Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

# ОТЧЕТ
# ПО ЛАБОРАТОРНОЙ РАБОТЕ №4
**дисциплины**
**«Искусственный интеллект и машинное обучение»**
**Вариант 13**

Выполнил:
Милькевич Александр Александрович
2 курс, группа ИТС-б-о-23-1,
11.03.02 «Инфокоммуникационные технологии и системы связи, направленность (профиль) «Интеллектуальная обработка данных в инфокоммуникационных системах и сетях», очная форма обучения

_____
(подпись)

Проверил:
Ассистент департамента цифровых, робототехнических систем и электроники Хацукова А.И

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2025 г.

**Тема: изучение структуры Serias и базовых операций**

**Цель:** познакомить с основами работы с библиотекой pandas, в частности, со структурой данных Series.

**Ход выполнения работы:**

Ссылка на гит хаб: https://github.com/lolndo/AI_Lab4.git

1. Создание Series из списка.

```
s1 = pd.Series([1, 2, 3, 4, 5],['a', 'b', 'c', 'd', 'e'])
print(s1)
```

```
a    1
b    2
c    3
d    4
e    5
dtype: int64
```

Рисунок 1. Выполнение задания 1

2. Получение элемента Series.

```
s2 = pd.Series([12, 24, 36, 48, 60],['A','B','C','D','E'])
print(s2.iloc[2])
print(s2.loc['C'])
```

```
36
36
```

Рисунок 2. Выполнение задания 2

3. Фильтрация данных с помощью логической индексации.

```
ndarr = np.array([4, 9, 16, 25, 36, 49, 64])
s3 = pd.Series(ndarr)
print(s3[s3>20])
```

```
3    25
4    36
5    49
6    64
dtype: int32
```

Рисунок 3. Выполнение задания 3

4. Просмотр первых и последних элементов.

```
nparr = np.random.randint(0,100,size=(50))
s4 = pd.Series(nparr)
print(f'первые 5:\n{s4.head(7)}\n\nпоследние 7:\n{s4.tail(7)}')
```

```
первые 5:
0    14
1    68
2    35
3     7
4     5
5    79
6    39
dtype: int32

последние 7:
43    20
44    39
45    42
46    49
47    92
48     3
49    45
dtype: int32
```

Рисунок 4. Выполнение задания 4

5. Просмотр первых и последних элементов.

```
animals = pd.Series(['cat', 'dog', 'rabbit', 'parrot', 'fish'])
print(animals.dtype)

#преобразование типа списка
animals = animals.astype('category')
print(animals.dtype)
```

```
object
category
```

Рисунок 5. Выполнение задания 5

6. Проверка пропущенных значений.

```
s6= pd.Series([1.2, np.nan, 3.4, np.nan, 5.6, 6.8])
print(f'Проверка на NaN:\n{s6.notnull()}')
nan_index = s6.index[s6.isnull()]
print(f'Индексы NaN элементов:\n{nan_index}')
```

```
Проверка на NaN:
0     True
1    False
2     True
3    False
4     True
5     True
dtype: bool
Индексы NaN элементов:
Index([1, 3], dtype='int64')
```

Рисунок 6. Выполнение задания 6

7. Заполнение пропущенных значений.

```
s7=s6.fillna(s6.mean())
print(f'Среднее значение всех непустых элемнтов:{s6.mean()}')
print(f'Новый ряд:\n{s7}')
```

```
Среднее значение всех непустых элемнтов:4.25
Новый ряд:
0    1.20
1    4.25
2    3.40
3    4.25
4    5.60
5    6.80
dtype: float64
```

Рисунок 7. Выполнение задания 7

8. Арифметические операции с series.

```
s81 = pd.Series([10, 20, 30, 40], ['a', 'b', 'c', 'd'])
s82 = pd.Series([5, 15, 25, 35], ['b', 'c', 'd', 'e'])
print(f'Результат сложения двух списков:\n{s81+s82}')
```

```
Результат сложения двух списков:
a     NaN
b    25.0
c    45.0
d    65.0
e     NaN
dtype: float64
```

```
print(f'Замена пустых значений на 0:\n{(s81+s82).fillna(0)}')
```

```
Замена пустых значений на 0:
a     0.0
b    25.0
c    45.0
d    65.0
e     0.0
dtype: float64
```

