Практические задания на Python 3 для выполнения в Jupyter Lite
Задание 1: Линейная регрессия для прогнозирования цен на дома
Описание: Реализуйте модель линейной регрессии для прогнозирования цен на основе площади и количества комнат.
python

```
# Импорт необходимых библиотек
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

# Генерация данных (площадь, количество комнат, цена)
data = pd.DataFrame({
    'Площадь': [1200, 1500, 1700, 2000, 2500],
    'Комнаты': [3, 4, 3, 5, 4],
    'Цена': [300000, 400000, 350000, 500000, 600000]
})

# Разделение данных на признаки (X) и целевую переменную (y)
X = data[['Площадь', 'Комнаты']]
y = data['Цена']

# Разделение данных на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Создание модели линейной регрессии
model = LinearRegression()
model.fit(X_train, y_train)

# Прогнозирование цен для тестовой выборки
y_pred = model.predict(X_test)

# Визуализация результатов
plt.scatter(y_test, y_pred)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red',
label='Идеальная линия')
plt.xlabel('Фактические цены')
plt.ylabel('Предсказанные цены')
plt.legend()
plt.title('Сравнение фактических и предсказанных цен')
plt.show()

# Вывод коэффициентов модели
```

print("Коэффициенты модели:", model.coef_)
print("Свободный член:", model.intercept_)
Выводы:

- Модель линейной регрессии позволяет предсказывать цены на основе площади и количества комнат.
- Визуализация показывает точность предсказаний.

Задание 2: Кластеризация клиентов с использованием K-Means
Описание: Разделите клиентов на группы по их покупательскому поведению.
python
from sklearn.cluster **import** KMeans

 # Данные о клиентах (сумма покупок и частота покупок)
data = np.array([[1000, 2], [1500, 4], [2000, 6], [2500, 8], [3000, 10]])

 # Кластеризация с использованием K-Means (2 кластера)
kmeans = KMeans(n_clusters=2)
kmeans.fit(data)

 # Метки кластеров и центры кластеров
labels = kmeans.labels_
centers = kmeans.cluster_centers_

 # Визуализация кластеров
plt.scatter(data[:, 0], data[:, 1], c=labels, cmap='viridis', label='Клиенты')
plt.scatter(centers[:, 0], centers[:, 1], color='red', marker='x', label='Центры кластеров')
plt.xlabel('Сумма покупок ($)')
plt.ylabel('Частота покупок')
plt.legend()
plt.title('Кластеризация клиентов')
plt.show()
Выводы:

- Алгоритм K-Means успешно разделил клиентов на группы.
- Центры кластеров показывают средние значения для каждой группы.

Задание 3: Логистическая регрессия для классификации
Описание: Определите вероятность поступления студента на основе результатов двух экзаменов.
python
from sklearn.linear_model **import** LogisticRegression

 # Данные: результаты экзаменов и решение о поступлении (1 - поступил, 0 - не поступил)
X = np.array([[80, 85], [60, 70], [90, 95], [50, 55], [75, 80]])
y = np.array([1, 0, 1, 0, 1])

```
# Создание модели логистической регрессии
model = LogisticRegression()
model.fit(X, y)

# Прогнозирование вероятности поступления для нового студента
new_student = np.array([[70, 75]])
probability_of_admission = model.predict_proba(new_student)

print(f"Вероятность поступления нового студента:
{probability_of_admission[0][1]:.2f}")
```

Выводы:

- Логистическая регрессия позволяет оценить вероятность поступления.
- Вероятность зависит от результатов экзаменов.

Задание 4: Снижение размерности данных с помощью PCA

Описание: Уменьшите размерность данных для визуализации.

```
python
from sklearn.decomposition import PCA

# Данные: искусственный набор признаков (10 объектов с признаками
размерности 5)
data = np.random.rand(10, 5)

# Снижение размерности до двух признаков с использованием PCA
pca = PCA(n_components=2)
reduced_data = pca.fit_transform(data)

# Визуализация сниженной размерности данных
plt.scatter(reduced_data[:, 0], reduced_data[:, 1])
plt.xlabel('Признак 1')
plt.ylabel('Признак 2')
plt.title('Снижение размерности данных с помощью PCA')
plt.show()

print("Сниженная размерность данных:")
print(reduced_data)
```

Выводы:

- PCA позволяет уменьшить размерность данных без значительной потери информации.
- Сниженные данные можно визуализировать для анализа.

Заключение

Эти задания охватывают основные концепции машинного обучения — от регрессии до кластеризации и снижения размерности. Они адаптированы для выполнения в Jupyter Lite и включают визуализацию результатов.