САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Отчет по лабораторной работе №5
по курсу «Современные инструменты анализа данных»
Тема: **Уменьшение размерности данных. Сравнение методов**

Выполнили:
-

Проверила:
-

Санкт-Петербург
2023 г.

Задание 1. Загрузка данных

С ресурса rp5.ru загрузили данные о метеоусловиях из Вены (аэропорт Швехате) за период 08.04.2023 – 08.11.2023 с промежутком в каждые 30 минут

Обозначение и пояснение некоторых признаков:

- T - темп воздуха на высоте 2 метра
- P0 - атмосферное давление на высоте станции
- P - атмосферное давление на уровне моря
- U - относительная влажность на высоте 2 метра в %
- DD - направление ветра
- Ff - скорость ветра на высоте 10 метров
- ff10 - макс. значение порывов ветра
- VV - горизонтальная дальность видимости в км

Задание 2. Предобработка и исследование данных

Были проделаны следующие шаги:

- удалены признаки-дефекты (при чтении csv файла считались пустые колонки без данных)
- удалены колонки с очень маленьким количеством объектов относительно всей выборки (100-200 значений на 10275 значений всей выборки)
- удалены строки с оставшейся html-разметкой
- переименованы некоторые кириллические колонки (для удобства манипуляций с данными)
- время закодировано в формате тригонометрических функций – по две колонки для месяцев и дат со значением cos и sin для сохранения информации о дистанции и заклиненности дней и месяцев относительно друг друга
- данные агрегированы до периодичности раз в сутки (для каждого дня оставлена одна строка)
- колонка `VV` переведена во float значения (из строковых данных описания выбраны конкретные числа)
- удалена колонка `P`, т.к. она имеет 100% корреляцию с колонкой `P0` (обе колонки демонстрируют уровень влажности в разных условиях, но одна из них линейно выражается через другую)
- Категориальные признаки закодированы через OneHotEncode с удалением первой колонки, т.к. бинарные значения k-1 колонок определяют k категорий

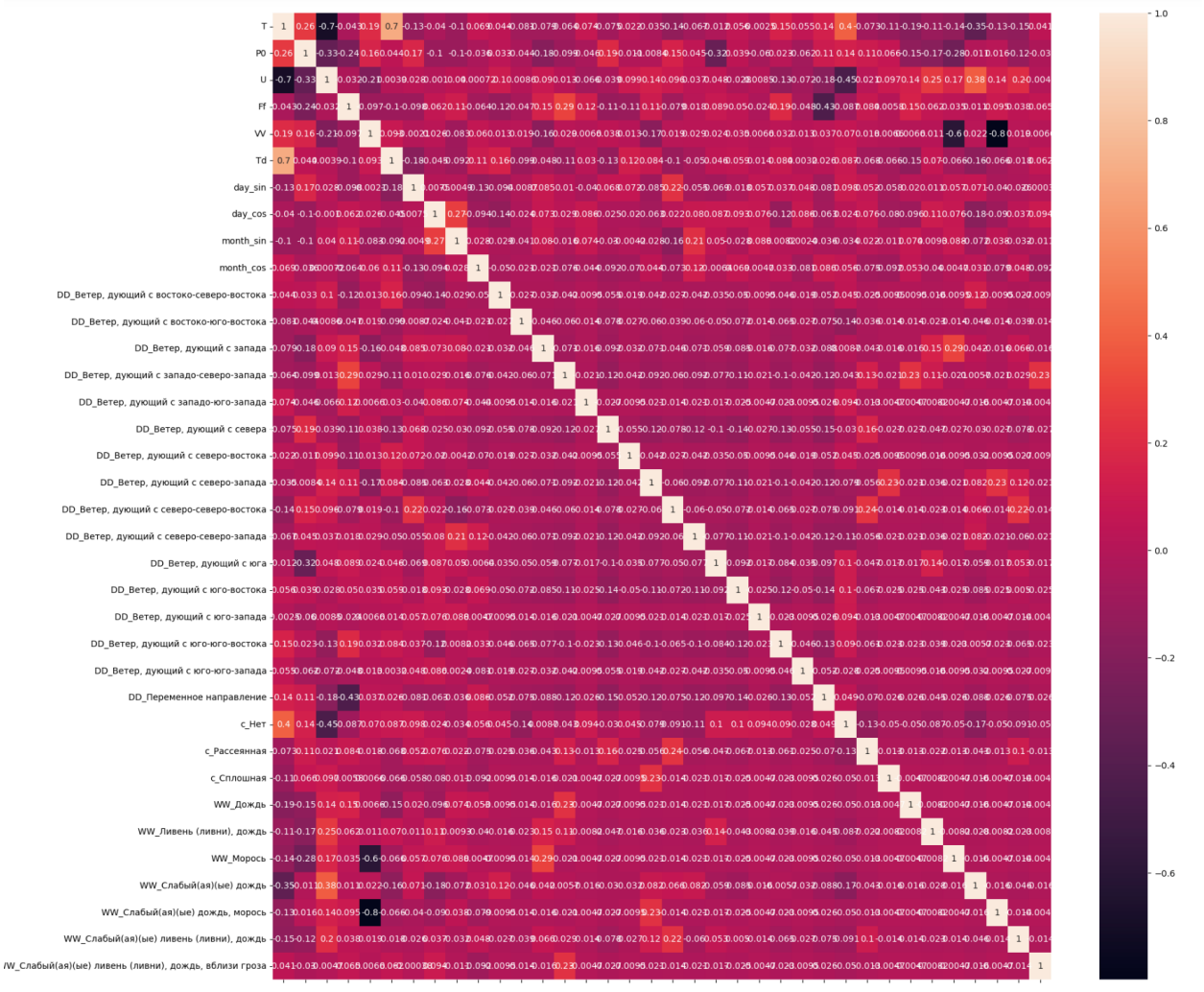
	T	P0	U	Ff	VV	Td	day_sin	day_cos	month_sin	month_cos	...	c_Нет	c_Рассеянная	c_Сплошная	WW_Дождь	WW_Ливень (ливни), дождь	WW_
43	13.0	745.5	63	2	10.0	6.0	0.994522	0.104528	-5.000000e-01	-8.660254e-01	...	True	False	False	False	False	
90	17.0	741.5	39	7	10.0	3.0	0.951057	0.309017	1.224647e-16	-1.000000e+00	...	True	False	False	False	False	
138	13.0	724.4	67	2	10.0	7.0	0.866025	0.500000	5.000000e-01	-8.660254e-01	...	False	False	False	False	False	
186	11.0	731.5	72	6	10.0	6.0	0.743145	0.669131	8.660254e-01	-5.000000e-01	...	False	False	False	False	False	
234	8.0	724.1	93	6	7.0	7.0	0.587785	0.809017	1.000000e+00	6.123234e-17	...	False	False	False	False	False	

