оценка априорной вероятности каждого класса;
3. оценка условной вероятности каждого признака для каждого класса;
4. определение апостериорной вероятности для каждого класса на основе формулы Байеса;
5. классификация объекта на основе максимальной апостериорной вероятности.

Наивный байесовский классификатор может быть использован для решения задач бинарной и многоклассовой классификации. Для бинарной классификации алгоритм использует один классификатор, который выдаёт вероятность принадлежности объекта к классу 1. Для многоклассовой классификации используется несколько классификаторов, каждый из которых относится к своему классу.

Преимущества наивного байесовского классификатора:

* простота и скорость алгоритма;
* хорошая производительность на больших объемах данных;
* может быть использован для категориальных и непрерывных признаков.

Недостатки наивного байесовского классификатора:

* ограниченность предположения о независимости признаков;
* может приводить к неправильным результатам, если условная независимость не выполняется;
* требует большого количества данных для точных оценок вероятностей.

Это основные принципы и свойства наивного байесовского классификатора. Однако, существует множество вариаций наивного байесовского классификатора, таких как мультиномиальный, бернуллиев, гауссовский и другие. Каждая из этих вариаций имеет свои особенности и может быть использована в различных задачах классификации.

Реализовать наивный байесовский классификатор на Python с использованием библиотеки Scikit-learn для многоклассовой классификации текстовых данных можно следующим образом:

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.metrics import accuracy\_score

def naive\_bayesian\_classifier(X\_train, y\_train, X\_test, y\_test):

    # Создаем векторизатор

    vectorizer = CountVectorizer()

    # Преобразуем текст в числовые признаки

    X\_train\_vec = vectorizer.fit\_transform(X\_train)

    # Преобразуем текст тестовой выборки в числовые признаки

    X\_test\_vec = vectorizer.transform(X\_test)

    # Создаем классификатор на основе наивного байесовского алгоритма

    clf = MultinomialNB()

    # Обучаем классификатор на обучающей выборке

    clf.fit(X\_train\_vec, y\_train)

    # Предсказываем метки классов для тестовой выборки

    y\_pred = clf.predict(X\_test\_vec)

    return (y\_pred, accuracy\_score(y\_test, y\_pred))

# Создаем обучающую выборку

X\_train = ["This is a good movie",

           "This is a bad movie",

           "This movie is not good",

           "This movie is good",

           "This movie is very bad"]

# Задаем метки классов для обучающей выборки

y\_train = ["positive", "negative", "negative", "positive", "negative"]

# Создаем тестовую выборку

X\_test = ["This is a very good movie",

          "This is a very bad movie"]

# Вычисляем точность классификации

y\_test = ['positive', 'negative']

output = naive\_bayesian\_classifier(X\_train, y\_train, X\_test, y\_test)

print(

    'Ожидаемые данные:',

    y\_test,

    '\nРезультаты классификации:',

    output[0],

    '\nТочность классификации:',

    output[1]

)

Этот пример демонстрирует, как использовать наивный байесовский классификатор для классификации текстовых данных на два класса (положительные и отрицательные отзывы). Векторизатор CountVectorizer используется для преобразования текста в числовые признаки, а классификатор MultinomialNB используется для обучения и прогнозирования меток классов. Результаты классификации выводятся на экран, а также вычисляется точность классификации на тестовой выборке.

Результат работы программы:

Ожидаемые данные: ['positive', 'negative']

Результаты классификации: ['positive' 'negative']

Точность классификации: 1.0

Вывод: точность работы классификатора равна 1 – это означает что все отзывы правильно сгруппированы. Стоить отметить, что это довольно простой пример, основанный к тому же на небольшой выборке исходных данных. При использовании большого числа данных для обучения придется вручную классифицировать отзывы.

Второй пример: классификация текстовых документов на несколько категорий. Исходные данные для классификации текстовых документов были взяты из корпуса текстовых документов "20 Newsgroups", который является стандартным набором данных для задач классификации текстовых документов. Этот корпус содержит текстовые документы из 20 различных категорий новостных групп, таких как компьютерное железо, религия, политика, спорт и т.д.

Корпус "20 Newsgroups" был собран в 1995 году для проведения экспериментов в области обработки естественного языка и классификации текстовых документов. С тех пор он был использован во многих исследованиях и стал стандартным набором данных для задач классификации текстовых документов. В настоящее время этот набор данных доступен для загрузки на многих ресурсах в Интернете.

