САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ ИНФОРМАТИКИ

Отчет по работе
по курсу «Современные инструменты анализа данных»

Тема: Уменьшение размерности данных и сравнение методов

Выполнила:
Гусейнова М. Э.

Проверила:
Добренко Н.В.

Санкт-Петербург
2024 г.

Цель работы. Освоить практические навыки работы с методами уменьшения размерности данных, такими как PCA и t-SNE, а также интерпретации их результатов.

Задание:

1. Сбор и предобработка данных: a) Загрузить данные о погоде за период не менее 6 месяцев с ресурса rp5.ru(город на выбор). b) Провести предварительную обработку данных:
    - Заменить нечисловые данные на числовые (создать шкалу соответствия);
    - Выделить один отчет в сутки;
    - Удалить переменные с нулевой дисперсией;
    - Оставить 10-12 параметров погоды. c) Построить корреляционную матрицу и дать интерпретацию выявленным зависимостям. d) Построить график каменистой осыпи.
2. Реализация и применение PCA: a) Реализовать алгоритм PCA (можно использовать sklearn.decomposition.PCA). b) Вычислить объясненную дисперсию для каждой компоненты. c) Вычислить процент объясненной дисперсии для первых n компонент. d) Построить проекцию данных на новое пространство;
3. Реализация и применение t-SNE: a) Реализовать алгоритм t-SNE (можно использовать sklearn.manifold.TSNE). b) Применить t-SNE с разными значениями перплексии. c) Визуализировать результаты;
4. Сравнительный анализ: a) Сравнить результаты PCA и t-SNE на исследуемых данных. b) Проанализировать преимущества и недостатки каждого метода в контексте ваших данных.

Ход работы:

1 a) Я выбрала город Самару и взяла данные о погоде за 6 месяцев (период выбран рандомно).

```
column_names = [
   "Местное время в Самаре", "T", "Po", "P", "Pa", "U", "DD",
"Ff", "ff10", "ff3",
   "N", "WW", "W1", "W2", "Tn", "Tx", "Cl", "Nh", "H", "Cm", "Ch",
"VV", "Td", "RRR", "tR",
   "E", "Tg", "E'", "sss"
]
# Чтение Excel файла
data = pd.read_excel(
"samara.xls",  # Путь к файлу
skiprows=6,  # Пропустить первые 6 строк
names=column_names)
```

	Местное время в Самаре	T	Po	P	Pa	U	DD	Ff	ff10	ff3	...	Cm	Ch	VV	Td	RRR	tR	E	Tg	E'	sss
0	08.11.2023 22:00	7.6	747.1	759.7	0.6	94.0	Ветер, дующий с западо-юго-запада	2	NaN	NaN	...	Высококучевых, высокослоистых или слоисто-дожд...	Перистых, перисто-кучевых или перисто-слоистых...	10.0	6.7	NaN	NaN	NaN	NaN	NaN	NaN
1	08.11.2023 19:00	8.4	746.5	758.9	0.8	86.0	Ветер, дующий с запада	2	NaN	NaN	...	Высококучевые просвечивающие, расположенные на...	Перистых, перисто-кучевых или перисто-слоистых...	10.0	6.1	Осадков нет	12.0	NaN	NaN	NaN	NaN
2	08.11.2023 16:00	8.9	745.7	758.2	1.3	77.0	Ветер, дующий с запада	2	NaN	NaN	...	Высококучевые просвечивающие, расположенные на...	NaN	10.0	5.1	NaN	NaN	NaN	NaN	NaN	NaN
3	08.11.2023 13:00	8.9	744.4	756.7	1.0	81.0	Ветер, дующий с западо-северо-запада	2	NaN	NaN	...	NaN	NaN	10.0	5.8	NaN	NaN	NaN	NaN	NaN	NaN
4	08.11.2023 10:00	7.3	743.4	755.8	1.0	96.0	Ветер, дующий с западо-юго-запада	2	NaN	NaN	...	NaN	NaN	10.0	6.7	NaN	NaN	NaN	NaN	NaN	NaN