Рисунок 8. Выполнение задания 8

9. Применение функции к series.

```
s9 = pd.Series([2, 4, 6, 8, 10])
new_series = s9.apply(lambda x: np.sqrt(x))
print(f'Применение функции к series:\n{new_series}')
```

```
Применение функции к series:
0    1.414214
1    2.000000
2    2.449490
3    2.828427
4    3.162278
dtype: float64
```

Рисунок 9. Выполнение задания 9

## 10. Основные статистические методы

```
s10 = pd.Series(np.random.randint(50,150,size=20))
print(f'Сумма ряда: {s10.sum()}')
print(f'Среднее ряда: {s10.mean()}')
print(f'Максимальное значение: {s10.min()}')
print(f'Минимальноее значение: {s10.max()}')
print(f'Стандартное отклонение: {s10.std()}')
```

```
Сумма ряда: 2019
Среднее ряда: 100.95
Максимальное значение: 50
Минимальноее значение: 149
Стандартное отклонение: 30.37567413157062
```

```
series = pd.Series(np.random.uniform(10,100,size=10),index=pd.date_range(start='2024-03-01', periods=10, freq='D'))
print(f'Данные за 5-8 марта:\n{series.loc['2024-03-05':'2024-03-08']}')
```

```
Данные за 5-8 марта:
2024-03-05    26.399813
2024-03-06    42.838101
2024-03-07    92.705814
2024-03-08    43.579789
Freq: D, dtype: float64
```

Рисунок 10. Выполнение задания 10

## 11. Работа с временными рядами.

```
series = pd.Series(np.random.uniform(10,100,size=10),index=pd.date_range(start='2024-03-01', periods=10, freq='D'))
print(f'Данные за 5-8 марта:\n{series.loc['2024-03-05':'2024-03-08']}')
```

```
Данные за 5-8 марта:
2024-03-05    26.399813
2024-03-06    42.838101
2024-03-07    92.705814
2024-03-08    43.579789
Freq: D, dtype: float64
```

Рисунок 11. Выполнение задания 11

## 12. Работа с временными рядами

```
s12 = pd.Series([10, 20, 30, 40, 50, 60],['A', 'B', 'A', 'C', 'D', 'B'])
if s12.index.is_unique==False:
    s12 = s12.groupby(level=0).sum()
print(s12)
```

```
A    40
B    80
C    40
D    50
dtype: int64
```

Рисунок 12. Выполнение задания 12

13. Преобразование строковых дат в DatetimeIndex

```
s13 = pd.Series([100, 200, 300],index=['2024-03-10', '2024-03-11','2024-03-12'])
print(s13)
s13.index = pd.to_datetime(s13.index)
print(f'Тип данных индекса: {s13.index.dtype}')
```

```
2024-03-10    100
2024-03-11    200
2024-03-12    300
dtype: int64
Тип данных индекса: datetime64[ns]
```

Рисунок 13. Выполнение задания 13

14. Чтение данных из CSV-файла

```
%%writefile data.csv
Дата,Цена
2024-03-01,100
2024-03-02,110
2024-03-03,105
2024-03-04,120
2024-03-05,115
```

```
Writing data.csv
```

```
df = pd.read_csv('data.csv', parse_dates=['Дата'])
series = pd.Series(df['Цена'].values, index=df['Дата'])
print(series)
```

```
Дата
2024-03-01    100
2024-03-02    110
2024-03-03    105
2024-03-04    120
2024-03-05    115
dtype: int64
```

Рисунок 14. Выполнение задания 14

15. Построение графиков на основе series

```
import matplotlib.pyplot as plt
number = np.random.randint(50,150,size=30)
series = pd.Series(number,index=pd.date_range(start='2024-03-01', periods=30, freq='D'))
series.plot(kind='bar')
plt.title('Пример графика для Series')
plt.xlabel('Индекс')
plt.ylabel('Значения')
plt.grid(True)
plt.show()
```