Код программы:

from sklearn.datasets import fetch\_20newsgroups

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.metrics import classification\_report

# Загружаем данные (категории новостей) из библиотеки Scikit-learn

newsgroups\_train = fetch\_20newsgroups(subset='train')

newsgroups\_test = fetch\_20newsgroups(subset='test')

# Преобразуем текст в числовые признаки с помощью векторизатора CountVectorizer

vectorizer = CountVectorizer()

X\_train = vectorizer.fit\_transform(newsgroups\_train.data)

X\_test = vectorizer.transform(newsgroups\_test.data)

# Создаем классификатор на основе наивного байесовского алгоритма

clf = MultinomialNB()

# Обучаем классификатор на обучающей выборке

clf.fit(X\_train, newsgroups\_train.target)

# Предсказываем метки классов для тестовой выборки

y\_pred = clf.predict(X\_test)

# Выводим отчет о классификации, включая точность, полноту и F-меру

print(classification\_report(

    newsgroups\_test.target,

    y\_pred,

    target\_names=newsgroups\_test.target\_names

))

Этот пример демонстрирует, как использовать наивный байесовский классификатор для классификации текстовых документов на несколько категорий (категории новостей). Векторизатор CountVectorizer используется для преобразования текста в числовые признаки, а классификатор MultinomialNB используется для обучения и прогнозирования меток классов. Отчет о классификации выводится на экран, включая точность, полноту и F-меру.

Результат работы программы:

precision recall f1-score support

alt.atheism 0.79 0.77 0.78 319

comp.graphics 0.67 0.74 0.70 389

comp.os.ms-windows.misc 0.20 0.00 0.01 394

comp.sys.ibm.pc.hardware 0.56 0.77 0.65 392

comp.sys.mac.hardware 0.84 0.75 0.79 385

comp.windows.x 0.65 0.84 0.73 395

misc.forsale 0.93 0.65 0.77 390

rec.autos 0.87 0.91 0.89 396

rec.motorcycles 0.96 0.92 0.94 398

rec.sport.baseball 0.96 0.87 0.91 397

rec.sport.hockey 0.93 0.96 0.95 399

sci.crypt 0.67 0.95 0.78 396

sci.electronics 0.79 0.66 0.72 393

sci.med 0.87 0.82 0.85 396

sci.space 0.83 0.89 0.86 394

soc.religion.christian 0.70 0.96 0.81 398

talk.politics.guns 0.69 0.91 0.79 364

talk.politics.mideast 0.85 0.94 0.89 376

talk.politics.misc 0.58 0.63 0.60 310

talk.religion.misc 0.89 0.33 0.49 251

accuracy 0.77 7532

macro avg 0.76 0.76 0.75 7532

weighted avg 0.76 0.77 0.75 7532

Интерпретация выходных данных: эти выходные данные представляют собой отчет о классификации текстовых документов на несколько категорий с использованием многоклассового классификатора на основе наивного байесовского алгоритма.

Каждая строка отчета соответствует категории, а каждый столбец — это мера (precision, recall, F1-score) для этой категории. В таблице показаны значения мер точности (precision), полноты (recall) и F1-меры (F1-score) для каждой категории.

Например, первая строка отчета: означает, что классификатор правильно классифицировал 79% текстовых документов в категории "alt.atheism" (точность), из всех документов, которые были помечены как "alt.atheism", классификатор правильно идентифицировал 77% (полнота), а F1-мера (среднее гармоническое точности и полноты) равна 0.78.

В конце отчета также показана общая точность классификации (accuracy), которая определяется как отношение числа правильно классифицированных документов ко всем документам. В этом случае, точность классификации равна 0.77, что означает, что 77% всех текстовых документов были правильно классифицированы.

Пример 3 – классификация спам-сообщений. Исходные данные взяты из репозитория на Github по ссылке: <https://github.com/rajeevratan84/datascienceforbusiness/blob/master/spam.csv>. В этом файле около 5500 строчек данных. Он часто применяется для машинного обучения и анализа данных.

Код программы:

import pandas as pd

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.metrics import accuracy\_score, confusion\_matrix

# Загружаем данные (сообщения) из CSV-файла

data = pd.read\_csv('D:/GH/university/term6/Kursach/spam.csv', encoding='latin-1')

# Разделяем данные на обучающую и тестовую выборки

X\_train = data.v2[:4400].values

X\_test = data.v2[4400:].values

y\_train = data.v1[:4400].values

y\_test = data.v1[4400:].values

# Преобразуем текст в числовые признаки с помощью векторизатора CountVectorizer

vectorizer = CountVectorizer()

X\_train\_vec = vectorizer.fit\_transform(X\_train)

X\_test\_vec = vectorizer.transform(X\_test)

# Создаем классификатор на основе наивного байесовского алгоритма

clf = MultinomialNB()

# Обучаем классификатор на обучающей выборке

clf.fit(X\_train\_vec, y\_train)

# Предсказываем метки классов для тестовой выборки

y\_pred = clf.predict(X\_test\_vec)

# Выводим матрицу ошибок и точность классификации

print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

Этот пример демонстрирует, как использовать наивный байесовский классификатор для классификации спам-сообщений. Данные (сообщения) загружаются из CSV-файла и разделяются на обучающую и тестовую выборки. Векторизатор CountVectorizer используется для преобразования текста в числовые признаки, а классификатор MultinomialNB используется для обучения и прогнозирования меток классов. Матрица ошибок и точность классификации выводятся на экран.

