**АНАЛИЗ ГЛАВНЫХ КОМПОНЕНТОВ С ПОМОЩЬЮ PYTHON**

**Анализ основных компонентов** — это, в основном, статистическая процедура для преобразования набора наблюдений за возможными коррелированными переменными в набор значений линейно некоррелированных переменных.
Каждый из основных компонентов выбирается таким образом, чтобы он описывал большую часть все еще доступной дисперсии, и все эти основные компоненты ортогональны друг другу. Во всех основных компонентах первый главный компонент имеет максимальную дисперсию.

РЕКЛАМА

**Использование PCA:**

* Он используется для поиска взаимосвязи между переменными в данных.
* Он используется для интерпретации и визуализации данных.
* Поскольку количество переменных уменьшается, это упрощает дальнейший анализ.
* Он часто используется для визуализации генетической дистанции и взаимосвязи между популяциями.

Они в основном выполняются на квадратной симметричной матрице. Это может быть чистая сумма квадратов и матрицы перекрестных произведений или матрицы ковариации или матрицы корреляции. Корреляционная матрица используется, если индивидуальная дисперсия сильно отличается.

**Цели СПС:**

* Это в основном независимая процедура, в которой она сокращает пространство атрибутов от большого количества переменных до меньшего числа факторов.
* PCA — это, по сути, процесс сокращения измерений, но нет гарантии, что измерение можно интерпретировать.
* Основная задача в этом PCA состоит в том, чтобы выбрать подмножество переменных из большего набора, на основе которого исходные переменные имеют наибольшую корреляцию с основной суммой.

**Метод главной оси:** PCA в основном ищет линейную комбинацию переменных, чтобы мы могли извлечь максимальную дисперсию из переменных. Как только этот процесс завершается, он удаляет его и ищет другую линейную комбинацию, которая дает объяснение максимальной доли оставшейся дисперсии, которая в основном приводит к ортогональным факторам. В этом методе мы анализируем общую дисперсию.

**Собственный** вектор: это ненулевой вектор, который остается параллельным после умножения матрицы. Предположим, что x является собственным вектором размерности r матрицы M с размерностью r \* r, если Mx и x параллельны. Затем нам нужно решить Mx = Ax, где x и A неизвестны, чтобы получить собственный вектор и собственные значения.
В разделе «Собственные векторы» можно сказать, что главные компоненты показывают как общую, так и уникальную дисперсию переменной. По сути, это ориентированный на дисперсию подход, направленный на воспроизведение общей дисперсии и корреляции со всеми компонентами. Основными компонентами являются в основном линейные комбинации исходных переменных, взвешенные по их вкладу для объяснения дисперсии в определенном ортогональном измерении.

**Собственные значения:** это в основном известно как характерные корни. Он в основном измеряет дисперсию по всем переменным, которая учитывается этим фактором. Отношение собственных значений является отношением объяснительной важности факторов по отношению к переменным. Если коэффициент низкий, то он вносит меньший вклад в объяснение переменных. Проще говоря, он измеряет количество отклонений в общей заданной базе данных, учитываемой фактором. Мы можем вычислить собственное значение фактора как сумму его квадратичной загрузки фактора для всех переменных.

Теперь давайте разберемся с анализом главных компонентов с помощью Python.

Чтобы получить набор данных, используемый в реализации, нажмите [**здесь**](https://media.geeksforgeeks.org/wp-content/uploads/Wine.csv) .

**Шаг 1:** Импорт библиотек

|  |
| --- |
| # импорт необходимых библиотекimport numpy as npimport matplotlib.pyplot as pltimport pandas as pd |

**Шаг 2:** Импорт набора данных

Импортируйте набор данных и распределяйте набор данных по компонентам X и y для анализа данных.

|  |
| --- |
| # импорт или загрузка набора данныхdataset = pd.read\_csv('wines.csv')  # распределение набора данных по двум компонентам X и YX = dataset.iloc[:, 0:13].valuesy = dataset.iloc[:, 13].values |

