



Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУК «Информатика и управление»

КАФЕДРА ИУК5 «Информатика и вычислительная техника»

Лабораторная работа №2

«Метод главных компонент и кластеризация»

ДИСЦИПЛИНА: «Проектирование программного обеспечения»

Выполнил: студент гр. ИУК4-11М _____ (Сафронов Н.С.)
(подпись) (Ф.И.О.)

Проверил: _____ (Потапов А.Е.)
(подпись) (Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:

- Оценка:

Калуга, 2024

Цель работы: формирование практических навыков создания интеллектуальных систем с обучением без учителя (unsupervised learning).

Задачи: подготовить данные для эксперимента. Выполнить анализ при помощи метода главных компонент. Выполнить кластеризацию при помощи методов K-Means и агломеративного. Проанализировать эффективность вариации гиперпараметров моделей и применения композиции методов.

Результаты выполнения работы

Отмасштабируем выборку с помощью StandardScaler с параметрами по умолчанию и понизим размерность с помощью PCA.

```
scaler = StandardScaler()  
X_scaled = scaler.fit_transform(X)
```

```
pca = PCA(n_components=0.9, random_state=RANDOM_STATE).fit(X_scaled)  
X_pca = pca.transform(X_scaled)
```

Рисунок 1 – Масштабирование и понижение размерности выборки

Вопрос 1:

Какое минимальное число главных компонент нужно выделить, чтобы объяснить 90% дисперсии исходных (отмасштабированных) данных?

```
X_pca.shape
```

```
(10299, 65)
```

Рисунок 2 – Форма результата понижения размерности методом главных компонент

Таким образом, в результате понижения размерности было выделено 65 главных компонент.

Ответ: 65.

Вопрос 2:

Сколько процентов дисперсии приходится на первую главную компоненту? Округлите до целых процентов.

```
round(float(pca.explained_variance_ratio_[0] * 100))
```

51

Рисунок 3 – Результат получения дисперсии первой главной компоненты

Ответ: 51.