б) Посмотрим на содержимое датасета:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1480 entries, 0 to 1479
Data columns (total 29 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Местное время в Самаре  1480 non-null   object 
 1   T                       1479 non-null   float64
 2   Po                      1480 non-null   float64
 3   P                       1480 non-null   float64
 4   Pa                      1477 non-null   float64
 5   U                       1479 non-null   float64
 6   DD                      1480 non-null   object 
 7   Ff                      1480 non-null   int64  
 8   ff10                    12 non-null     float64
 9   ff3                     89 non-null     float64
 10  N                       1478 non-null   object 
 11  WW                      1480 non-null   object 
 12  W1                      302 non-null    object 
 13  W2                      302 non-null    object 
 14  Tn                      210 non-null    float64
 15  Tx                      185 non-null    float64
 16  Cl                      1195 non-null   object 
 17  Nh                      1195 non-null   object 
 18  H                       1198 non-null   object 
 19  Cm                      1110 non-null   object 
 20  Ch                      885 non-null    object 
 21  VV                      1479 non-null   float64
 22  Td                      1480 non-null   float64
 23  RRR                     370 non-null    object 
 24  tR                      370 non-null    float64
 25  E                       171 non-null    object 
 26  Tg                      171 non-null    float64
 27  E'                      1 non-null      object 
 28  sss                     1 non-null      float64
dtypes: float64(14), int64(1), object(14)
memory usage: 335.4+ KB
```

Как видим, есть столбцы с большим количеством пустых значений. Давайте удалим столбцы, где более 50% данных отсутствует:

```
[62] data = data.loc[:, data.isnull().mean() < 0.5]
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1480 entries, 0 to 1479
Data columns (total 17 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Местное время в Самаре  1480 non-null   object 
 1   T                       1479 non-null   float64
 2   Po                      1480 non-null   float64
 3   P                       1480 non-null   float64
 4   Pa                      1477 non-null   float64
 5   U                       1479 non-null   float64
 6   DD                      1480 non-null   object 
 7   Ff                      1480 non-null   int64  
 8   N                       1478 non-null   object 
 9   WW                      1480 non-null   object 
 10  Cl                      1195 non-null   object 
 11  Nh                      1195 non-null   object 
 12  H                       1198 non-null   object 
 13  Cm                      1110 non-null   object 
 14  Ch                      885 non-null    object 
 15  VV                      1479 non-null   float64
 16  Td                      1480 non-null   float64
dtypes: float64(7), int64(1), object(9)
memory usage: 196.7+ KB
```

Из 29 столбцов осталось 17. Теперь заменим нечисловые данные на числовые.

```
# Заменяем нечисловые данные на числовые
def encode_column(column):
    if column != "Местное время в Самаре":
        if data[column].dtype == 'object':
            unique_values = data[column].dropna().unique()
            mapping = {value: idx for idx, value in enumerate(unique_values)}
            data[column] = data[column].map(mapping)
            return mapping
        return None

encodings = {}
for col in data.columns:
    encoding = encode_column(col)
    if encoding:
        encodings[col] = encoding
```