Результат работы программы:

Confusion Matrix:

[[1015 8]

[ 8 141]]

Accuracy: 0.9863481228668942

Интерпретация выходных данных: матрица ошибок (Confusion matrix) — это таблица, которая показывает число верно и неверно классифицированных примеров в каждой из двух категорий. В данном случае, первая строка и первый столбец матрицы соответствуют категории 0 (не спам), а вторая строка и второй столбец - категории 1 (спам). Таким образом, матрица ошибок имеет следующий вид:

1. [[1015 8] – 1015 не-спам сообщений правильно классифицированы, 8 неправильно классифицированы как спам
2. [ 8 141]] – 141 спам сообщений правильно классифицированы, 8 неправильно классифицированы как не-спам

Точность (Accuracy) — это мера правильности классификации, которая определяется как отношение числа верно классифицированных примеров ко всем примерам. В данном случае, точность равна 0.9863, что означает, что 98,63% всех сообщений были правильно классифицированы.

Подведем итоги: "Наивный байесовский классификатор" — это простой и быстрый алгоритм машинного обучения для классификации текстовых документов на основе вероятностной модели. Алгоритм основан на предположении о независимости факторов, что делает его "наивным".

Основные преимущества наивного байесовского классификатора:

* простота: алгоритм быстро реализуется и обучается.
* низкие вычислительные затраты: алгоритм работает быстро и требует относительно мало ресурсов.
* хорошая производительность: наивный байесовский классификатор может давать хорошие результаты в задачах классификации текстовых документов.
* хорошая интерпретируемость: алгоритм основан на вероятностной модели, что делает его результаты легко интерпретируемыми.

Некоторые из недостатков наивного байесовского классификатора:

* предположение о независимости факторов может быть неверным для некоторых наборов данных.
* алгоритм может давать неоптимальные результаты в задачах, в которых существует много зависимых переменных.
* некоторые типы данных могут не подходить для использования с наивным байесовским классификатором.

Несмотря на эти ограничения, наивный байесовский классификатор по-прежнему широко используется в задачах классификации текстовых документов и считается одним из самых эффективных алгоритмов для этой задачи.

## 2.2 Метод опорных векторов (SVM)

Метод опорных векторов (SVM) может быть применен для различных задач классификации и регрессии, включая распознавание образов, классификацию текстовых данных, биоинформатику, финансовые прогнозы, анализ изображений и многие другие.

SVM может быть особенно полезен в задачах классификации, когда данных много и они линейно разделимы. В таких случаях SVM может дать лучшие результаты, чем наивный байесовский классификатор. Кроме того, SVM может быть использован для работы с нелинейно разделимыми данными, благодаря использованию функций ядра.

Однако, для малых объемов данных, наивный байесовский классификатор может быть более эффективным и простым в использовании, чем SVM. Наивный байесовский классификатор также может дать хорошие результаты для задач классификации текстовых данных и других типов данных, когда данные имеют простую структуру и нет сильной зависимости между признаками.

Таким образом, выбор между SVM и наивным байесовским классификатором зависит от типа задачи и объема данных, которые необходимо обработать. Если данных много, и они сложные, SVM может быть более эффективным. Если же данных мало, и они простые, то наивный байесовский классификатор может быть более подходящим выбором.

Сравним результаты классификации новостных статей, реализованной с помощью метода опорных векторов и наивного байесовского классификатора.

Код программы:

from sklearn.datasets import fetch\_20newsgroups

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.model\_selection import train\_test\_split

from sklearn.svm import SVC

from sklearn.metrics import classification\_report

# Загрузка данных из корпуса "20 Newsgroups"

newsgroups = fetch\_20newsgroups(subset='all')

# Преобразование текстовых данных в числовой формат

vectorizer = TfidfVectorizer()

X = vectorizer.fit\_transform(newsgroups.data)

# Разделение данных на обучающую и тестовую выборки

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, newsgroups.target, test\_size=0.3, random\_state=42)

# Обучение SVM

clf = SVC(kernel='linear', C=1, gamma='auto')

clf.fit(X\_train, y\_train)

# Оценка качества модели на тестовой выборке

y\_pred = clf.predict(X\_test)

print(classification\_report(y\_test, y\_pred, target\_names=newsgroups.target\_names))

Результаты:

precision recall f1-score support

alt.atheism 0.94 0.92 0.93 236

comp.graphics 0.74 0.86 0.79 287

comp.os.ms-windows.misc 0.86 0.84 0.85 290

comp.sys.ibm.pc.hardware 0.76 0.80 0.78 285

comp.sys.mac.hardware 0.91 0.86 0.89 312

comp.windows.x 0.89 0.88 0.88 308

misc.forsale 0.81 0.87 0.84 276

rec.autos 0.93 0.93 0.93 304

rec.motorcycles 1.00 0.95 0.97 279

rec.sport.baseball 0.98 0.97 0.97 308

rec.sport.hockey 0.97 0.96 0.97 309

sci.crypt 0.99 0.96 0.97 290

sci.electronics 0.85 0.86 0.86 304

sci.med 0.96 0.93 0.95 300

sci.space 0.96 0.95 0.95 297

soc.religion.christian 0.95 0.98 0.96 292

talk.politics.guns 0.95 0.96 0.95 270

talk.politics.mideast 1.00 0.98 0.99 272

talk.politics.misc 0.95 0.92 0.93 239

talk.religion.misc 0.92 0.84 0.88 196

accuracy 0.91 5654

macro avg 0.91 0.91 0.91 5654

weighted avg 0.91 0.91 0.91 5654

Выводы: метод опорных векторов справился с задачей классификации статей гораздо лучше, чем НБК. Однако, замечу, на обучение модели потребовалось больше времени – в 5 раз больше.