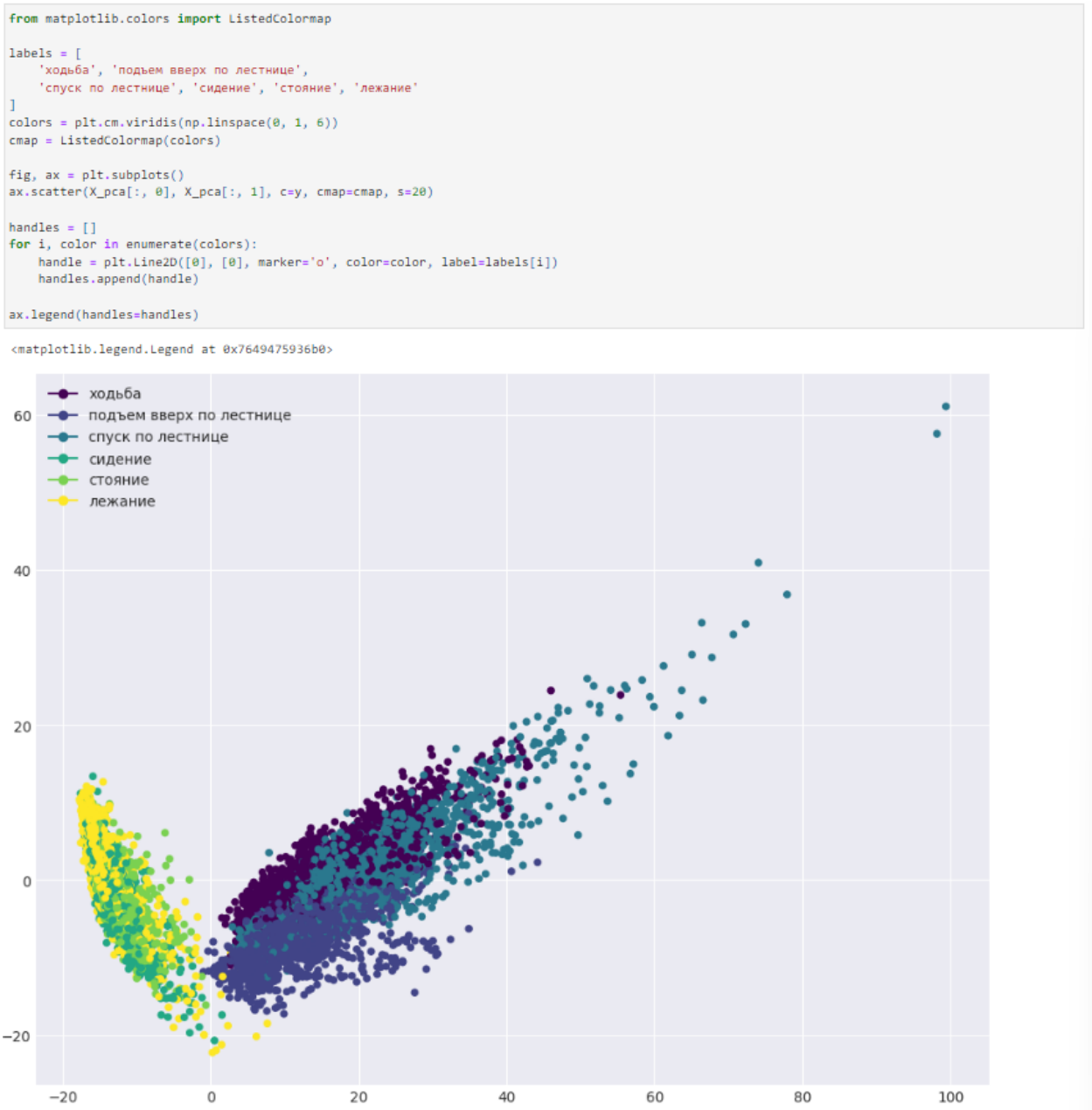


Рисунок 4 – Визуализация данных в проекции на первые две главные компоненты

Вопрос 3:

Если все получилось правильно, Вы увидите сколько-то кластеров, почти идеально отделенных друг от друга. Какие виды активности входят в эти кластеры?

Ответ: 2 кластера: (ходьба, подъем вверх по лестнице, спуск по лестнице) и (сидение, стояние, лежание).

```
kmeans = KMeans(n_clusters=n_classes, n_init=100, random_state=RANDOM_STATE)
kmeans.fit(X_pca)
cluster_labels = kmeans.labels_
```

Рисунок 5 – Кластеризация методом KMeans



Рисунок 6 – Визуализация кластеризации методом KMeans

```

tab = pd.crosstab(y, cluster_labels, margins=True)
tab.index = ['ходьба', 'подъем вверх по лестнице',
             'спуск по лестнице', 'сидение', 'стояние', 'лежание', 'все']
tab.columns = ['cluster' + str(i + 1) for i in range(6)] + ['все']
tab

```

| | cluster1 | cluster2 | cluster3 | cluster4 | cluster5 | cluster6 | все |
|--------------------------|----------|----------|----------|----------|----------|----------|-------|
| ходьба | 0 | 903 | 741 | 78 | 0 | 0 | 1722 |
| подъем вверх по лестнице | 0 | 1241 | 296 | 5 | 2 | 0 | 1544 |
| спуск по лестнице | 0 | 320 | 890 | 196 | 0 | 0 | 1406 |
| сидение | 1235 | 1 | 0 | 0 | 450 | 91 | 1777 |
| стояние | 1344 | 0 | 0 | 0 | 562 | 0 | 1906 |
| лежание | 52 | 5 | 0 | 0 | 329 | 1558 | 1944 |
| все | 2631 | 2470 | 1927 | 279 | 1343 | 1649 | 10299 |

Рисунок 7 – Составы полученных кластеров

Вопрос 4:

Какой вид активности отделился от остальных лучше всего в терминах простой метрики, описанной выше?

```

pd.Series(
    tab.iloc[:,-1].max(axis=1).values / tab.iloc[:,-1].values,
    index=tab.index[:,-1],
)

```

| | |
|--------------------------|----------|
| ходьба | 0.524390 |
| подъем вверх по лестнице | 0.803756 |
| спуск по лестнице | 0.633001 |
| сидение | 0.694992 |
| стояние | 0.705142 |
| лежание | 0.801440 |

dtype: float64

Рисунок 8 – Максимальная доля объектов в классе, отнесенных к кластеру

Ответ: перечисленные варианты не подходят.