Пример обработанных данных (осталось 214 строк и получилось 36 числовых признаков)

```
T       39.961410
P0      27.095642
U      243.060792
Ff       5.388531
VV       0.116296
```

Дисперсия оставшихся числовых признаков (не закодированных)

Построим корреляционную матрицу для признаков. Если значение корреляции для двух признаков стремится к 1, то они пропорционально зависят друг от друга, если значение стремится к -1, то признаки обратно пропорциональны, а если значение стремится к 0, то между признаками отсутствует какая-либо линейная зависимость



Матрица корреляции

На тепловой карте матрицы корреляции можно заметить, что `U` (отн.влажность) достаточно сильно коррелирует с температурой `T` (значение корреляции -0.7), т.е. влажность и температура почти обратно пропорционально друг другу изменяются. Также морось и слабый дождь имеют некоторую обратно пропорциональную зависимость с `VV` (дальность видимости в км.), т.к. их значения корреляции -0.6 и -0.8 соответственно.

Чтобы построить график каменистой осыпи, сначала применим PCA для всех признаков, чтобы посмотреть, какую часть всей информации (объясненная дисперсия) отражает разное количество признаков.

Перед применением общего PCA, т.е. n_components=<кол-во всех признаков>, нужно обязательно нормализовать данные, т.к. PCA алгоритм прибегает к линейным преобразованиям в виде вычисления собственных векторов и проецирования, для чего необходимо правильно учитывать масштабы векторов признаков в n-мерном пространстве. Для нормализации применим StandardScaler из sklearn, чтобы свести все к среднему значению 0 и дисперсии 1.

