чтобы их можно было запускать вместе в изолированной среде.

3. Запустите docker-compose up, и Compose запустит и запустит все ваше приложение.

Compose-файл предоставляет способ настроить все зависимости службы приложения (базы данных, очереди, кэши, API-интерфейсы веб-службы и т. д.). С помощью инструмента командной строки Compose вы можете создать и запустить один или несколько контейнеров для каждой зависимости с помощью одной команды (docker-compose up).

Важной частью любого процесса непрерывного развертывания или непрерывной интеграции является набор автоматизированных тестов. Для автоматизированного сквозного тестирования требуется среда, в которой можно запускать тесты. Compose предоставляет удобный способ создания и уничтожения изолированных сред тестирования для вашего набора тестов. Определив полную среду в compose-файле, вы можете создавать и уничтожать эти среды всего несколькими командами.

Compose традиционно был сосредоточен на рабочих процессах разработки и тестирования, но с каждым выпуском docker-compose делает успехи в более ориентированных на производство функциях.

Для демонстрации возможностей Docker и docker-compose был реализован проект для развертывания и запуска веб-приложения на языке C# [2] с использованием СУБД PostgeSQL. Проект состоит из двух контейнеров Docker. В одном контейнере происходит компиляция и запуск веб-приложения с использованием dotnet-sdk и сервера ASP.Net. Во втором контейнере разворачивается СУБД. Контейнеры связываются между собой с помощью docker-compose.

Полученные файлы Docker и docker-compose можно использовать для быстрого разворачивания приложения на любом сервере, где установлен Docker, без необходимости ручной установки и настройки веб-сервера и сервера баз данных.

Список использованных источников

1. Merkel D. Docker: lightweight linux containers for consistent development and deployment // Linux journal. 2014. Vol. 2014, № 239. P. 2.
2. Hejlsberg A., Wiltamuth S., Golde P. C# language specification. Addison-Wesley Longman Publishing Co., Inc., 2003.

УДК 004.43:378

СРЕДСТВА АНАЛИЗА ДАННЫХ В PYTHON

Ринейская Р.К., маг., Казаков В.Е., к.т.н., доц., Корниенко А.А., д. ф.-м. н., проф.
Витебский государственный технологический университет,
г. Витебск, Республика Беларусь

Реферат. *В этой статье рассмотрены некоторые инструменты анализа данных в Python и возможности их применения в разработке системы автоматизации кластерного анализа успеваемости учащихся.*

Ключевые слова: Python, кластерный анализ, обработка данных.

Python – один из самых популярных языков программирования, который на сегодняшний день широко используется для разработки программного обеспечения и анализа данных. Рассмотрим основные библиотеки Python для анализа и обработки данных.

1. Pandas – это библиотека программного обеспечения для языка программирования Python, которая используется для работы с данными. Она очень популярна в мире анализа данных благодаря своим функциям, позволяющим обрабатывать данные, очищать данные, агрегировать данные, группировать данные, и многое другое [1].

2. NumPy – это библиотека программного обеспечения для языка программирования Python, которая является базовой для многих других библиотек, таких как Pandas и SciPy. NumPy предоставляет широкий спектр функций для работы с массивами и матрицами. Эта библиотека ускоряет вычисления и используется для численных вычислений [2].

3. Matplotlib – это библиотека для Python, которая помогает создавать графики, диаграммы и другие визуализации данных. Она поддерживает множество различных типов диаграмм и

графиков, включая гистограммы, диаграммы рассеяния, линейные графики и т.д. [3].

4. Seaborn – это библиотека для Python, которая помогает создавать красивые и информативные графики. Она основана на Matplotlib, но включает дополнительные функции, которые помогают создавать более сложные графики и визуализации данных [4].

5. Scikit-learn – это библиотека для Python, которая используется для машинного обучения. Она предоставляет большой выбор алгоритмов машинного обучения, таких как линейная регрессия, логистическая регрессия, деревья решений, случайные леса и т.д. Scikit-learn также включает функции для обработки данных и оценки моделей [5].

Количество инструментов для анализа данных в Python довольно велико, и к этому списку можно добавить еще множество других библиотек, таких как TensorFlow, Keras, PyTorch и т.д. Но Pandas, NumPy, Matplotlib, Seaborn и Scikit-learn являются основными библиотеками, нужными для работы с данными в Python.

Кластерный анализ – это мощный метод анализа данных, который позволяет группировать данные по их схожести. Он применяется в различных областях, таких как маркетинг, медицина, экономика и многих других. В этой статье мы рассмотрим разработку системы автоматизации кластерного анализа с использованием Python и основных библиотек для работы с данными – Pandas, NumPy, Matplotlib, Seaborn и Scikit-learn.

Рассматриваемая система автоматизации кластерного анализа может использоваться в сфере образования для анализа данных и определения шаблонов поведения учащихся.

Анализ поведения учащихся. Загрузив данные об активности и поведении учащихся в систему, можно использовать алгоритмы кластеризации для группировки учащихся по их поведению и определения тех, которые имеют общие черты в поведении. Например, система может выделить группы учащихся, которые предпочитают определенный вид деятельности, либо учащихся, которые более склонны к активности вне учебного процесса. Этот анализ поведения учащихся может помочь преподавателям находить новые методы работы с учащимися и идентифицировать тех, кто может потребовать дополнительной поддержки.

Анализ успеваемости учащихся. Алгоритмы кластеризации можно использовать для группировки учащихся по их успеваемости и поведению на занятиях. Это поможет преподавателям определить тенденции в поведении учащихся и адаптировать свой подход к обучению.

Шаг 1: Подготовка данных

Данные о успеваемости учащихся могут быть собраны из различных источников, таких как базы данных школы или университета, и могут включать оценки по различным предметам, информацию о пропущенных занятиях, уровне образования и т. д. Данные могут быть представлены в формате CSV, Excel или других форматах, и их можно загрузить в Python с использованием библиотеки Pandas.