Воспользуемся методом локтя:

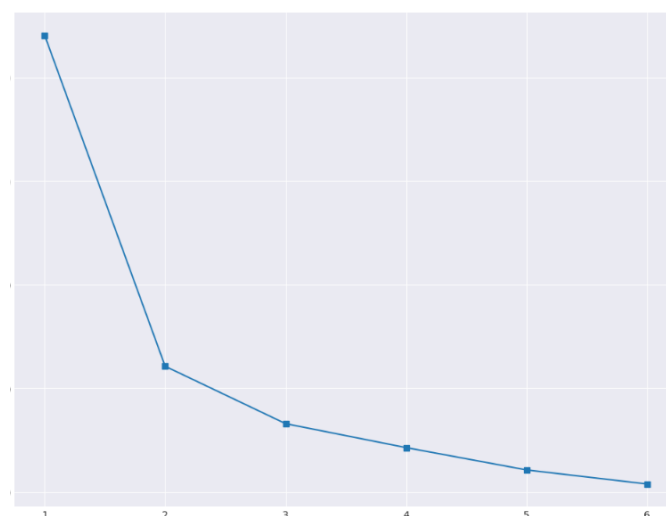


Рисунок 9 – Величина инерции для различного количества кластеров

```

d = {}
for k in range(2, 6):
    i = k - 1
    d[k] = (inertia[i] - inertia[i + 1]) / (inertia[i - 1] - inertia[i])
d

{2: np.float64(0.1734475356009401),
 3: np.float64(0.41688555755864765),
 4: np.float64(0.9332198909748659),
 5: np.float64(0.6297014287707848)}

```

Рисунок 10 – Значение метрики разности для различных значений количества кластеров

Вопрос 5:

Какое количество кластеров оптимально выбрать, согласно методу локтя?

Выбираем наименьшее значение метрики разности.

Ответ: 2.

```

metrics.adjusted_rand_score(y, cluster_labels)

0.4198070012602345

metrics.adjusted_rand_score(y, ag.labels_)

0.49362763373004886

```

Рисунок 11 – Значение ARI для KMeans и агломеративной кластеризации

Вопрос 6:

Отметьте все верные утверждения.

KMeans имеет меньшую метрику, следовательно справился хуже.

ARI опирается на пары объектов и оценивает, находятся ли они в одном кластере или в разных кластерах в двух сравниваемых разбиениях (истинном и предсказанном).

Если разбиение произведено случайным образом, вероятность того, что одна и та же пара объектов попадет в один и тот же кластер в двух разных разбиениях, очень низка. Поэтому вклад этих пар в итоговый индекс также будет низким, что и приводит к значению ARI, близкому к нулю.

Ответ:

- Согласно ARI, KMeans справился с кластеризацией хуже, чем Agglomerative Clustering

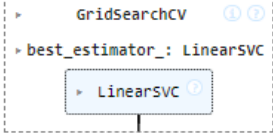
- Для ARI не имеет значения какие именно метки присвоены кластерам, имеет значение только разбиение объектов на кластеры
- В случае случайного разбиения на кластеры ARI будет близок к нулю

Воспользуемся методом опорных векторов:

```
# Ваш код здесь
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

svc = LinearSVC(random_state=RANDOM_STATE)
svc_params = {'C': [0.001, 0.01, 0.1, 1, 10]}

# Ваш код здесь
best_svc = GridSearchCV(svc, svc_params, n_jobs=4, cv=3, verbose=1)
best_svc.fit(X_train_scaled, y_train)
```



```
best_svc.best_params_, best_svc.best_score_

({'C': 0.1}, np.float64(0.9379785010699506))
```

Рисунок 12 – Метод опорных векторов

Вопрос 7

Какое значение гиперпараметра C было выбрано лучшим по итогам кросс-валидации?

Ответ: 0.1.