```
data
```

	Местное время в Самаре	T	Po	P	Pa	U	DD	Ff	N	WW	Cl	Nh	H	Cm	Ch	VV	Td
0	08.11.2023 22:00	7.6	747.1	759.7	0.6	94.0	0	2	0.0	0	0.0	0.0	0.0	0.0	0.0	10.0	6.7
1	08.11.2023 19:00	8.4	746.5	758.9	0.8	86.0	1	2	1.0	1	0.0	1.0	0.0	1.0	0.0	10.0	6.1
2	08.11.2023 16:00	8.9	745.7	758.2	1.3	77.0	1	2	2.0	1	0.0	2.0	0.0	1.0	NaN	10.0	5.1
3	08.11.2023 13:00	8.9	744.4	756.7	1.0	81.0	2	2	2.0	1	0.0	3.0	0.0	NaN	NaN	10.0	5.8
4	08.11.2023 10:00	7.3	743.4	755.8	1.0	96.0	0	2	2.0	1	0.0	3.0	1.0	NaN	NaN	10.0	6.7
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1475	08.05.2023 13:00	10.1	757.4	770.0	-1.1	33.0	11	2	9.0	1	8.0	5.0	0.0	0.0	0.0	10.0	-5.4
1476	08.05.2023 10:00	7.5	758.5	771.3	0.0	37.0	10	3	4.0	1	0.0	6.0	0.0	1.0	0.0	10.0	-6.3
1477	08.05.2023 07:00	4.4	758.5	771.4	0.8	52.0	2	1	4.0	1	4.0	4.0	2.0	0.0	1.0	10.0	-4.8
1478	08.05.2023 04:00	3.1	757.7	770.6	0.8	54.0	2	1	7.0	1	NaN	NaN	NaN	NaN	NaN	10.0	-5.3
1479	08.05.2023 01:00	5.0	756.9	769.7	0.0	47.0	11	1	7.0	1	NaN	NaN	NaN	NaN	NaN	10.0	-5.6

1480 rows × 17 columns

Удалим колонки с нулевой дисперсией:

```
[109] # Удаление колонок с нулевой дисперсией
      zero_var_cols = [col for col in data.columns if data[col].nunique() <= 1]
      data = data.drop(columns=zero_var_cols)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1480 entries, 0 to 1479
Data columns (total 17 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Местное время в Самаре  1480 non-null   object
 1   T                       1479 non-null   float64
 2   Po                      1480 non-null   float64
 3   P                       1480 non-null   float64
 4   Pa                      1477 non-null   float64
 5   U                       1479 non-null   float64
 6   DD                      1480 non-null   int8
 7   Ff                      1480 non-null   int64
 8   N                       1480 non-null   int8
 9   WW                      1480 non-null   int8
 10  Cl                      1480 non-null   int8
 11  Nh                      1480 non-null   int8
 12  H                       1480 non-null   int8
 13  Cm                      1480 non-null   int8
 14  Ch                      1480 non-null   int8
 15  VV                      1479 non-null   float64
 16  Td                      1480 non-null   float64
dtypes: float64(7), int64(1), int8(8), object(1)
memory usage: 115.8+ KB
```

(Все осталось как прежде)

Выделим один отчет в сутки (первый):

```
# Выделяем один отчет в сутки (например, оставляем только первую запись каждого дня)
data['Дата'] = pd.to_datetime(data['Местное время в Самаре'], format = "%d.%m.%Y %H:%M").dt.date
data_daily = data.groupby('Дата').first().reset_index()
```