Пример 2 – распознавание текста. Для распознавания текста необходимо обучить модель на большом числе исходных данных. Для этого можно воспользоваться встроенными в библиотеку sklearn массивом исходных данных. Алгоритм выглядит аналогично примеру с классификацией новостных статей. Поэтому покажу только результат обучения:

Accuracy: 0.9796296296296296

precision recall f1-score support

0 1.00 1.00 1.00 53

1 0.98 0.98 0.98 50

2 0.98 1.00 0.99 47

3 1.00 0.96 0.98 54

4 0.98 0.98 0.98 60

5 0.97 0.97 0.97 66

6 1.00 1.00 1.00 53

7 0.96 0.98 0.97 55

8 0.95 0.98 0.97 43

9 0.97 0.95 0.96 59

accuracy 0.98 540

macro avg 0.98 0.98 0.98 540

weighted avg 0.98 0.98 0.98 540

Чтобы применить обученную модель на живом примере и распознать число на картинке, нам нужно выполнить следующие шаги:

1. Загрузить изображение с числом, которое мы хотим распознать.
2. Обработать изображение, чтобы привести его к формату, который может быть использован в качестве входных данных для модели. Обычно это означает изменение размера изображения и преобразование его в черно-белый формат.
3. Подать обработанное изображение на вход обученной модели и получить предсказание класса числа.
4. Отобразить предсказанное число на экране.

Вот пример, который демонстрирует, как выполнить эти шаги с помощью Python и OpenCV:

import cv2

import numpy as np

import joblib

# Загрузка обученной модели

model = joblib.load('D:/GH/university/term6/Kursach/svm\_model.pkl')

# Загрузка изображения с числом

image = cv2.imread('D:/GH/university/term6/Kursach/number.png')

# Преобразование изображения в черно-белый формат

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

# Изменение размера изображения

resized = cv2.resize(gray, (8, 8), interpolation=cv2.INTER\_AREA)

# Преобразование изображения в одномерный массив

flattened = np.ravel(resized)

# Подача массива на вход модели и получение предсказания класса числа

prediction = model.predict([flattened])

# Отображение предсказанного числа на экране

cv2.putText(image, str(prediction[0]), (10, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 2, (0, 255, 0), 2)

cv2.imshow('Number', image)

cv2.waitKey(0)

cv2.destroyAllWindows()

Результат показан на рисунке 1.

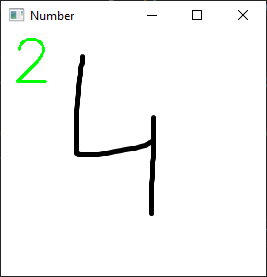


Рисунок 1 – Распознавание рукописного числа на после обучения модели.

Видим, что распознанный результат не совпадает с ожидаемым. Причины могут быть следующими:

* малый объём обучающих данных;
* нерепрезентативный объём данных
* параметры обучаемой модели
* переобучение модели.

Понять какая из них оказала большее влияние на полученный результат иногда оказывается затруднительным. И каждая причина требует своего решения. Порой придется полностью перестроить модель

## 2.3 Метод главных компонент

Метод главных компонент (англ. PCA) — это метод снижения размерности данных, который позволяет представить многомерные данные в виде меньшего числа компонент (переменных), которые объясняют максимальную дисперсию в данных. PCA находит новые признаки, которые являются линейными комбинациями исходных признаков, и выбирает наиболее информативные из них.

PCA может использоваться для различных задач, таких как сжатие данных, визуализация данных, устранение шума и улучшение качества моделирования.

Процесс работы PCA состоит из следующих шагов:

1. стандартизация данных: данные масштабируются таким образом, чтобы каждый признак имел среднее значение, равное 0, и единичную дисперсию;
2. вычисление ковариационной матрицы: ковариационная матрица вычисляется на основе стандартизированных данных и показывает, насколько сильно связаны между собой различные признаки;
3. вычисление собственных значений и собственных векторов: собственные значения и собственные векторы ковариационной матрицы вычисляются с помощью линейной алгебры;
4. сортировка и выбор компонент: собственные значения сортируются по убыванию, и выбираются первые k компонент, которые объясняют наибольшую дисперсию в данных;
5. преобразование данных: исходные данные преобразуются в новое пространство признаков, используя выбранные компоненты.

### 2.3.3 Объяснение работы метода главных компонент на двумерном примере

Для объяснения метода главных компонент использована геометрическая интерпретация. Рассмотрим пример двух переменных, которые могут быть представлены на плоскости.