Определим вид активности для тестовой выборки и сравним с заданными значениями.

```
y_predicted = best_svc.predict(X_test_scaled)

tab = pd.crosstab(y_test, y_predicted, margins=True)
tab.index = ['ходьба', 'подъем вверх по лестнице', 'спуск по лестнице',
             'сидение', 'стояние', 'лежание', 'все']
tab.columns = tab.index
tab
```

| | ходьба | подъем вверх по лестнице | спуск по лестнице | сидение | стояние | лежание | все |
|--------------------------|--------|--------------------------|-------------------|---------|---------|---------|------|
| ходьба | 494 | 2 | 0 | 0 | 0 | 0 | 496 |
| подъем вверх по лестнице | 12 | 459 | 0 | 0 | 0 | 0 | 471 |
| спуск по лестнице | 2 | 4 | 413 | 1 | 0 | 0 | 420 |
| сидение | 0 | 4 | 0 | 426 | 61 | 0 | 491 |
| стояние | 0 | 0 | 0 | 15 | 517 | 0 | 532 |
| лежание | 0 | 0 | 0 | 0 | 11 | 526 | 537 |
| все | 508 | 469 | 413 | 442 | 589 | 526 | 2947 |

Рисунок 13 – Сравнение результата метода опорных векторов и исходных данных (матрица неточностей)

Вопрос 8:

Какой вид активности SVM определяет хуже всего в терминах точности? Полноты?

Точность равняется отношению соответствующего диагонального элемента матрицы неточностей и суммы всей строки класса. Полнота – отношению диагонального элемента матрицы и суммы всего столбца класса

Таким образом, наименьшее значения точности у сидения:

$$Precision = \frac{426}{491} = 0.86$$

Наименьшее значения полноты у стояния:

$$Recall = \frac{517}{589} = 0.88$$

Ответ: по точности – у сидения, по полноте – у стояния.

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

pca = PCA(n_components=0.9, random_state=RANDOM_STATE)
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)

svc = LinearSVC(random_state=RANDOM_STATE)
svc_params = {"C": [0.001, 0.01, 0.1, 1, 10]}

best_svc_pca = GridSearchCV(svc, svc_params, n_jobs=4, cv=3, verbose=1)
best_svc_pca.fit(X_train_pca, y_train)
```

Fitting 3 folds for each of 5 candidates, totalling 15 fits

```
> GridSearchCV ① ②
> best_estimator_: LinearSVC
  > LinearSVC ③
```

```
best_svc_pca.best_params_, best_svc_pca.best_score_

({'C': 0.1}, np.float64(0.8983982658750974))

round(100 * (best_svc_pca.best_score_ - best_svc.best_score_))
```

-4

Рисунок 14 – Обучение PCA на обучающей выборке и применение к основной, настройка гиперпараметра C

Вопрос 9:

Какова разность между лучшим качеством (долей верных ответов) на кросс-валидации в случае всех 561 исходных признаков и во втором случае, когда применялся метод главных компонент? Округлите до целых процентов.

Ответ: 4.

Вопрос 10:

Выберите все верные утверждения.

```
best_svc.score(X_test_scaled, y_test)
0.9619952494061758

best_svc_pca.score(X_test_pca, y_test)
0.9192399049881235
```

Рисунок 15 – Значение точности для PCA и простой модели

Как мы можем заметить, значения отличаются менее чем на 10%.

PCA используется для визуализации многомерных данных, так как сводит их к двум или трем измерениям, что удобно для отображения на графиках. PCA основан на вычислении собственных значений и собственных векторов ковариационной матрицы данных, что делает его линейным методом. Это вычислительно менее затратный процесс по сравнению с t-SNE.

PCA строит новые признаки (главные компоненты) как линейные комбинации исходных признаков. Каждая главная компонента — это взвешенная сумма исходных признаков, где веса (коэффициенты) определяются как направления максимальной вариации в данных. Эти линейные комбинации могут не иметь явного физического смысла или легко интерпретируемого значения. Например, если в исходных данных есть признаки, такие как «вес» и «рост», PCA может создать компоненту, которая представляет собой нечто вроде $0.5 \times \text{"вес"} + 0.7 \times \text{"рост"}$, что не всегда легко объяснить в реальном мире.

Ответ:

- PCA можно использовать для визуализации данных, однако для этой задачи есть и лучше подходящие методы, например, tSNE. Зато PCA имеет меньшую вычислительную сложность
- PCA строит линейные комбинации исходных признаков, и в некоторых задачах они могут плохо интерпретироваться человеком

Вывод: в ходе выполнения лабораторной работы были получены практические навыки создания интеллектуальных систем с обучением без учителя (unsupervised learning).