```
data_daily = data_daily.drop(columns=['Местное время в Самаре'])
```

```
data_daily
```

	Дата	T	Po	P	Pa	U	DD	Ff	N	WW	Cl	Nh	H	Cm	Ch	VV	Td
0	2023-05-08	6.9	758.5	771.2	1.2	44.0	8	2	7.0	1	4.0	4.0	2.0	0.0	1.0	10.0	-4.5
1	2023-05-09	11.1	755.3	767.8	-0.3	30.0	9	1	2.0	1	4.0	4.0	2.0	0.0	3.0	10.0	-6.1
2	2023-05-10	14.4	751.3	763.6	0.5	23.0	9	1	2.0	1	0.0	6.0	3.0	1.0	1.0	10.0	-6.3
3	2023-05-11	14.8	750.1	762.4	0.2	40.0	7	0	2.0	1	0.0	0.0	0.0	1.0	1.0	10.0	1.1
4	2023-05-12	10.3	747.0	759.4	0.7	67.0	9	1	2.0	0	7.0	7.0	0.0	1.0	1.0	10.0	4.3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
180	2023-11-04	1.9	757.2	770.2	0.5	77.0	4	1	7.0	1	4.0	4.0	2.0	0.0	1.0	10.0	-1.8
181	2023-11-05	4.7	753.9	766.6	-0.6	93.0	4	3	7.0	1	4.0	4.0	2.0	0.0	1.0	10.0	3.7
182	2023-11-06	7.0	747.4	760.0	-1.1	88.0	4	3	6.0	1	4.0	6.0	2.0	1.0	1.0	10.0	5.1
183	2023-11-07	5.8	741.6	754.1	-0.4	97.0	1	1	2.0	3	2.0	3.0	1.0	1.0	1.0	10.0	5.4
184	2023-11-08	7.6	747.1	759.7	0.6	94.0	0	2	0.0	0	0.0	0.0	0.0	0.0	0.0	10.0	6.7

185 rows × 17 columns

Оставляем 10 столбцов-параметров погоды. Я выбрала:

- температура воздуха на высоте 2 метра над поверхностью земли,
- атмосферное давление на уровне станции,