Каждая строка в таблице соответствует точке на плоскости с координатами, обозначенными пустыми кружками на рисунке 2.1. Максимальное изменение данных происходит вдоль прямой, проведенной через эти точки, которая называется первой главной компонентой (PC1) и выделена синим цветом на рисунке. Все исходные точки проецируются на эту ось и закрашены красным цветом, предполагая, что они должны были лежать на этой новой оси. Отклонения от новой оси можно считать шумом, если быть уверенным, что это ненужная информация. Чтобы проверить, является ли это шумом или все еще важной частью данных, можно найти в остатках ось максимальных изменений, которая называется второй главной компонентой (PC2). Действовать таким образом следует до тех пор, пока шум не станет хаотическим набором величин.

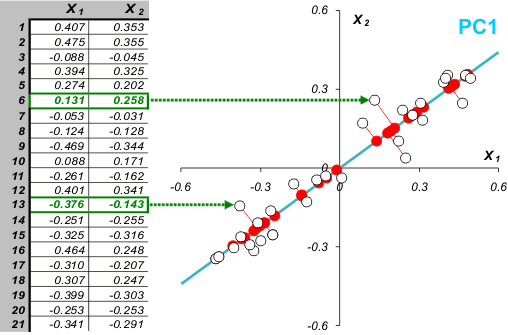


Рисунок 2.1 – Геометрическая интерпретация метода главных компонент на двумерны случай.

Для многомерного случая процесс выделения главных компонент происходит следующим образом:

* 1. определяется центр облака данных, который становится новым началом координат (PC0);
  2. выбирается направление с максимальным изменением данных — это первая главная компонента (PC1);
  3. при наличии шума и отсутствии полной описательности данных выбирается еще одно направление (PC2), перпендикулярное к первой, для описания оставшихся изменений.

В итоге мы преобразуем множество переменных в новое представление с гораздо меньшей размерностью. Часто можно значительно упростить данные, перейдя от 1000 переменных к всего лишь двум. При этом все переменные сохраняются, а незначительные данные отделяются и превращаются в шум. Главные компоненты являются скрытыми переменными, которые управляют данными.

### 2.3.3 Демонстрация работы метода главных компонент

Метод главных компонент иллюстрируется примером, собранным на данных европейского демографического исследования, опубликованных в книге [К. Эсбенсена](https://web.archive.org/web/20090528101113/http:/www.chemometrics.ru/materials/textbooks/references.htm#L3.1).

В исследовательских целях используется набор всего из 32 человек, включающий 16 респондентов из Северной Европы (Скандинавии) и 16 из Южной (Средиземноморья). Для обеспечения баланса между мужчинами и женщинами было выбрано равное количество каждого пола - по 16 человек. Каждый человек характеризуется двенадцатью различными переменными, которые перечислены в таблице 1.

Таблица 1– Переменные, использованные в демографическом анализе

|  |  |
| --- | --- |
| *Height* | Рост: в сантиметрах |
| *Weight* | Вес: в килограммах |
| *Hair* | Волосы: короткие: -1, или длинные: +1 |
| *Shoes* | Обувь: размер по европейскому стандарту |
| *Age* | Возраст: в годах |
| *Income* | Доход: в тысячах евро в год |
| *Beer* | Пиво: потребление в литрах в год |
| *Wine* | Вино: потребление в литрах в год |
| *Sex* | Пол: мужской: -1, или женский: +1 |
| *Strength* | Сила: индекс, основанный на проверке физических способностей |
| *Region* | Регион: север: -1, или юг: +1 |
| *IQ* | Коэффициент интеллекта, измеряемый по стандартному тесту |

Собранные данные проиллюстрированы на рисунке 2 и оформлены в виде таблицы

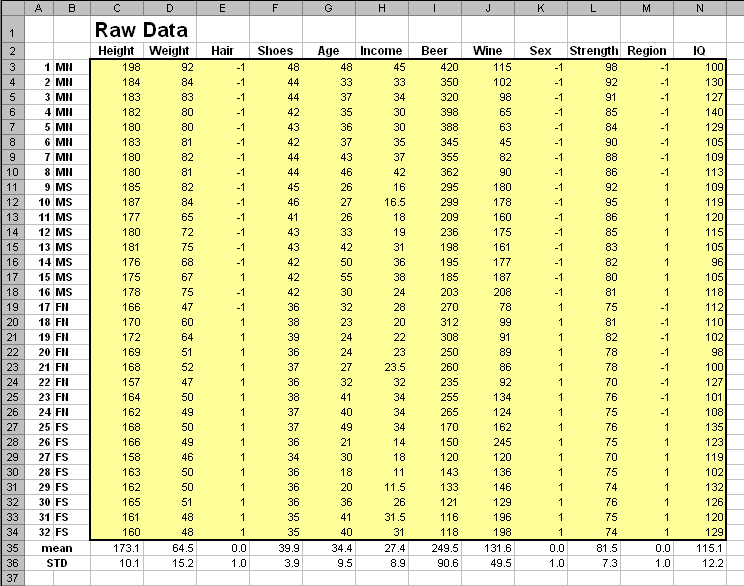


Рисунок 2 – Анализируемые данные

Перед тем как приступить к анализу данных, интересно изучить графики, которые показывают, взаимосвязь между различными переменными. Может ли рост (Height) зависеть от веса (Weight)? Отличается ли потребление вина (Wine) у женщин и мужчин? Есть ли связь между доходом (Income) и возрастом (Age)? Может ли потребление пива (Beer) влиять на вес (Weight)?

