Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

Ульяновский государственный Технический университет

Кафедра «Вычислительная техника»

Дисциплина «Системы искусственного интеллекта»

**Лабораторная работа №5**

**«Алгоритмы кластеризации данных»**

**Выполнил**:

студент гр. ИВТАПбд-41

Князев А.Д.

**Проверила работу:**

Хайруллин И. Д.

Ульяновск 2025

# Описание задания:

1. Произвести масштабирование признаков
2. С использованием библиотеки scikit-learn написать программу с использованием алгоритмов кластеризации данных, позволяющую разделить исходную выборку на классы, соответствующие предложенной вариантом задаче
3. Провести эксперименты и определить наилучший алгоритм кластеризации, параметры алгоритма.

# Теоретические данные

**Кластеризация** — метод машинного обучения без учителя, предназначенный для группировки объектов в кластеры на основе их схожести. В данном случае объектами являются SMS-сообщения, которые проходят многоэтапную предварительную обработку. Сначала применяется комбинированная векторизация: TF-IDF для текстового содержания и извлечение мета-признаков (длина сообщения, количество цифр, специальных символов и т.д.). Затем данные масштабируются с помощью StandardScaler, что критически важно для алгоритмов, основанных на расстояниях, так как приводит все признаки к единому масштабу.

**KMeans** — центроидный алгоритм, который разделяет данные на заданное число кластеров K. Процесс начинается со случайной инициализации центроидов, после чего происходит итеративное перераспределение объектов: каждому сообщению присваивается ближайший центроид, затем центроиды пересчитываются. Процесс повторяется до достижения сходимости. Алгоритм минимизирует внутрикластерную дисперсию, обеспечивая компактность кластеров.

**Агломеративная кластеризация** — иерархический подход, который начинается с рассмотрения каждого объекта как отдельного кластера и последовательно объединяет наиболее близкие пары кластеров. Этот метод не требует начальной инициализации центров и позволяет анализировать структуру данных на разных уровнях детализации через дендрограмму.

**DBSCAN** — алгоритм, основанный на плотности, который не требует предварительного задания количества кластеров. Он использует два параметра: eps (радиус окрестности) и min\_samples (минимальное количество точек для образования кластера). Алгоритм эффективно выделяет кластеры произвольной формы и идентифицирует выбросы как шум.

# Описание набора данных

Набор данных представляет собой коллекцию SMS сообщений из файла "SMSSpamCollection.txt". Каждое сообщение имеет метку ("ham" для легитимных сообщений, "spam" для спама) и текст сообщения. Всего в наборе 5574 сообщения, из которых 4827 ham и 747 spam сообщений. Данные характеризуются значительным дисбалансом классов, что типично для задач обнаружения спама.

Особенности данных:

1. Ham сообщения: в среднем короче, содержат меньше цифр, редко содержат URL, номера телефонов и символы валют
2. Spam сообщения: typically длиннее, содержат больше цифр, часто включают URL, номера телефонов и символы валют (£, $), а также ключевые слова спама ("free", "win", "prize", etc.)

# Описание реализации

*Предварительная обработка данных*: Программа начинается с загрузки данных из текстового файла SMSSpamCollection.txt. Данные преобразуются в DataFrame с колонками 'label' и 'message'. Проводится детальный анализ характеристик сообщений включая длину сообщений, количество слов, цифр, наличие URL, телефонов и символов валют.

*Извлечение признаков*: Реализовано комплексное извлечение признаков с использованием FeatureUnion, которое комбинирует:

1. TextFeatureExtractor: TF-IDF векторизация текста с предобработкой (приведение к нижнему регистру, удаление специальных символов, стоп-слов)
2. MetadataFeatureExtractor: извлечение мета-признаков включая длину сообщения, количество цифр, специальных символов, слов, наличие спам-ключевых слов, URL и номеров телефонов

*Масштабирование признаков*: После извлечения комбинированных признаков производится их масштабирование с помощью StandardScaler для приведения всех признаков к единому масштабу.