- атмосферное давление, приведенное к среднему уровню моря,
- относительная влажность на высоте 2 метра над поверхностью земли,
- направление ветра на высоте 10-12 метров над земной поверхностью,
- скорость ветра на высоте 10-12 метров над земной поверхностью
- общая облачность,
- количество всех наблюдающихся облаков,
- высота основания самых низких облаков,
- горизонтальная дальность видимости.

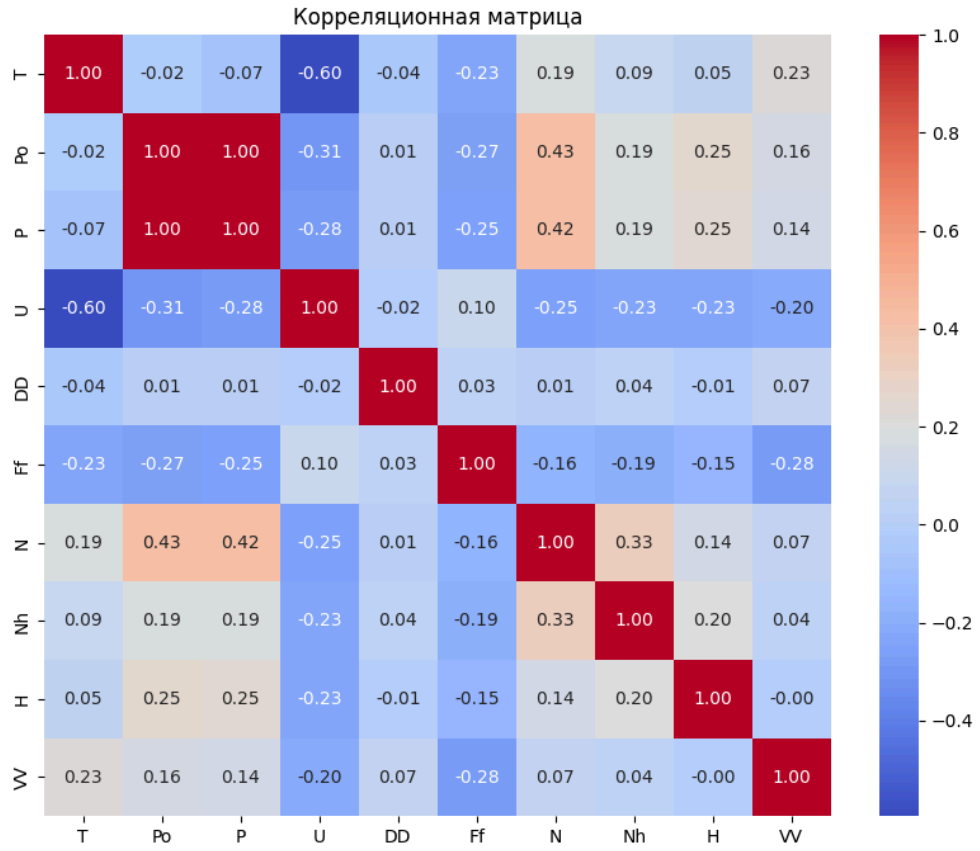
```
selected_columns = ['T', 'Po', 'P', 'U', 'DD', 'Ff', 'N', 'Nh', 'H', 'VV',]
data_selected = data_daily[selected_columns]
data_selected
```

	T	Po	P	U	DD	Ff	N	Nh	H	VV
0	6.9	758.5	771.2	44.0	8	2	7.0	4.0	2.0	10.0
1	11.1	755.3	767.8	30.0	9	1	2.0	4.0	2.0	10.0
2	14.4	751.3	763.6	23.0	9	1	2.0	6.0	3.0	10.0
3	14.8	750.1	762.4	40.0	7	0	2.0	0.0	0.0	10.0
4	10.3	747.0	759.4	67.0	9	1	2.0	7.0	0.0	10.0
...	...	...	...	...	...	...	...	...	...	...
180	1.9	757.2	770.2	77.0	4	1	7.0	4.0	2.0	10.0
181	4.7	753.9	766.6	93.0	4	3	7.0	4.0	2.0	10.0
182	7.0	747.4	760.0	88.0	4	3	6.0	6.0	2.0	10.0
183	5.8	741.6	754.1	97.0	1	1	2.0	3.0	1.0	10.0
184	7.6	747.1	759.7	94.0	0	2	0.0	0.0	0.0	10.0

185 rows × 10 columns

c) Построим корреляционную матрицу