На рисунке 3 наглядно продемонстрированы несколько зависимостей. Для того чтобы сделать графики более понятными, использовались единые обозначения: женщины обозначены кружками (F), мужчины - квадратами (M), север - голубым цветом (N), а юг - красным (S).

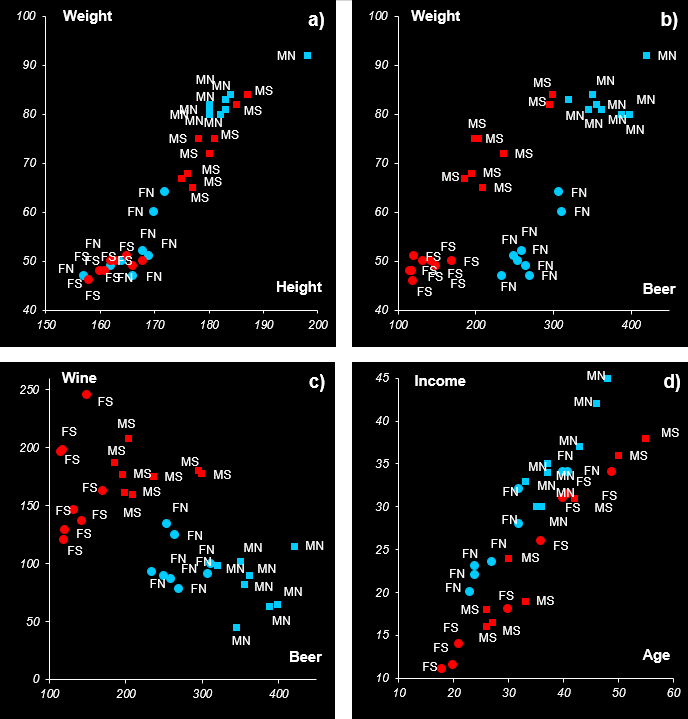


Рисунок 3 – Связи между переменными в примере People. Женщины (F) обозначены кружками ● и ●, а мужчины (M) - квадратами ■ и ■.  Север (N) представлен голубым ■, а юг (S) - красным цветом ●.

Рисунок 3a демонстрирует связь между ростом (Height) и весом (Weight), которая представлена очевидной прямой (положительной) пропорциональностью. Также стоит отметить, что мужчины (M) в целом имеют более высокий рост и больший вес, чем женщины (F).

На рисунке 3b представлена диаграмма зависимости между весом (Weight) и потреблением пива (Beer), которая раскрывает факт, что крупные люди обычно употребляют больше алкоголя, в то время как женщины пьют его в меньшем количестве, чем мужчины. Кроме того, на этой диаграмме можно выделить две группы - северян и южан - которые потребляют разное количество пива при одинаковом весе. Южане, например, пьют меньше пива при сравнимом весе.

На рисунке 3c показана зависимость между потреблением вина (Wine) и пива (Beer), которая демонстрирует отрицательную корреляцию между этими переменными - чем более пьют пива, тем меньше потребляют вина. Как и в случае с диаграммой на Рис. 16b, можно выделить две группы - южан и северян - которые предпочитают разный тип алкоголя.

На рисунке 3d представляет зависимость между возрастом (Age) и доходом (Income) и обнаруживает как положительную, так и отрицательную корреляцию между этими переменными.

Проведем анализ данных методом главных компонент. Сперва преобразуем данные – произведем автошкалирование, так, чтобы среднее значение переменных было равно 0, а отклонение 1.

Построим далее PCA декомпозицию: вычислим все 12 возможных главных компонент. Каждой ячейке будет соответствовать значение исходных данных, обработанное с помощью функции ScoresPCA расширения Chemometrics для MS Excel, в которую будут переданы: необработанный массив данных, автошаклированное значение, 3– тип преобразования(автошкалирование).