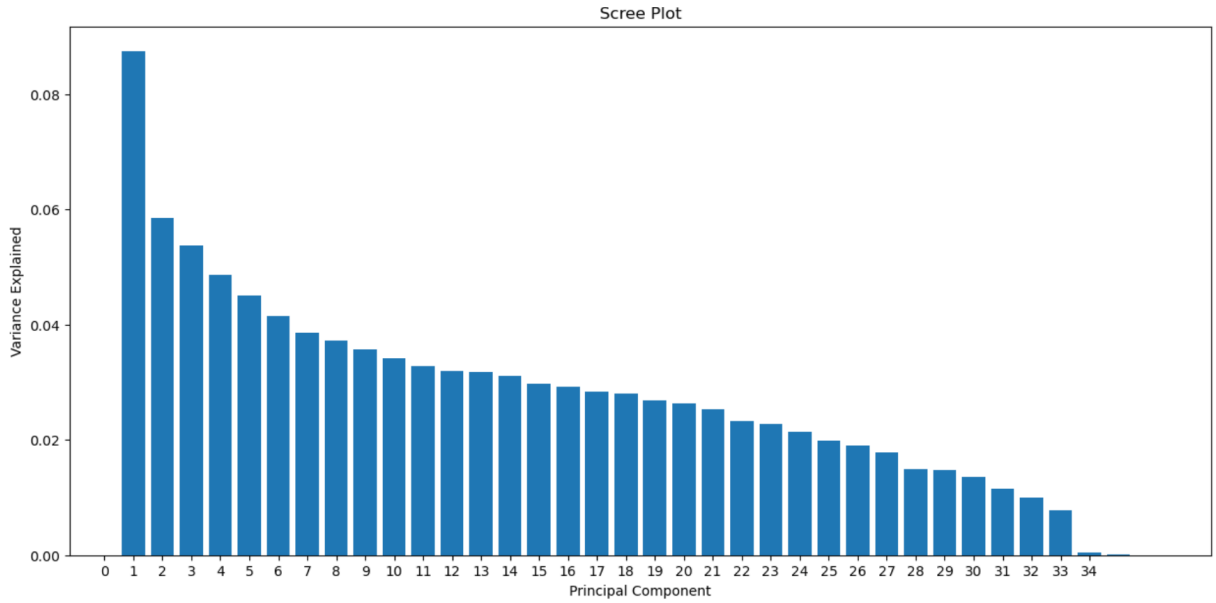


График каменистой осыпи в виде столбчатой диаграммы

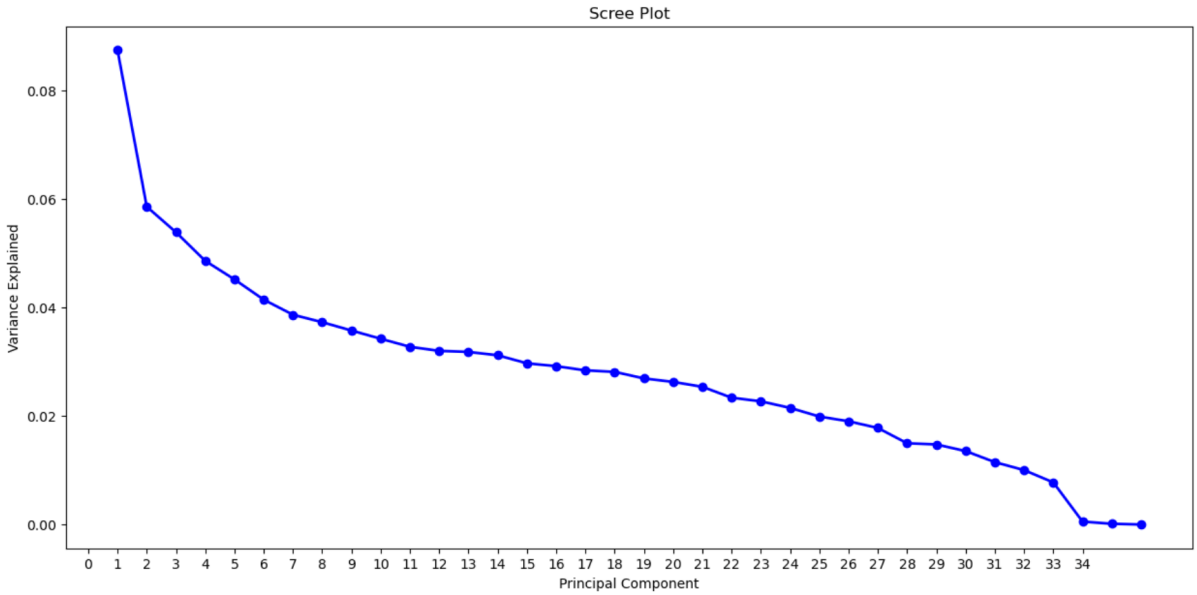


График каменистой осыпи

Задание 3. Алгоритм PCA

Для алгоритма PCA мы вычисляем корреляционную матрицу всех признаков, для которых находим собственные векторы, т.е. те векторы, которые показывают основные оси направления распределений наших признаков в n-мерном пространстве. По графику каменистой осыпи для всех признаков из предыдущего пункта мы видим сколько признаков демонстрирует некоторый уровень объясненной дисперсии (их сумма равна 1, т.к. все признаки объясняют все данные, и мы не теряем информацию).