```
# Построение корреляционной матрицы
import seaborn as sns
import matplotlib.pyplot as plt

corr_matrix = data_selected.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Корреляционная матрица')
plt.show()
```

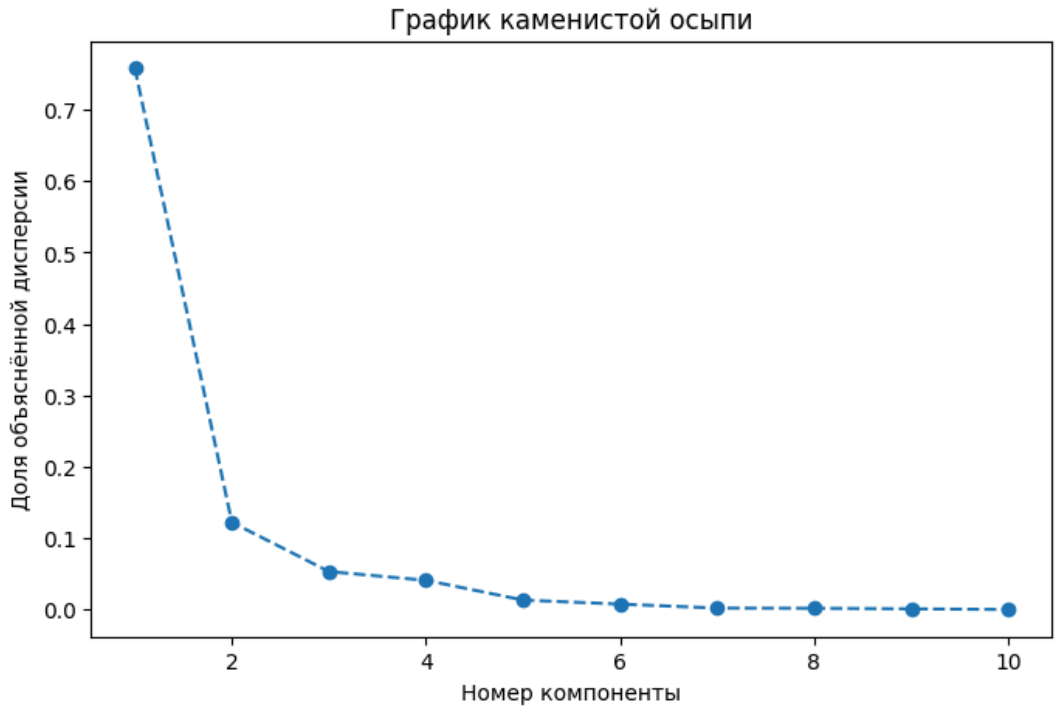


Корреляционная матрица показывает, какие параметры связаны друг с другом. Например:

- Пара U (влажность) и T (температура) имеет сильную корреляцию: -0.60 (когда температура повышается, влажность может уменьшаться).
- Пара P (атмосферное давление, приведенное к среднему уровню моря) и Po (атмосферное давление на уровне станции) имеет сильную корреляцию: 1.00 (логично).

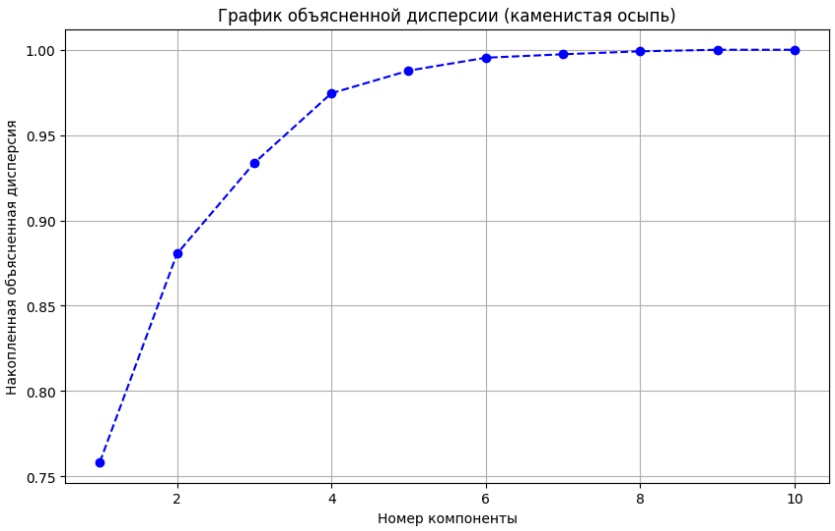
d) Построим график каменистой осыпи (здесь же реализуем алгоритм PCA):

```
# Построение графика каменистой осыпи
from sklearn.decomposition import PCA

pca = PCA()
pca.fit(data_selected.dropna())
explained_variance = pca.explained_variance_ratio_

plt.figure(figsize=(8, 5))
plt.plot(range(1, len(explained_variance) + 1), explained_variance, marker='o', linestyle='--')
plt.title('График каменистой осыпи')
plt.xlabel('Номер компоненты')
plt.ylabel('Доля объяснённой дисперсии')
plt.show()
```



2. б) Объясненная дисперсия:

```python
# Объясненная дисперсия для каждой компоненты
explained_variance = pca.explained_variance_ratio_

# График объясненной дисперсии (график "каменистой осыпи")
plt.figure(figsize=(10, 6))
plt.plot(range(1, len(explained_variance) + 1), explained_variance.cumsum(), marker='o', linestyle='--', color='b')
plt.xlabel('Номер компоненты')
plt.ylabel('Накопленная объясненная дисперсия')
plt.title('График объясненной дисперсии (каменистая осыпь)')
plt.grid()
plt.show()
```



```
print("Объяснённая дисперсия для каждой компоненты:")
for i, var_ratio in enumerate(explained_variance):
   print(f"Компонента {i+1}: {var_ratio:.4f}")
```