**Шаг 3:** Разделение набора данных на тренировочный набор и тестовый набор

|  |
| --- |
| # Разделение X и Y на# Учебный набор и набор для тестированияfrom sklearn.model\_selection import train\_test\_split  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0) |

**Шаг 4:** Масштабирование функций

Выполнение предварительной обработки на обучающем и тестовом наборе, например подгонка к стандартной шкале.

|  |
| --- |
| # выполнение препроцессорной частиfrom sklearn.preprocessing import StandardScalersc = StandardScaler()  X\_train = sc.fit\_transform(X\_train)X\_test = sc.transform(X\_test) |

**Шаг 5:** Применение функции PCA

Применение функции PCA в обучающем и тестовом наборе для анализа.

|  |
| --- |
| # Применение функции PCA на тренировках# и тестовый набор X-компонентаfrom sklearn.decomposition import PCA  pca = PCA(n\_components = 2)  X\_train = pca.fit\_transform(X\_train)X\_test = pca.transform(X\_test)  explained\_variance = pca.explained\_variance\_ratio\_ |

**Шаг 6:** Подгонка логистической регрессии к тренировочному набору

|  |
| --- |
| # Подгонка логистической регрессии к тренировочному наборуfrom sklearn.linear\_model import LogisticRegression    classifier = LogisticRegression(random\_state = 0)classifier.fit(X\_train, y\_train) |



**Шаг 7.** Прогнозирование результата теста

|  |
| --- |
| # Прогнозирование результата теста с использованием# функция прогнозирования в LogisticRegressiony\_pred = classifier.predict(X\_test) |

**Шаг 8:** Создание матрицы путаницы

|  |
| --- |
| # создание путаницы между# тестовый набор Y и прогнозируемое значение.from sklearn.metrics import confusion\_matrix  cm = confusion\_matrix(y\_test, y\_pred) |

**Шаг 9:** Прогнозирование результата тренировочного набора

|  |
| --- |
| # Прогнозирование тренировочного набора# результат по точечной диаграммеfrom matplotlib.colors import ListedColormap  X\_set, y\_set = X\_train, y\_trainX1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1,                     stop = X\_set[:, 0].max() + 1, step = 0.01),                     np.arange(start = X\_set[:, 1].min() - 1,                     stop = X\_set[:, 1].max() + 1, step = 0.01))  plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),             X2.ravel()]).T).reshape(X1.shape), alpha = 0.75,             cmap = ListedColormap(('yellow', 'white', 'aquamarine')))  plt.xlim(X1.min(), X1.max())plt.ylim(X2.min(), X2.max())  for i, j in enumerate(np.unique(y\_set)):    plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],                c = ListedColormap(('red', 'green', 'blue'))(i), label = j)  plt.title('Logistic Regression (Training set)')plt.xlabel('PC1') # для Xlabelplt.ylabel('PC2') # для Ylabelplt.legend() # показать легенду  # показать точечный графикplt.show() |



**Шаг 10:** Визуализация результатов тестового набора

|  |
| --- |
| # Визуализация результатов тестового набора по точечной диаграммеfrom matplotlib.colors import ListedColormap  X\_set, y\_set = X\_test, y\_test  X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1,                     stop = X\_set[:, 0].max() + 1, step = 0.01),                     np.arange(start = X\_set[:, 1].min() - 1,                     stop = X\_set[:, 1].max() + 1, step = 0.01))  plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),             X2.ravel()]).T).reshape(X1.shape), alpha = 0.75,             cmap = ListedColormap(('yellow', 'white', 'aquamarine')))   plt.xlim(X1.min(), X1.max())plt.ylim(X2.min(), X2.max())  for i, j in enumerate(np.unique(y\_set)):    plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],                c = ListedColormap(('red', 'green', 'blue'))(i), label = j)  # название для точечного графикаplt.title('Logistic Regression (Test set)') plt.xlabel('PC1') # для Xlabelplt.ylabel('PC2') # для Ylabelplt.legend()  # показать точечный графикplt.show() |



Рекомендуемые посты:

* [**ML | Анализ основных компонентов (PCA)**](http://espressocode.top/ml-principal-component-analysispca/)