Несмотря на то, что есть некоторые сильные изгибы на графике, примерно только 24 признака объясняет только 0.8700595054019403 дисперсии всех данных, т.е. приблизительно 87%, дальше это число будет равномерно убывать, поэтому попробуем остановиться на сокращении размерности наших признаков с 32 до 24, чтобы не терять значительную часть информации (опять же нормализуем данные перед алгоритмом).

```
# PCA dimensions' reduction
pca_fit = PCA(n_components = n_components).fit_transform(scaled_matrix)
pca_fit.shape
```

```
(214, 24)
```

Код реализации, где n_components = 24 задано заранее выше

Задание 4. Алгоритм t-SNE

Этот алгоритм уже не является линейным преобразованием признаков и подходит тогда, когда мы пытаемся сократить размерность данных, выявив какие-то нелинейные зависимости.

Для этого сначала вычисляем матрицу попарных расстояний (евклидовых) между объектами, а потом через экспоненцирование этих расстояний для нормализации задаем условные вероятности для каждой пары признаков (в числителе просто экспонента в степени расстояния, а в знаменателе сумма экспонент в степенях расстояний). Алгоритм

является обучаемым, т.к. мы должны минимизировать некую функцию потерь (расстояние Кульбака-Лейблера) с помощью градиентного спуска, чтобы подобрать наилучшее распределение новых данных (в вычисленных условных вероятностях также присутствует обучаемый параметр). Также есть ряд гиперпараметров как шаг градиентного спуска, перплексивность, выбор модификации алгоритма и т.д.

Методом экспериментов было решено установить перплексию равную 3, а также случайную инициализацию начальных векторов (поэтому зафиксирован random_state) вместо изначальной инициализации через тот же PCA.

```
# t-sne dimensions' reduction
tsne_fit = TSNE(n_components=n_components, learning_rate='auto', init='random',
                perplexity=3, method='exact', random_state=1).fit_transform(scaled_matrix)
tsne_fit.shape
```

```
(214, 24)
```

Код реализации, где n_components = 24 задано заранее выше

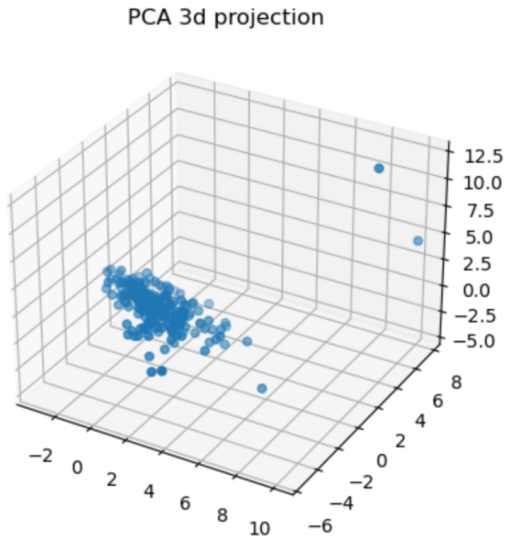
Задание 5. Сравнение алгоритмов

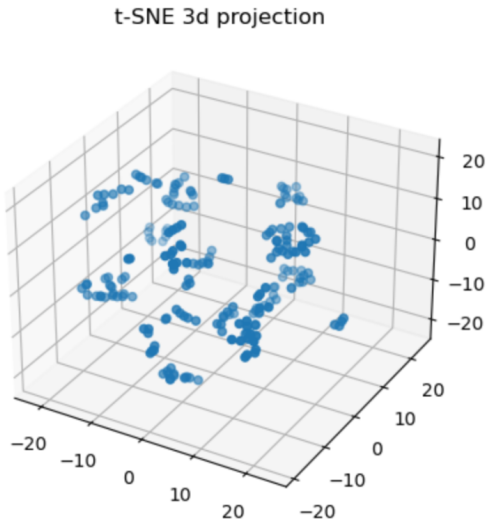
Для начала посмотрим на новые значения дисперсий каждого признака и их средних для двух алгоритмов.

	mean value for all features
pca var	1.311216e+00
t-sne var	1.252600e+01
pca mean	-1.772552e-18
t-sne mean	7.634617e-03