```
Объяснённая дисперсия для каждой компоненты:
Компонента 1: 0.7583
Компонента 2: 0.1221
Компонента 3: 0.0532
Компонента 4: 0.0408
Компонента 5: 0.0132
Компонента 6: 0.0076
Компонента 7: 0.0020
Компонента 8: 0.0017
Компонента 9: 0.0009
Компонента 10: 0.0000
```

c)

```
# Вычислить процент объяснённой дисперсии для первых n компонент
n = 4  # Например, 4 компоненты
explained_variance_n = sum(explained_variance[:n]) * 100
print(f"Процент объяснённой дисперсии для первых {n} компонент: {explained_variance_n:.2f}%")
```

```
Процент объяснённой дисперсии для первых 4 компонент: 97.45%
```

d)

```
# Построить проекцию данных на новое пространство
# Создаём DataFrame для новых данных
data_projected = pd.DataFrame(data_pca[:, :n], columns=[f"PC{i+1}" for i in range(n)])

# Визуализируем проекцию в пространстве двух главных компонент
plt.figure(figsize=(8, 6))
plt.scatter(data_projected["PC1"], data_projected["PC2"], alpha=0.7, edgecolor='k')
plt.title('Визуализация данных в пространстве первых двух главных компонент')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.grid()
plt.show()
```

3. а) Предварительно нормализовав данные, воспользуемся алгоритмом t-SNE:

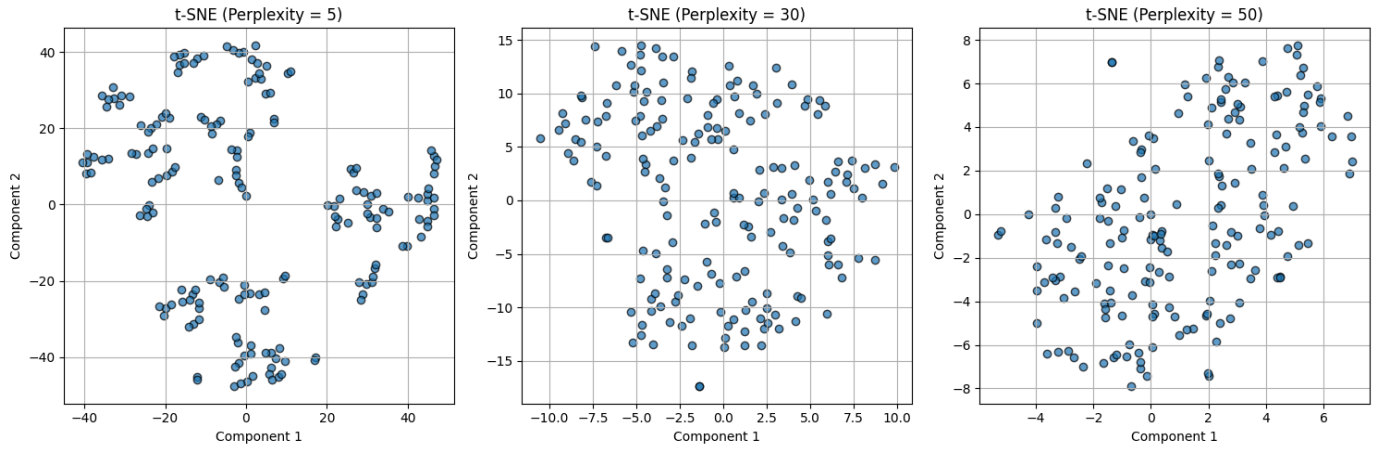
```
# Применяем t-SNE с базовыми параметрами
tsne = TSNE(n_components=2, perplexity=30, random_state=42)
data_tsne = tsne.fit_transform(data_normalized)
```

б и с) Теперь попробуем разные значения перплексии (5, 30 и 50) и визуализируем результаты:

```
# 3b. Применить t-SNE с разными значениями перплексии
perplexities = [5, 30, 50]
tsne_results = {}

for perplexity in perplexities:
    tsne = TSNE(n_components=2, perplexity=perplexity, random_state=42)
    tsne_results[perplexity] = tsne.fit_transform(data_normalized)
```

```
# 3c. Визуализировать результаты
plt.figure(figsize=(15, 5))
for i, perplexity in enumerate(perplexities, 1):
    plt.subplot(1, len(perplexities), i)
    tsne_data = tsne_results[perplexity]
    plt.scatter(tsne_data[:, 0], tsne_data[:, 1], alpha=0.7, edgecolor='k')
    plt.title(f"t-SNE (Perplexity = {perplexity})")
    plt.xlabel("Component 1")
    plt.ylabel("Component 2")
    plt.grid()

plt.tight_layout()
plt.show()
```

Полученные результаты:



Анализ влияния параметра перплексии на результат.

Параметр перплексии в t-SNE — это гиперпараметр, который определяет, насколько “локальной” или “глобальной” будет структура, сохраняемая алгоритмом. Он влияет на то, как точки распределяются в пространстве.

При перплексии = 5 появляются чёткие локальные кластеры. Алгоритм сильно фокусируется на соседях каждой точки, что может привести к разделению данных на мелкие группы. Глобальная структура данных частично искажена — некоторые кластеры кажутся слишком далеко друг от друга.

При перплексии = 30 сохраняется баланс между локальной и глобальной структурой. Кластеры сохраняют более естественное распределение, точки в каждом кластере ближе друг к другу. Видно больше связей между точками в кластерах, но глобальные расстояния между кластерами всё ещё выражены.

При перплексии = 50 глобальная структура данных выражена сильнее, кластеры ближе друг к другу. Локальная структура менее детализирована по сравнению с предыдущими графиками. Возможна потеря четкости внутри кластеров — некоторые точки, принадлежащие к разным кластерам, могут оказаться ближе друг к другу.

Таким образом, средняя перплексия (30) показала наиболее сбалансированный результат. Если анализируется локальная структура, то перплексия, равная 5, даёт более чёткую локализацию точек. Перплексия 50 подходит, если важны глобальные связи между кластерами.

4. Выводы

PCA: График показывает распределение данных в пространстве первых двух главных компонент. Данные выглядят равномерно распределенными, однако четкого разделения на кластеры не наблюдается.

t-SNE: Графики t-SNE с различной перплексией (5, 30, 50) показывают более явное разделение данных на кластеры, хотя глобальная структура искажена.

Перечислим преимущества и недостатки методов.

PCA преимущества:

1. Сохранение глобальной структуры.

2. Простота интерпретации.
3. Быстродействие.
4. Не требует настройки гиперпараметров.

PCA недостатки:

1. PCA работает только с линейными зависимостями, поэтому сложно захватывать сложные, нелинейные структуры.
2. Частичная потеря информации.
3. Отсутствие кластеризации.

t-SNE преимущества:

1. t-SNE отлично сохраняет локальные зависимости между точками, выявляя кластеры в данных.
2. Подходит для нелинейных зависимостей.
3. Возможность настройки перплексии позволяет варьировать фокус на локальных или глобальных структурах.

t-SNE недостатки:

1. Алгоритм плохо сохраняет глобальные расстояния между кластерами, что может искажать общую картину.
2. t-SNE медленно работает на больших данных.
3. Результат сильно зависит от выбора гиперпараметра (перплексии), что требует экспериментов для оптимального значения.

Рекомендации по использованию

PCA:

- Использовать для задач, где важна глобальная структура данных.
- Подходит для подготовки данных перед моделированием.
- Оптимален, если данные имеют линейные зависимости.

t-SNE:

- Рекомендуется для визуализации данных с нелинейными зависимостями.
- Полезен для обнаружения локальных кластеров или аномалий.

- Не подходит для задач, где требуется сохранение глобальной структуры или обработка больших объемов данных.