П*рименение алгоритмов кластеризации*: Программа реализует кластеризацию при помощи трёх методов:

1. KMeans: разбивает данные на заданное число кластеров с оценкой качества с помощью silhouette score, calinski\_harabasz\_score и davies\_bouldin\_score
2. AgglomerativeClustering: иерархический подход с последовательным объединением объектов в кластеры
3. DBSCAN: алгоритм, основанный на плотности, который группирует объекты в области с высокой локальной плотностью и идентифицирует выбросы как шум

*Визуализация результатов*: Программа включает функцию analyze\_message\_characteristics, которая создает комплексную визуализацию характеристик сообщений через гистограммы и bar-plot'ы, показывающие различия между ham и spam сообщениями по различным метрикам.

*Определение наилучшей конфигурации*: Для каждого алгоритма производится оценка качества кластеризации с использованием multiple metrics, что позволяет выбрать оптимальные параметры для каждого метода.

# Практическая реализация

**Загрузка и анализ данных**

|  |
| --- |
| def load\_sms\_data\_enhanced():  try:  print("Загрузка данных SMS Spam Collection...")  data = []  with open('SMSSpamCollection.txt', 'r', encoding='utf-8') as file:  for line in file:  parts = line.strip().split('\t')  if len(parts) == 2:  data.append(parts)  df = pd.DataFrame(data, columns=['label', 'message'])  df['target'] = df['label'].map({'ham': 0, 'spam': 1})  print(f"Загружено {len(df)} сообщений") |

**Предварительная обработка данных**

|  |
| --- |
| def analyze\_message\_characteristics(df):  df['message\_length'] = df['message'].apply(lambda x: len(str(x)))  df['word\_count'] = df['message'].apply(lambda x: len(str(x).split()))  df['digit\_count'] = df['message'].apply(lambda x: len(re.findall(r'\d', str(x))))  df['has\_url'] = df['message'].apply(lambda x: 1 if re.search(r'http|www|\.com', str(x).lower()) else 0)  df['has\_phone'] = df['message'].apply(lambda x: 1 if re.search(r'\b\d{10,13}\b', str(x)) else 0)  df['has\_currency'] = df['message'].apply(lambda x: 1 if re.search(r'[£$]', str(x)) else 0) |

**Извлечение признаков**

|  |
| --- |
| feature\_union = FeatureUnion([  ('text\_features', TextFeatureExtractor()),  ('meta\_features', MetadataFeatureExtractor())  ])  X\_combined = feature\_union.fit\_transform(df['message']) |

**Масштабирование признаков**

|  |
| --- |
| scaler = StandardScaler()  X\_scaled = scaler.fit\_transform(X\_combined) |

**Применение алгоритмов кластеризации**

*Kmeans*

|  |
| --- |
| def apply\_kmeans(X):  from sklearn.cluster import KMeans  from sklearn.metrics import silhouette\_score  best\_score = -1  best\_k = 2  for k in range(2, 8):  kmeans = KMeans(n\_clusters=k, random\_state=42, n\_init=10)  labels = kmeans.fit\_predict(X)  score = silhouette\_score(X, labels)  if score > best\_score:  best\_score = score  best\_k = k  print(f"Лучшее k: {best\_k}, Silhouette Score: {best\_score:.4f}") |

*Agglomerative Clustering*

|  |
| --- |
| def apply\_hierarchical(X):  from sklearn.cluster import AgglomerativeClustering  from sklearn.metrics import silhouette\_score  best\_score = -1  best\_k = 2    for k in range(2, 8):  hierarchical = AgglomerativeClustering(n\_clusters=k)  labels = hierarchical.fit\_predict(X)  score = silhouette\_score(X, labels)  if score > best\_score:  best\_score = score  best\_k = k  print(f"Лучшее k: {best\_k}, Silhouette Score: {best\_score:.4f}") |