Шаг 2: Обработка данных

Она может включать в себя удаление лишних столбцов, заполнение пропущенных значений, преобразование данных в нужный формат и т. д. Для этого мы можем использовать функции Pandas, такие как drop(), fillna() и другие (рис. 1).

```
# Удаление лишних столбцов
data = data.drop(['ID', 'Фамилия', 'Имя'], axis=1)

# Заполнение пропущенных значений
data = data.fillna(0)

# Преобразование данных в числовой формат
data['Математика'] = pd.to_numeric(data['Математика'])
data['Русский'] = pd.to_numeric(data['Русский'])
data['Английский'] = pd.to_numeric(data['Английский'])
data['Физика'] = pd.to_numeric(data['Физика'])
```

Рисунок 1 – Предварительная обработка данных

Шаг 3: Выполнение кластерного анализа

Для этого были использованы библиотеки NumPy, Matplotlib, Seaborn и Scikit-learn. NumPy предоставляет функции для работы с числовыми данными, Matplotlib и Seaborn – для визуализации результатов, а Scikit-learn – для выполнения кластерного анализа (рис. 2).

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

# Масштабирование данных
scaler = StandardScaler()
scaled_data = scaler.fit_transform(data)

# Выполнение кластерного анализа
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(scaled_data)

# Добавление меток кластеров в исходные данные
data['Cluster'] = kmeans.labels_

# Визуализация результатов
sns.scatterplot(x='Математика', y='Физика', hue='Cluster', data=data, palette='deep')
plt.xlabel('Оценка по математике')
plt.ylabel('Оценка по физике')
plt.title('Кластерный анализ успеваемости учащихся')
plt.show()
```

Рисунок 2 – Выполнение кластерного анализа

Шаг 4: Анализ результатов

После выполнения кластерного анализа мы можем проанализировать полученные результаты. Мы можем изучить характеристики каждого кластера, определить их особенности и сделать выводы. Например, мы можем проанализировать средние значения оценок в каждом кластере, построить графики распределения оценок и сравнить характеристики различных кластеров (рис. 3).

```
# Анализ характеристик кластеров
cluster_stats = data.groupby('Cluster').mean()
print(cluster_stats)

# Визуализация распределения оценок в различных кластерах
sns.histplot(data=data, x='Математика', hue='Cluster', multiple='stack', kde=True, palette='deep')
plt.xlabel('Оценка по математике')
plt.ylabel('Количество учащихся')
plt.title('Распределение оценок в различных кластерах')
plt.show()
```

Рисунок 3 – Анализ полученных результатов

Шаг 5: Интеграция системы автоматизации

Разработанную систему можно интегрировать в образовательный процесс. Например, результаты кластерного анализа могут быть использованы для выявления групп учащихся с различным уровнем успеваемости и предоставления им индивидуальных образовательных программ.

Кроме того, система автоматизации кластерного анализа может быть интегрирована в систему управления образовательным процессом (например, электронный журнал или платформу дистанционного обучения) для автоматического выполнения анализа и предоставления результатов учителям и администрации школы.

Преимущества системы автоматизации кластерного анализа успеваемости учащихся с использованием Python и библиотек Pandas, NumPy, Matplotlib, Seaborn и Scikit-learn:

1. Эффективность. Автоматизация кластерного анализа позволяет быстро обработать большие объемы данных о успеваемости учащихся и получить результаты в удобной форме для анализа.
2. Точность. Использование статистических алгоритмов кластерного анализа позволяет получить объективные результаты на основе математических методов, исключая субъективные предположения.
3. Гибкость. Система может быть настроена на основе специфических требований и целей

образовательного учреждения.

4. Визуализация. Использование библиотек Matplotlib и Seaborn позволяет визуализировать результаты анализа в удобной и понятной форме, что упрощает их интерпретацию и использование в практической работе.

5. Интеграция. Система может быть легко интегрирована в образовательный процесс, что позволяет автоматически выполнять анализ и использовать результаты для принятия решений и коррекции образовательных программ.

В заключение, разработка системы автоматизации кластерного анализа успеваемости учащихся с использованием Python предоставляет эффективный инструмент для анализа данных об успеваемости учащихся, определения групп с различным уровнем успеваемости и поддержки принятия решений в образовательном процессе. Это может значительно улучшить эффективность и точность оценки успеваемости учащихся, а также помочь учителям и администрации школы в принятии информированных решений по оптимизации образовательных программ, адаптации учебного процесса и оказанию дополнительной поддержки тем учащимся, которым она необходима.

Система позволяет быстро и точно провести анализ больших объемов данных, визуализировать результаты и использовать их для принятия решений и оптимизации образовательного процесса. Использование Python и указанных библиотек делает процесс разработки и интеграции системы относительно простым и гибким.

Список использованных источников

1. Официальная документация Pandas [Электронный ресурс]. – Режим доступа: https://pandas.pydata.org/docs/ – Дата доступа: 12.05.2023.
2. Официальная документация NumPy [Электронный ресурс] / Spring Framework – Режим доступа: https://numpy.org/doc/ – Дата доступа: 12.05.2023.
3. Официальная документация Matplotlib [Электронный ресурс]. – Режим доступа: https://pandas.pydata.org/docs/ – Дата доступа: 12.05.2023.
4. Официальная документация Seaborn [Электронный ресурс]. – Режим доступа: https://seaborn.pydata.org/tutorial.html. – Дата доступа: 12.05.2023.
5. Официальная документация Scikit-learn [Электронный ресурс]. – Режим доступа: https://scikit-learn.org/stable/documentation.html. – Дата доступа: 12.05.2023.