Далее стоится матрица нагрузок, размерность которой равна числу главных компонент – 12. Ее возвращает функция [LoadingsPCA](https://web.archive.org/web/20090528101113/http:/www.chemometrics.ru/materials/textbooks/projection.htm#Ch3.2) того же расширения. Ей на вход идет массив необработанных данных, 12 – число главных компонент, 3 – аналогично в функции ScoresPCA)

Теперь можем построить графики, используя, например первые два столбца матрицы декомпозиции

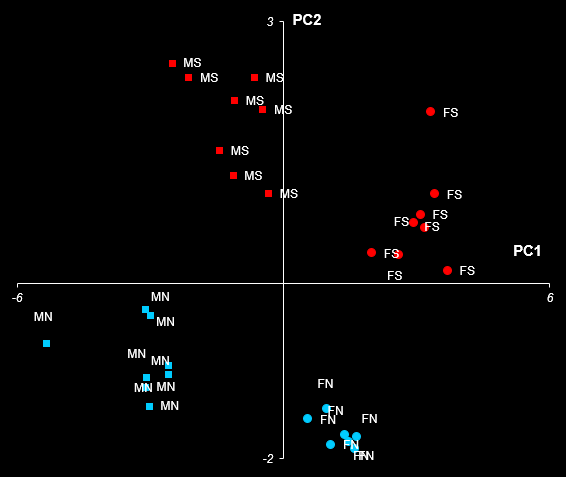


Рисунок чч – График счетов PC1 и PC2

На графике младших счетов PC1-PC2 мы видим четыре отдельные группы, разложенные по четырем квадрантам: слева - женщины (F), справа - мужчины (M), сверху - юг (S), а снизу - север (N). Из этого сразу становится ясен смысл первых двух направлений PC1 и PC2. Первая компонента разделяет людей по полу, а вторая - по месту жительства. Именно эти факторы наиболее сильно влияют на разброс свойств.

Теперь построим грфик на основе других счетов PC3 и PC4. Представим его на рисунке ххх

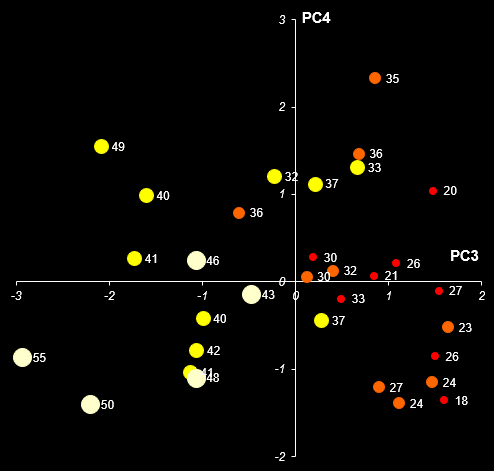


Рисунок ххх – График счетов PC3 и PC4

В этом графике уже не так четко видно разделение на группы, но если внимательно изучить его вместе с таблицей данных, то после некоторых усилий можно сделать вывод, что переменная PC3 разделяет людей на старых и богатых, и молодых и бедных. Мы изменили обозначения, чтобы сделать этот вывод более очевидным. Теперь каждый участник представлен в виде кружка, цвет и размеры которого зависят от их дохода, чем больше и светлей они, тем выше их доход. Также рядом указан возраст каждого участника. Как можно заметить, возраст и доходы уменьшаются слева направо вдоль оси PC3, а вот что означает PC4 пока не ясно.

Представленный пример является лишь небольшой долей того, что может предложить моделирование с помощью PCA. Мы изучили задачу анализа данных, которая не требует последующего использования полученной модели для прогнозирования или классификации.

## 2.4 Линейная регрессия

Линейная регрессия — это метод, который используется для прогнозирования значения зависимой переменной на основе одной или нескольких независимых переменных. Он определяет связь между зависимой переменной и независимыми переменными, используя линейную функцию.

Линейная функция имеет следующий вид:

где – зависимая переменная, которую мы хотим предсказать;

– коэффициенты регрессии, которые мы оцениваем;

– независимые переменные, которые мы используем для прогнозирования.

Коэффициенты регрессии оцениваются таким образом, чтобы минимизировать сумму квадратов ошибок (SSE) между предсказанными значениями и фактическими значениями. SSE — это сумма квадратов разностей между фактическими значениями и предсказанными значениями. Цель линейной регрессии - минимизировать эту ошибку.

Когда мы получаем новое значение для независимых переменных, мы можем использовать коэффициенты регрессии, чтобы предсказать значение зависимой переменной.

Линейная регрессия широко применяется в различных областях, где требуется анализ зависимостей между переменными и прогнозирование будущих значений. Некоторые из наиболее распространенных применений линейной регрессии включают:

* экономика и финансы: линейная регрессия может быть использована для прогнозирования прибыли, спроса на товары и услуги, курса акций и других финансовых показателей.
* наука о данных: линейная регрессия может быть использована для анализа зависимостей между переменными в больших наборах данных, таких как данные клиентов, данные о продажах или данные о климате.
* инженерия: линейная регрессия может быть использована для анализа влияния различных факторов на производительность машин, эффективность процессов и другие показатели в производственных процессах.
* медицина: линейная регрессия может быть использована для анализа влияния различных факторов на здоровье пациентов, таких как возраст, рост, вес и другие физические показатели.
* образование: линейная регрессия может быть использована для анализа влияния различных факторов на успеваемость учащихся, таких как количество часов, потраченных на учебу, количество выполненных домашних заданий и другие.

Это лишь некоторые из множества областей, в которых можно применять линейную регрессию. Из-за своей простоты и эффективности линейная регрессия является широко используемым инструментом в анализе данных и машинном обучении.