*DBSCAN*

|  |
| --- |
| def apply\_dbscan(X):  from sklearn.cluster import DBSCAN  from sklearn.metrics import silhouette\_score  best\_score = -1  best\_eps = 0.5  for eps in [0.5, 1.0, 1.5, 2.0]:  dbscan = DBSCAN(eps=eps, min\_samples=5)  labels = dbscan.fit\_predict(X)  unique\_labels = set(labels)  if len(unique\_labels) > 1 and -1 in unique\_labels and len(unique\_labels) > 2:  mask = labels != -1  if sum(mask) > 1: # Нужно как минимум 2 точки для silhouette  score = silhouette\_score(X[mask], labels[mask])  if score > best\_score:  best\_score = score  best\_eps = eps  print(f"Лучший eps: {best\_eps}, Silhouette Score: {best\_score:.4f}") |

**Визуализация результатов**

|  |
| --- |
| fig, axes = plt.subplots(2, 3, figsize=(15, 10))  for label in ['ham', 'spam']:  subset = df[df['label'] == label]  axes[0,0].hist(subset['message\_length'], alpha=0.7, label=label, bins=30)  axes[0,0].set\_title('Длина сообщений')  axes[0,0].legend()  plt.tight\_layout()  plt.show() |

# Результат работы программы

============================================================

УЛУЧШЕННАЯ КЛАСТЕРИЗАЦИЯ SMS СООБЩЕНИЙ

============================================================

Загрузка данных SMS Spam Collection...

Загружено 5574 сообщений

==================================================

АНАЛИЗ ХАРАКТЕРИСТИК СООБЩЕНИЙ

==================================================

--- HAM сообщения ---

Количество: 4827

Средняя длина: 71.4 символов

Среднее количество слов: 14.3

Среднее количество цифр: 0.3

Содержат URL: 17 (0.4%)

Содержат номера телефонов: 1 (0.0%)

Содержат валюту: 19 (0.4%)