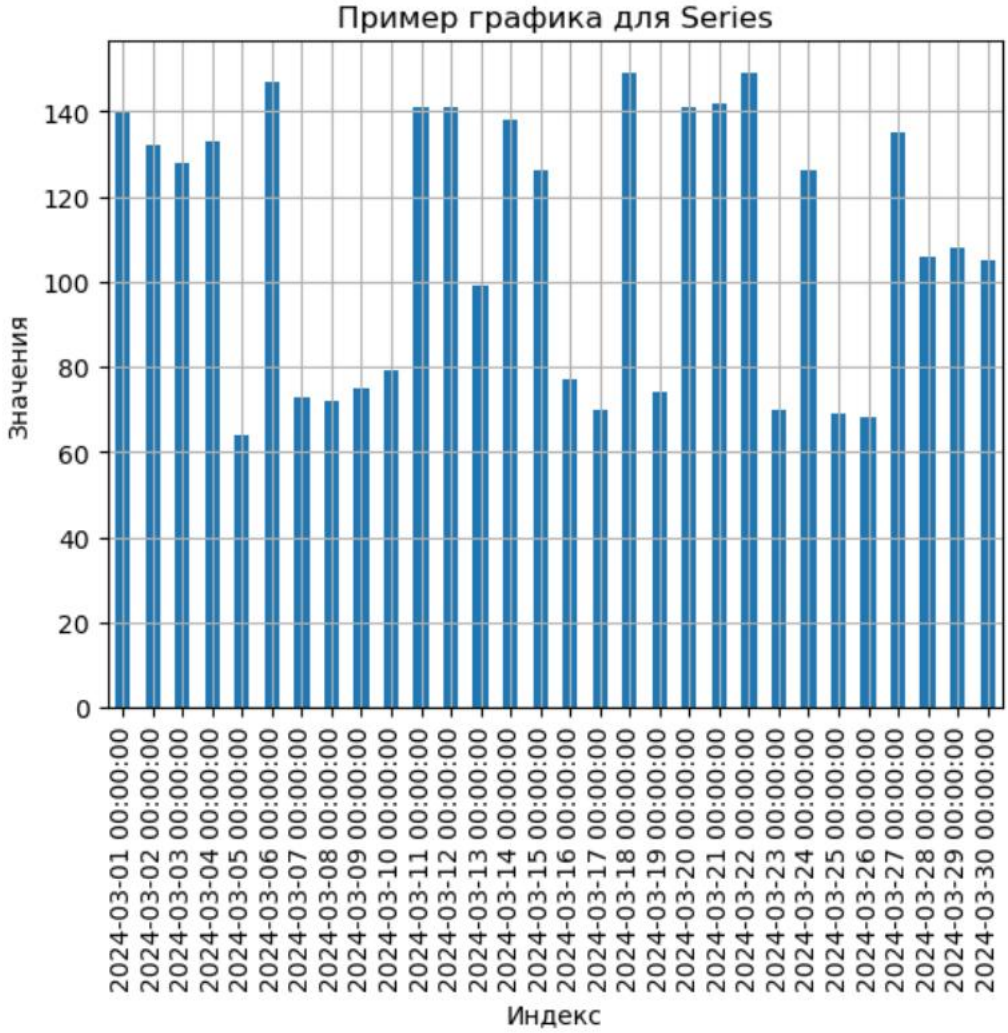


Рисунок 15. Выполнение задания 15

16. Создайте Series , где индексами будут даты с 1 по 30 августа 2024 года, а значениями – случайное количество подписчиков (от 5000 до 15000). Вычислите относительный прирост ( pct_change() ), отобразите на графике и выделите даты с приростом выше 5%.

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```python
stocks = np.random.randint(90,110,size=(20))
serias = pd.Series(stocks,pd.date_range("2025-07-01",periods=20, freq="D"))
print(serias)
```

```
2025-07-01    102
2025-07-02    102
2025-07-03    101
2025-07-04     94
2025-07-05    104
2025-07-06    101
2025-07-07    105
2025-07-08     99
2025-07-09     93
2025-07-10    104
2025-07-11    100
2025-07-12    106
2025-07-13    103
2025-07-14    107
2025-07-15     95
2025-07-16     90
2025-07-17     97
2025-07-18    108
2025-07-19    103
2025-07-20     94
Freq: D, dtype: int32
```

```python
difference = abs(serias.pct_change()*100)
dif_bol = abs(serias.pct_change()*100)>3
data = (serias[dif_bol])
dif_ser = pd.Series(difference,data)
print(dif_ser)
```

```
94    NaN
104   NaN
105   NaN
99    NaN
93    NaN
104   NaN
100   NaN
106   NaN
107   NaN
95    NaN
90    NaN
97    NaN
108   NaN
103   NaN
94    NaN
dtype: float64
```