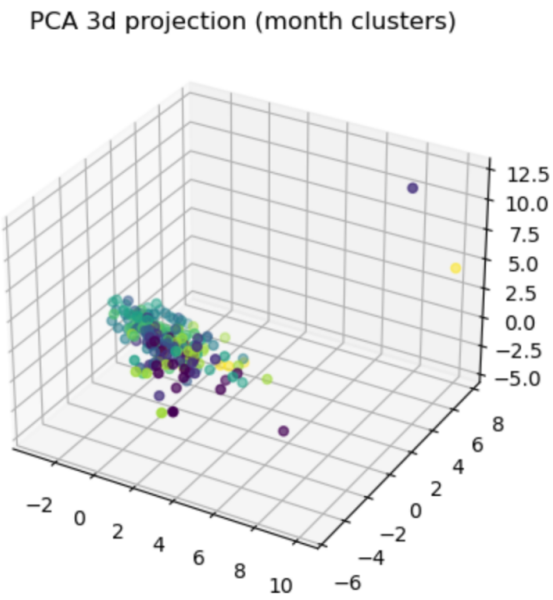
	pca var	t-sne var	pca mean	t-sne mean
0	3.161487	2.571119	4.980440e-17	-0.080677
1	2.117659	15.716062	-3.320293e-17	0.003328
2	1.947363	4.198503	-1.660147e-17	0.087766
3	1.756744	14.868125	4.565403e-17	0.201246
4	1.633183	1.996165	4.980440e-17	0.048937
5	1.498260	12.024145	-3.320293e-17	0.058849
6	1.397860	5.221970	2.490220e-17	-0.087789
7	1.348146	6.170745	-8.300733e-18	0.111384
8	1.292639	6.593685	-4.772921e-17	0.049565
9	1.237787	20.185541	-8.300733e-18	0.066018
10	1.184202	30.495968	-8.300733e-18	0.367099
11	1.157126	15.628245	1.556387e-17	-0.293327
12	1.150137	21.283491	-8.300733e-17	0.147530
13	1.127745	8.601070	-3.320293e-17	0.038570
14	1.074305	29.470411	1.037592e-17	0.203846
15	1.055233	4.145868	-2.490220e-17	0.137162
16	1.027198	5.426380	-2.075183e-17	0.066351
17	1.017359	55.824921	-2.490220e-17	-0.690643
18	0.973426	15.713128	6.225550e-17	0.042124
19	0.950412	6.314998	2.905257e-17	-0.192110
20	0.917699	6.431191	0.000000e+00	-0.109874
21	0.845013	3.607717	-4.150366e-18	0.012548
22	0.821011	3.192011	1.660147e-17	-0.069279
23	0.777202	4.942512	0.000000e+00	0.064605

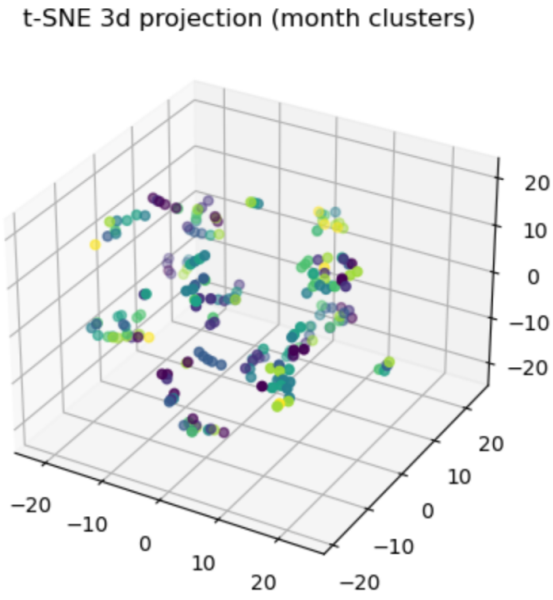
Теперь попробуем еще раз уменьшить все данные уже до 3 измерений каждым алгоритм и посмотреть, как они располагаются в пространстве.





Добавим классы в виде каждого месяца, т.е. на каждый месяц должно приходиться от 30 до 31 точек данных.





Несмотря на то, что в обоих случаях кластеры по месяцам разбросаны достаточно случайно и не вместе (возможно изначальная предобработка категориальных данных или таргет признак в виде месяца выбраны не совсем верно из-за очень сложных зависимостей между признаками), все равно наблюдается разница. В первом случае все объекты больше «кучкуются», т.к. формально на выбранные три оси проецируются все остальные, группируясь в одном месте, также по осям видно. Во втором же случае алгоритм более сложный, разводящий объекты друг от друга, точнее пытающийся как-то сохранить некоторую меру, некоторое расстояние между объектами из старого пространства в уже новом. Также в обоих случаях по осям видно, что все данные нормированы.