## 2.4.1 Практическое применение линейной регрессии для различных задач

Воспользуемся линейной регрессией для предсказания цены на недвижимость. Для её прогнозирования мы можем использовать данные о различных характеристиках недвижимости, таких как размер дома, количество спален, количество ванных комнат, год постройки, расположение и другие. Мы можем использовать эти характеристики как независимые переменные в линейной регрессии, а цену на недвижимость как зависимую переменную.

Для простоты можем взать 3 параметра: площадь квартиры, количество комнат, удаление от центра.

Данные мы сгенерируем сами, используя модуль NumPy. Для обучения модели будет достаточно 1000 домов с разными характеристиками. Затем мы зададим модели интересующие нас параметры жилья, чтобы примерно оценить его цену. Следующий код демонстрирует работу модели, обученной на линейной регрессии.

import numpy as np

from sklearn.linear\_model import LinearRegression

# Генерация данных для 1000 домов

np.random.seed(0)

size = np.round(np.random.uniform(30, 120, 1000))

bedrooms = np.round(np.random.uniform(1, 4, 1000))

distances = np.arange(0, 15.5, 0.5)

distances = np.tile(distances, 33)[:1000]

prices = 1000000 + (size \* 20000) + (bedrooms \* 45000) - (distances \* 100000) + np.random.normal(0, 500000, 1000)

# Создание экземпляра модели линейной регрессии и обучение ее на сгенерированных данных

model = LinearRegression()

model.fit(np.column\_stack((size, bedrooms, distances)), prices)

# Вывод предсказанной цены на экран

print("размер | комнаты | от центра | цена")

data = np.array([[32, 1, 0]])

print(f"{data[0][0]:6d} | {data[0][1]:7d} | {data[0][2]:8.1f} | {model.predict(data)[0]:.2f}")

data = np.array([[45, 2, 0]])

print(f"{data[0][0]:6d} | {data[0][1]:7d} | {data[0][2]:8.1f} | {model.predict(data)[0]:.2f}")

data = np.array([[64, 3, 0]])

print(f"{data[0][0]:6d} | {data[0][1]:7d} | {data[0][2]:8.1f} | {model.predict(data)[0]:.2f}")

data = np.array([[80, 4, 0]])

print(f"{data[0][0]:6d} | {data[0][1]:7d} | {data[0][2]:8.1f} | {model.predict(data)[0]:.2f}")

data = [[80, 4, 2.5]]

print(f"{data[0][0]:6d} | {data[0][1]:7d} | {data[0][2]:8.1f} | {model.predict(data)[0]:.2f}")

data = np.array([[80, 4, 5]])

print(f"{data[0][0]:6d} | {data[0][1]:7d} | {data[0][2]:8.1f} | {model.predict(data)[0]:.2f}")

data = np.array([[45, 3, 5]])

print(f"{data[0][0]:6d} | {data[0][1]:7d} | {data[0][2]:8.1f} | {model.predict(data)[0]:.2f}")

data = np.array([[34, 2, 6]])

print(f"{data[0][0]:6d} | {data[0][1]:7d} | {data[0][2]:8.1f} | {model.predict(data)[0]:.2f}")

Результат работы таков:

размер | комнаты | от центра | цена

32 | 1 | 0.0 | 1713695.93

45 | 2 | 0.0 | 1988531.34

64 | 3 | 0.0 | 2381016.62

80 | 4 | 0.0 | 2714676.96

80 | 4 | 2.5 | 2469359.47

80 | 4 | 5.0 | 2224041.97

45 | 3 | 5.0 | 1517823.69

34 | 2 | 6.0 | 1184077.91

Здесь видим, как цена меняется c ростом числа комнат и, соответственно, площади, в домах, расположенных в центре, а потом цены на жилье с теми же параметрами, но уже на удалении от центра.

Кроме того, модель позволяет получить значения коэффициентов , определяющие вес каждого параметра модели (площади, количеству спален, удалению от центра). В нашем случае они равны:

[ 19608.31247552 19927.34328888 -98126.99781444]

Так же можно получить значение коэффициента , определяющего базовый уровень зависимой переменной, когда независимые переменные равны 0. И ,например, если мы строим модель, которая предсказывает цену на жилье в зависимости от площади квартиры, то будет представлять собой базовую цену на жилье, когда площадь квартиры равна 0 (что, конечно, не имеет практического смысла).

Значение также может использоваться для оценки модели линейной регрессии. Если значение равно нулю или близко к нулю, то это может указывать на то, что модель хорошо подходит для данных. Если же значение существенно отличается от нуля, то это может указывать на то, что модель не подходит для данных или что в данных есть систематическая ошибка. В нашем примере это значение равно 1066302, что не удивительно, исходя из уравнения генерации цены.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

*К. Эсбенсен*. [Анализ многомерных данных](https://web.archive.org/web/20090202110337/http:/rcs.chph.ras.ru/mdabook.htm), сокр. пер. с англ. под ред. О. Родионовой, Из-во ИПХФ РАН, 2005 [K.H. Esbensen. Multivariate Data Analysis - In Practice 4-th Ed., CAMO, 2000]