```python
plt.bar(serias.index, serias.values, color='skyblue', label='Значения')
plt.title('Пример графика для Series')
plt.xlabel('Индекс')
plt.ylabel('Значения')
plt.grid(True)
plt.plot(data.index, data.values, 'ro', markersize=5, label='Маркеры')
plt.show()
```

Рисунок 16. Листинг индивидуального задания


Рисунок 17. График

**Вывод**: в ходе лабораторной работы были изучены структуры Serias и базовые операции по работе с ними.

Контрольные вопросы:

1. Что такое pandas.Series и чем она отличается от списка в Python

pandas.Series - это одномерный массив, способный хранить данные любого типа (целые числа, строки, числа с плавающей запятой и т.д.) и имеет ассоциированный с ним индекс. В отличие от списка, Series предоставляет больше возможностей для анализа данных, а также поддерживает метаданные.

2. Какие типы данных можно использовать для создания Series

Можно использовать целые числа, числа с плавающей запятой, строки, булевы значения и даже объекты Python, такие как списки или массивы.

3. Как задать индексы при создании Series

Индексы можно задать, передав аргумент index в конструктор Series, например: pd.Series(data, index=[0, 1, 2]).

4. Каким образом можно обратиться к элементу Series по его индексу

Можно использовать квадратные скобки: series[index], где index - это индекс искомого элемента.

5. В чём разница между .iloc и .loc при индексации Series

.iloc[] используется для индексации по целым числам (позиций), а .loc[] — по именованным индексам.

6. Как использовать логическую индексацию в Series

Можно использовать логическое выражение, чтобы создать маску: series[series > threshold], где threshold - это пороговое значение.

7. Какие методы можно использовать для просмотра первых и последних элементов Series

Методы .head() и .tail().

8. Как проверить тип данных элементов Series

Метод .dtype возвращает тип данных элементов Series.

9. Каким способом можно изменить тип данных Series

Используйте метод .astype(new_type), чтобы изменить тип данных.

10. Как проверить наличие пропущенных значений в Series

Метод .isnull().any() поможет выявить пропущенные значения.

11. Какие методы используются для заполнения пропущенных значений в Series

Методы .fillna(value) и .interpolate().

12. Чем отличается метод .fillna() от .dropna()

.fillna() заменяет пропущенные значения заданным значением, тогда как .dropna() удаляет элементы с пропущенными значениями.

13. Какие математические операции можно выполнять с Series

Можно выполнять арифметические операции, такие как сложение, вычитание, умножение и деление.

14. В чём преимущество векторизированных операций по сравнению с циклами Python

Векторизированные операции значительно быстрее за счёт оптимизации на уровне C, тогда как циклы в Python медленнее из-за интерпретируемого характера языка.

15. Как применить пользовательскую функцию к каждому элементу Series

Используйте метод .apply(func).

16. Какие агрегирующие функции доступны в Series

Сумма, среднее значение, максимальное и минимальное значения, стандартное отклонение и т.д.

17. Как узнать минимальное, максимальное, среднее и стандартное отклонение Series

Методы .min(), .max(), .mean(), .std().

18. Как сортировать Series по значениям и по индексам

Методы .sort_values() и .sort_index().

19. Как проверить, являются ли индексы Series уникальными

Метод .is_unique вернет True, если индексы уникальны.

20. Как сбросить индексы Series и сделать их числовыми

Используйте метод .reset_index(drop=True).

21. Как можно задать новый индекс в Series

Присвойте новый индекс через series.index = new_index.

22. Как работать с временными рядами в Series

Используйте pd.to_datetime() и методы для обработки временных данных, такие как .resample().

23. Как преобразовать строковые даты в формат DatetimeIndex

Используйте pd.to_datetime(series).

24. Каким образом можно выбрать данные за определённый временной диапазон

Используйте .loc[start_date:end_date].

25. Как загрузить данные из CSV-файла в Series

Используйте метод pd.read_csv('file.csv')['column_name'].

26. Как установить один из столбцов CSV-файла в качестве индекса Series

При загрузке используйте параметр index_col='column_name'.

27. Для чего используется метод .rolling().mean() в Series

Метод рассчитывает скользящее среднее для заданного окна.