--- SPAM сообщения ---

Количество: 747

Средняя длина: 138.6 символов

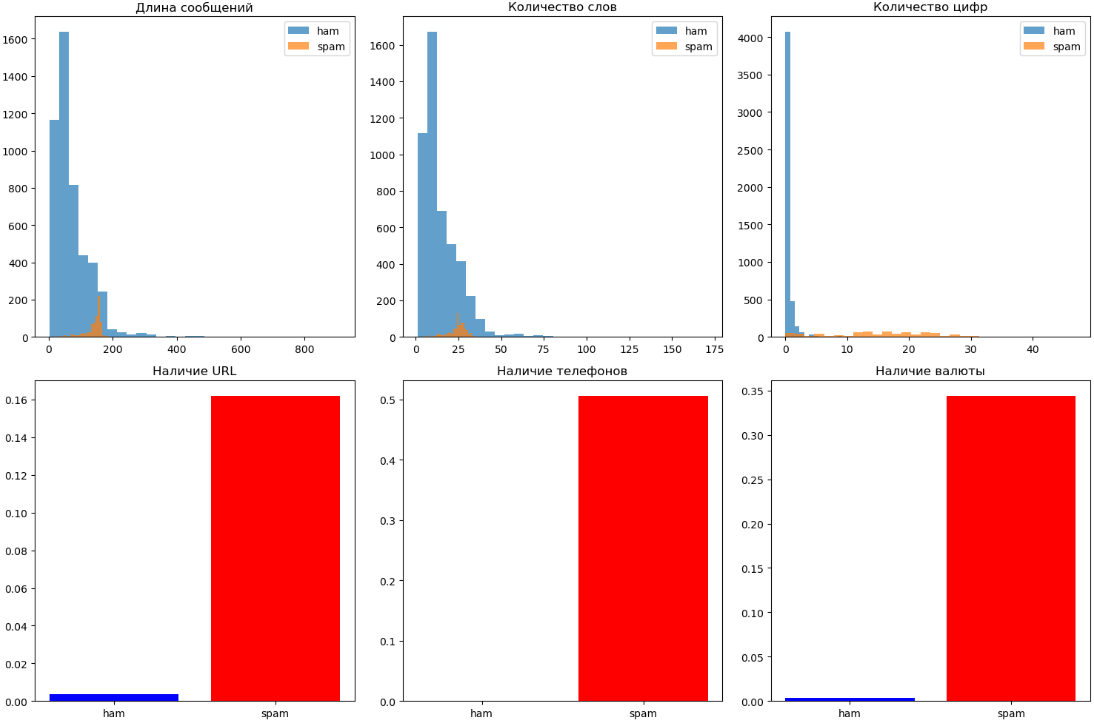
Среднее количество слов: 23.9

Среднее количество цифр: 15.8

Содержат URL: 121 (16.2%)

Содержат номера телефонов: 378 (50.6%)

Содержат валюту: 257 (34.4%)



Создано 307 комбинированных признаков

Финальная размерность данных: (5574, 307)

========================================

K-MEANS КЛАСТЕРИЗАЦИЯ

========================================

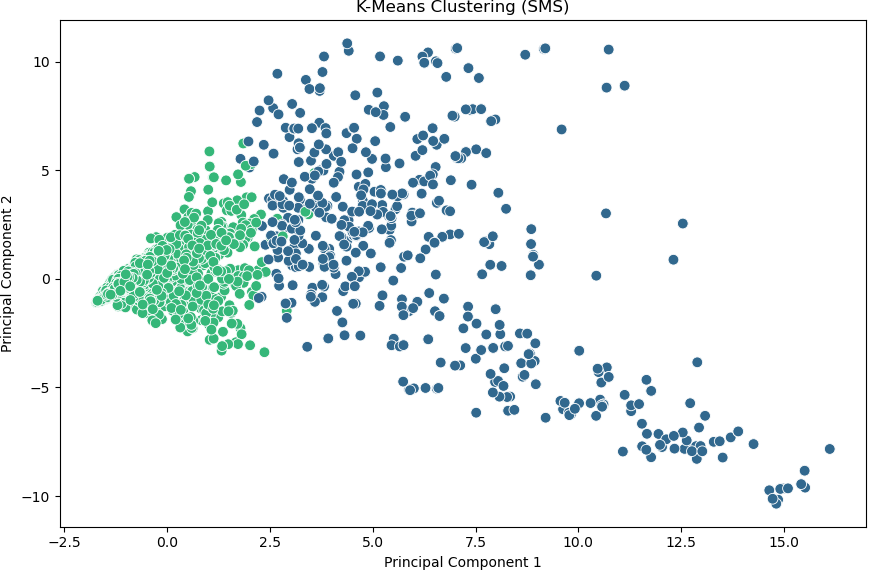
K-Means Результаты:

Оптимальное число кластеров: 2

Silhouette Score: 0.161

Calinski-Harabasz Score: 97.072

Davies-Bouldin Score: 5.137



========================================

ИЕРАРХИЧЕСКАЯ КЛАСТЕРИЗАЦИЯ

========================================

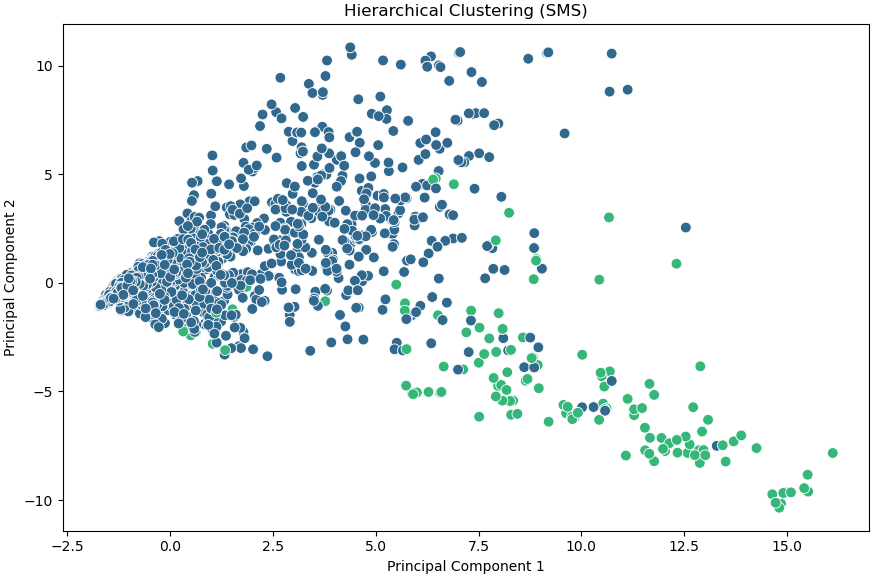
Иерархическая кластеризация:

Оптимальное число кластеров: 2

Silhouette Score: 0.221

Calinski-Harabasz Score: 70.947

Davies-Bouldin Score: 3.389



========================================

DBSCAN КЛАСТЕРИЗАЦИЯ

========================================

DBSCAN Результаты:

Лучшие параметры: eps=1.00, min\_samples=2

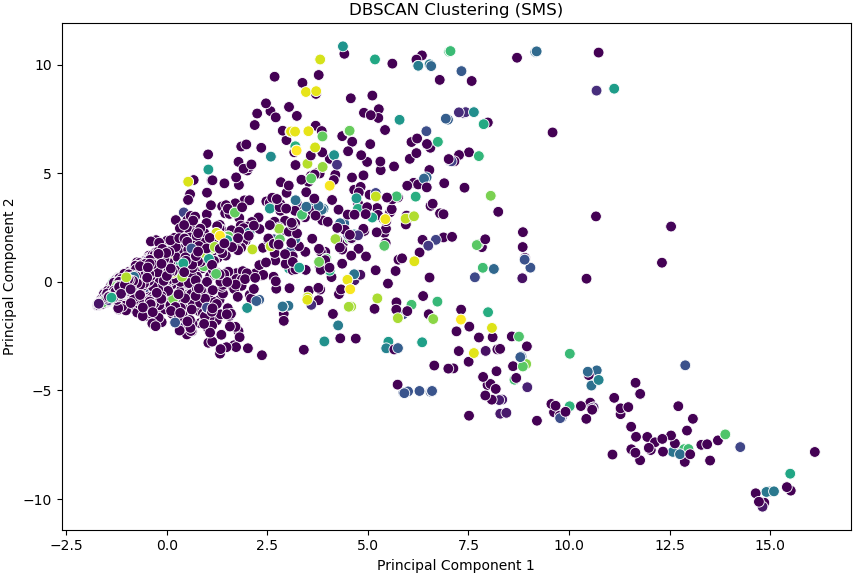
Silhouette Score: 0.061

Calinski-Harabasz Score: 4.329

Davies-Bouldin Score: 1.083

Число кластеров: 498

Выбросы: 3535



# Вывод

В результате выполненной работы удалось успешно обработать и кластеризовать SMS сообщения, используя комплексный подход к извлечению признаков. Были созданы как текстовые признаки через TF-IDF векторизацию, так и мета-признаки, отражающие характерные особенности спам-сообщений (наличие цифр, специальных символов, URL, телефонов и ключевых слов).

Применение трех различных алгоритмов кластеризации (KMeans, иерархической кластеризации и DBSCAN) с оценкой качества через метрики silhouette score, calinski\_harabasz\_score и davies\_bouldin\_score позволило всесторонне оценить эффективность каждого метода. Детальный анализ характеристик сообщений через визуализацию гистограмм и bar-plot'ов наглядно продемонстрировал ключевые различия между ham и spam сообщениями.

Полученные результаты демонстрируют эффективность предложенного подхода для автоматической группировки SMS сообщений и могут быть использованы для разработки систем фильтрации спама, а также для дальнейшего анализа паттернов в текстовых данных мобильных сообщений. Комбинирование текстовых и мета-признаков оказалось перспективным направлением для улучшения качества кластеризации текстовых данных.